RÉPUBLIQUE ALGÉRIENNE DÉMOCRATIQUE ET POPULAIRE

Ministère de l'Enseignement Supérieur et de la Recherche Scientifique

Université A.Mira, Béjaïa

Faculté des Sciences Exactes

Département d'Informatique

Mémoire de fin de Cycle

Pour l'Obtention du Diplôme Master professionnel en Informatique

Option : Administration et Sécurité des Réseaux Informatiques

Thème

Conception et réalisation d'une application mobile sensible au contexte pour un musée

Présentée par :

BOUDJADI Fatima & CHEKLAT Lamia

Juillet 2013

Présidente du jury :	Mme DJERROUD Souhila, Maître assistant, Université de Béjaïa.
Examinateurs :	Mme GHANEM Souhila, Maître assistant, Université de Béjaïa.
	Mme KOUICEM Amel, Maître assistante, Université de Béjaïa.
Encadrant :	Mme SABRI Salima, Maître assistante, Université de Béjaïa.
Co-Encadrant :	Mme YAICI Malika, Maître assistante, Université de Béjaïa.

Remerciements

***N**os vifs remerciements vont d'emblée à Dieu le tout puissant qui nous a doté d'une grande volonté et d'un savoir adéquat pour mener à bien cet humble travail.*

***N**ous adressons nos remerciements tout particulièrement :*

***Á** nos chers parents, notre fierté et bien sur la source de notre réussite car ils se sont sacrifiés pour nous fournir une atmosphère de travail disposant de toutes les meilleures conditions, sans eux rien n'aurait pu être facile, que dieux nous les garde et les protège afin que l'on puisse leurs rendre un peu du beaucoup qu'ils nous ont procuré.*

***Á** notre Encadreur et Co-Encadreur en l'occurrence Melle SABRI Salima et Mme YAICI Malika pour leurs disponibilités, savoir-faire et leurs soutiens, elles nous ont inculqué une grande confiance et elles nous ont orienté dans le bon sens quant à l'élaboration de ce projet.*

***A**ux membres du jury qui ont accepté d'évaluer notre travail.*

***Á** nos très chers frères, soeurs, ami (es) et à toutes les personnes qui ont contribué à la réussite de ce travail.*

***E**nfin, nous tenons à remercier toute la promotion 2012-2013 master 2 informatique recherche et professionnel. Ainsi que tous nos enseignants et les membres du département informatique de Université A/Mira, Bejaïa.*

Dédicaces

Je dédie ce modeste travail qui aurait pu aboutir et voir la lumière avec l'aide de dieu le tout puissant.

***Á** ma très chère honorable et aimable mère qui représentes pour moi le symbole de la bonté par excellence, la source de tendresse et l'exemple du dévouement qui n'a pas cessé de m'encourager et de prier pour moi..*

***Á** mon très cher père. Rien au monde ne vaut les efforts fournis jour et nuit pour mon education et mon bien être. Ce travail est le fruit de tes sacrifices que tu as consentis pour mon éducation et ma formation.*

***E**nsemble vous avez su m'encourager et me soutenir tout au long de mes études. Je dirai donc qu'aucune dédicace ne saurait être assez éloquente pour exprimer ce que vous méritez pour tous les sacrifices que vous n'avez cessé de me faire depuis ma naissance, durant mon enfance et même à l'âge adulte. Vous représentez la lumière de mon existence et l'étoile brillante de mon réjouissance. Que dieu vous protège.*

***Á**mes très chère grands parents qui sont mes deuxième parent que dieux me les gardes.*

***Á**mes frères Nabil, Hakim, ma soeur Meriem, ma belle-soeur Zahia ainsi que toute sa famille, sans oublie bien sur ma petite Cérine adorée ainsi que toute sa famille.*

***Á**ma camarade Lamia qui m'a supporté tout au long de la réalisation de ce travail ainsi qu'a toute sa famille.*

***Á** l'équipe du club sportif technologie en particulier Salim et khelil.*

***Á** l'équipe du club sportif SNV en particulier Amine, Nacer et Hamou.*

***Á**mon équipe de volley-ball SNV en particulier Ibtissem, Meriem, Narimen.*

***Á** mes anciens collègues de travail principalement Mr BEGHAL.S et Mr LAHDIRI.A*

***Á**mes nouveaux collègues de travail en particulier ma directrice Mme BELKHIR .D.*

***Á** mes amis (es) Sara, Kahina, SAbiha, Zahra, Nabil,Mourad,Samir.*

***Á** deux personnes particulières à savoir Mr HANI.H et Mr KICHER.Z.*

***Á** vous mon Encadreur et Co-Encadreur pour votre aide précieuse.*

***Á** toute ma famille, à tous mes amis et aux étudiants de notre promotion.*

Fatima.B.

Dédicaces

Je dédie ce modeste travail qui aurait pu aboutir et voir la lumière avec l'aide de dieu le tout puissant.

***Á** la mémoire de mon oncle Malek que j'aime beaucoup et que je n'oublierai jamais que Dieu l'accueille dans son vaste paradis.*

***Á** ma très chère honorable et aimable mère qui représentes pour moi le symbole de la bonté par excellence, la source de tendresse et l'exemple du dévouement qui n'a pas cessé de m'encourager et de prier pour moi..*

***Á** mon très cher père. Rien au monde ne vaut les efforts fournis jour et nuit pour mon education et mon bien être. Ce travail est le fruit de tes sacrifices que tu as consentis pour mon éducation et ma formation.*

***E**nsemble vous avez su m'encourager et me soutenir tout au long de mes études. Je dirai donc qu'aucune dédicace ne saurait être assez éloquente pour exprimer ce que vous méritez pour tous les sacrifices que vous n'avez cessé de me faire depuis ma naissance, durant mon enfance et même à l'âge adulte. Vous représentez la lumière de mon existence et l'étoile brillante de mon réjouissance. Que dieu vous protège.*

***Á**mes grands-parents que j'aime beaucoup, que dieux me les garde.*

***Á** mes adorables frères Lamine, Toufik et ma chère soeur Meriem.*

***Á**mes oncles Hakim et Mourad ainsi que leurs familles.*

***Á** toutes mes tantes Nora, sonia et Fadila.*

***Á**ma camarade Fatima qui m'a supporté tout au long de la réalisation de ce travail ainsi qu'a toute sa famille.*

***Á** mes amis (es) Amel, lamia, sihem, Sonia, Warda, Sara, Tiziri, louanas, Rabah ,Khoudir,Saber,Sofia,Nadir, Moumen,Rafik.*

***Á** vous mon Encadreur et Co-Encadreur pour votre aide précieuse.*

Á *toute ma famille, à tous mes amis et aux étudiants de notre promotion.*

Lamia.C.

Table des matières

Table des figures

Liste des tableaux

Introduction générale

L'avènement de l'informatique a donné un nouvel élan à une révolution qui a changé complètement la façon dont nous accédons à l'information. Cette révolution a engendré une évolution considérable des technologies et des habitudes d'utilisation de l'informatique. Désormais, les technologies de l'information et de la communication apparaissent de plus en plus dans notre vie de tous les jours.

La propagation des moyens de communication sans-fil couplés aux avancées technologiques matérielles permettent à présent de proposer des dispositifs portables qui tiennent dans la main, dotés d'une capacité de calcul et de moyens de communication sans-fil tels que les mini PC, les PDA, les téléphones portables et même les capteurs sans-fil.

Toutes ces innovations en termes de technologies sans fils, de systèmes embarqués et de miniaturisation de périphériques ont convergé vers la vision de l'informatique du 21ème siècle qu'introduisait Mark Weiser en 1991 dans son article intitulé **'The Computer for the 21st Century'** [1] où les systèmes informatiques sont omniprésents et capables de nous assister dans nos tâches quotidiennes. Cette vision est connue sous le nom de l'informatique ubiquitaire (en anglais "ubiquitous computing").

La première vague de l'informatique ubiquitaire a été l'accès à l'information par le grand public en tout lieu. Cela a été rendu possible par le nombre de plus en plus croissants des ordinateurs. Ensuite une nouvelle ère de l'informatique apparait : l'utilisation transparente de l'informatique appelée également informatique diffuse (en anglais "pervasive computing"). L'enjeu de cette évolution est de rendre l'informatique invisible, comme Mark Weiser affirma dans son article [1] de telle sorte que les interactions des systèmes informatiques avec l'extérieur changent de nature. En effet, nous allons donc passer d'une interaction de type "bureau", un-vers-un, vers une autre de type ubiquitaire,

plusieurs vers- plusieurs. Au lieu de nous asseoir face à un ordinateur pour lui demander des actions, une multitude d'ordinateurs, de capteurs et d'actionneurs vont interagir autour et avec nous dans nos bâtiments, dans nos moyens de transport, dans nos rues, etc. Pour désigner ces espaces riches en dispositifs et en interactions on parle par abus de langage d'environnements ubiquitaires. Ce type d'environnement se caractérise par une variabilité des capacités matérielles (mémoire, bande passante, batterie, etc.), de plus la mobilité de l'utilisateur introduit une variabilité de l'environnement qui entoure l'application. Par conséquent, les applications traditionnelles ne peuvent s'exécuter dans ce type d'environnement. Donc, il est nécessaire de considérer un nouveau type d'applications qui détectent les changements dans l'environnement et qui tirent parti de ces changements pour adapter leurs comportements en conséquence. Ce type d'applications est appelé applications sensibles au contexte.

La notion de sensibilité au contexte (en anglais : context-awareness) a été introduite dans le cadre des recherches sur l'informatique ubiquitaire. Schilit et Theimer [22] furent les premiers à définir cette notion : il s'agit de la capacité d'un système à découvrir et à réagir à des changements dans l'environnement où il se trouve.

De ce fait, l'un des objectifs important de l'informatique ubiquitaire et de pouvoir fournir une multitude de services aux utilisateurs en fonction de leur contexte courant (au bon moment et au bon endroit). Cela nécessite donc de gérer le contexte utilisateur d'une manière à ce qu'il sera accessible par les applications afin de détecter les services en question selon le contexte. C'est dans cette optique que s'inscrit notre projet.

L'objectif de notre travail est alors de développer une application mobile sensible au contexte. Cette dernière sera en mesure de découvrir des services et les adapter selon le contexte des utilisateurs afin de parvenir à satisfaire leur besoins avec un minimum d'action de leur part.

Pour comprendre l'intérêt résidant dans le fait qu'une application soit capable de détecter et prendre en compte l'environnement d'un utilisateur nous avons pris l'exemple d'une personne qui aimerait visiter un musée. On met à sa disposition une application mobile qui l'aiderait à se guider durant sa visite. Une telle application, si elle n'était pas consciente de l'environnement de l'utilisateur, se résumerait à une simple carte touristique couplée à un guide touristique sous forme de livre. Cependant, en la rendant capable

d'exploiter des informations comme la position de l'utilisateur et ses préférences, elle pourrait indiquer par exemple à un handicapé à chaise roulante son itinéraire à l'intérieur du musée où d'augmenter les effets de la lumière s'il s'agit d'un visiteur mal voyant.

Á la suite de ce préambule notre mémoire se divise principalement en : partie théorique et partie développement de l'application.

La partie théorique est axée sur trois chapitres :

Le premier chapitre présentera des généralités sur les systèmes ubiquitaires, à savoir : la définition de l'informatique ubiquitaire, son environnement, ses caractéristiques et ses contraintes, les équipements de l'environnement ubiquitaire, et nous citerons quelques domaines d'applications et les dernières avancées technologiques.

Dans le deuxième chapitre, nous allons aborder la notion du contexte, la notion de la sensibilité au contexte et quelques architectures sensible au contexte.

Le troisième chapitre sera consacré pour les appareils et les applications mobiles, nous citerons aussi quelques exemples d'applications et les principales plateformes de développement.

La deuxième partie quant à elle regroupe deux chapitres :

Le chapitre conception, là où nous entamerons la modélisation, mais d'abord nous introduisons la méthode utilisée et ses différentes étapes. Pour ce qui est conception notre choix se portera sur le formalisme UML suivant une démarche du processus unifié de Rational, le choix d'UML est motivé par sa flexibilité marquante.

Dans le dernier chapitre, nous y détaillerons la partie réalisation : les outils, logiciels et environnement de développement. Nous terminerons ce chapitre avec des prises d'écran de l'application

Enfin nous allons terminer par une conclusion générale.

Chapitre

1 Généralités sur les systèmes ubiquitaires

1.1 Introduction

Le développement grandissant de l'informatique et le besoin d'accès à l'information de partout et à tout instant, a fait apparaître le concept de l'**informatique ubiquitaire**. L'évolution de cette dernière a mis en gestation une nouvelle ère de l'informatique, baptisée, **informatique diffuse** qui vise à se tisser d'une manière transparente dans la vie quotidienne du grand public afin de subvenir à ses diverses services.

Pour mieux assimiler ces notions, nous allons aborder dans ce premier chapitre ces principales généralités à savoir : définition de l'informatique et du système ubiquitaire ainsi que son environnement, ses caractéristiques et ses contraintes, les équipements de l'environnement ubiquitaire là où on va définir l'environnement matériel et l'environnement logiciel, ensuite nous allons évoquer les différents domaines d'applications de l'informatique ubiquitaire, pour enfin terminer par une conclusion.

1.2 Informatique ubiquitaire

L'informatique ubiquitaire (en anglais : ubiquitous computing or ubicom) est le terme associé à la troisième ère de l'informatique moderne. Ce terme fut inventé par Mark Weiser en 1988 au Xerox PARC (Palo Alto Research Center) aux États unis, pendant qu'il était le directeur du laboratoire des sciences de l'informatique (CLS : Computer Science Laboratory), l'un des cinq laboratoires du célèbre centre de recherche Xerox PAR[15].

Mark Weiser [1] a introduit le concept d'informatique ubiquitaire pour marquer sa vision futuriste de l'informatique du XXIe siècle, en envisageant un monde largement

peuplé d'objets informatiques et numériques qui seraient reliés en réseaux à très grande échelle et interagiraient de manière autonome et transparente (communication directe entre machines) afin d'accomplir diverses tâches de la vie quotidienne. En effet, les utilisateurs doivent être en mesure de bénéficier des nouvelles technologies (les technologies de l'information et de la communication) le plus naturellement possible ; n'importe où, à tout instant et à partir de divers dispositifs, en adoptant ces technologies et en les mettant à leurs égard.

La vision de Mark Weiser est devenue aujourd'hui possible grâce à la combinaison de trois facteurs principaux : la miniaturisation et la puissance des composants électroniques, la chute des coûts de production, et l'omniprésence des technologies réseaux sans fil[2].

Il est à noter que les adjectifs "ubiquitaire", "Pervasif", "diffus" et "ambiant" sont assez proches. Les termes "ubiquitous computing" et "pervasive computing" sont considérés comme équivalents et leur définition peur être tirée de la fameuse citation du Mark Wieser introduite en 1991 dans son article publié dans Scientific American article en1991 [1,3] : *"Les technologies les plus profondes sont celles qui apparaissent. Elles se mêlent dans le tissu de la vie de tous les jours jusqu'au point où elles en deviennent indiscernable"*.

Pour ce qui nous concerne, nous utiliserons ces adjectifs avec les nuances suivantes [2] :

- **Ambiant :** Incorporé dans les objets quotidiens (Les ordinateurs embarqués dans des voitures, les appareils électroménagers, etc.) ;
- **Mobile :** Mise en place des dispositifs mobiles ;
- **Ubiquitaire :** c'est un terme qui veut à la fois dire que l'informatique est présente partout (ce qui recouvre le terme Pervasif) et à la fois que l'utilisateur a accès à ses applications de tout endroit et de tout dispositif, et qu'ainsi il peut intervenir de tout endroit sur des équipements distants (ce qui recouvre le terme de mobilité) ;
- **Pervasif (terme anglicisme) :** représente l'informatique omniprésente dans le sens où tout objet de la vie courante peut contenir des systèmes informatiques. L'informatique pervasive est utile pour les humains sans qu'ils aient à en être conscients et sans que cela leur demande des connaissances particulières ;
- **Diffuse :** c'est la traduction française de Pervasif mais ce terme est aujourd'hui assez peu utilisé ;

– **Sensibilité aux contextes :** Prise en compte des paramètres de l'environnement de l'application.

1.3 Les systèmes informatiques ubiquitaires

Le concept du système informatique ubiquitaire se définit comme étant l'Informatique pervasive utilisée par l'ensemble des équipements miniaturisés émergés dans notre environnement quotidien[4].

Grâce aux différents objets communiquants disponibles autour de nous, un système d'informatique ubiquitaire permet d'automatiser certaines tâches quotidiennes. Le principe de fonctionnement d'un tel système est le suivant [2] :

– Premièrement collecter les informations de l'environnement physique (lumière du soleil, présence d'un utilisateur) par les objets communiquants du système (capteurs, actionneurs) ;
– Deuxièmement Interpréter, filtrer et agréger les informations collectées afin d'obtenir des informations de plus haut niveau ;
– Au final certaines applications peuvent décider des actions à entreprendre (déclencher une alarme, augmenter le volume, etc.).

On peut schématiser ce fonctionnement comme suit :

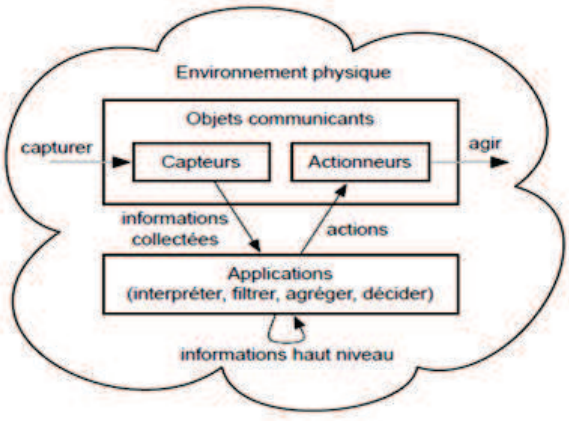


FIGURE 1.1 – Le fonctionnement d'un système informatique ubiquitaire. [2]

Les systèmes d'informatique ubiquitaire reposent sur deux notions centrales [2] :

- Le contexte à partir duquel ils réagissent et s'adaptent aux différents changements de l'environnement ;
- La découverte de services pour trouver quels objets communiquants sont disponibles dans l'environnement.

1.4 Environnements ubiquitaires

Un environnement ubiquitaire est un fonctionnement de la communication où les objets communicants se reconnaissent et se localisent automatiquement entre eux. Les objets interagissent entre eux sans action particulière de l'utilisateur[5].

1.5 Caractéristiques d'un environnement ubiquitaire

Les systèmes ubiquitaires se caractérisent par :

- **Ressources miniaturisées :** comme les téléphones portables et les PDA.
- **Distribution :** les terminaux dans un environnement ubiquitaire se trouvent dans des endroits différents, ils s'interconnectent entre eux via des réseaux filaires ou sans files en utilisant des infrastructures (middlewares) de communication permettant d'exposer des services, les exécuter, et les échanger entre les différents dispositifs en cas de nécessité. Ce qui fait de leur nature des systèmes ubiquitaires distribués[6].
- **Dynamicité :** elle est due au fait que les entités qui les composent sont mobiles. En effet, l'utilisation des réseaux sans fils accompagnée d'un déplacement incontrôlables des utilisateurs, engendre une dynamicité au niveau des dispositifs et de qualité de service proposé. De nouveaux dispositifs et services peuvent apparaître et d'autres disparaître[7].
- **Mobilité :** le fait d'offrir une mobilité, représente l'atout décisif dans les systèmes ubiquitaires. Les terminaux doivent être susceptibles d'accéder aux services quels que soient leurs vitesses de déplacement et l'endroit où ils se trouvent[8,13].

– **Hétérogénéité :** les services offerts par les composants matériels tels que les ordinateurs portables, les capteurs et les PDA sont équivalents dans les systèmes Pervasifs mais la différence peut résider soit au niveau des composants logiciels, au niveau des architectures matérielles ou au niveau de l'infrastructure de communication.

. L'hétérogénéité de communication : afin d'assurer une communication entre les composants d'un système ubiquitaire interconnectés via une connexion hybride, contenant aussi bien des connexions filaires ou connexions wifi, la mise une place d'une passerelle est indispensable. En effet, pour qu'une communication entre la norme IEEE 802.3 et la norme IEEE 802.11 s'établisse, la présence d'une passerelle est obligatoire[8].

. L'hétérogénéité au niveau des composants logiciels : la diversité des environnements de développement des systèmes ubiquitaires tel que (java, c, etc.) a créé une sorte d'hétérogénéité qui a engendré éventuellement une diversité des systèmes d'exploitations nécessaires à l'exécution des programmes[6, 8].

. L'hétérogénéité au niveau des architectures matérielles : elle est due à l'évolution de l'architecture de chaque dispositif, ainsi chacune d'elles dispose d'une configuration matérielle différente appropriée à elle-même[8].

– **Sensibilité aux contextes :** la sensibilité au contexte est reconnue comme étant une caractéristique importante de l'informatique ubiquitaire. Une application sensible au contexte doit s'adapter dynamiquement aux changements dans le contexte. Cela implique que dans différents contextes, les utilisateurs accèdent aux mêmes données et aux mêmes services mais reçoivent des réponses qui peuvent être différentes ou présentées différemment ou à des niveaux de détails différents[6].

1.6 Contraintes de l'environnement ubiquitaire

Le développement d'un environnement ubiquitaire est une tâche difficile, qui requiert à la fois de faire face aux [6,8] :

– Contraintes de ressources : le développement des applications mobiles est influencé par la limitation de capacités des équipements utilisés dans l'environnement mobile (téléphone portable, PDA, etc.), y compris leurs capacités d'affichage et leur puissance

de calcul ;

- Limitations réseaux : le problème majeur d'utilisation des réseaux sans fils dans les dispositifs mobiles consiste à l'utilisation d'une bande passante limitée et instable par rapport aux réseaux filaires. En effet, l'utilisation des technologies standards sans fils (Bluetooth, Wifi, GPRS, IEEE 802.a, IEEE 802.b, IEEE 802.g) ralentissent l'exécution de grandes applications et l'échange de données volumineuses entre mobiles, ce qui engendre la dégradation en terme de qualité de service offert à l'utilisateur ;
- Limites d'hétérogénéité des matériels et logiciels pour assurer la communication de ces derniers ;
- Limites de la mobilité en mettant en place des dispositifs de gestion pour garantir la continuité des services proposés ;
- Changements du contexte pour pouvoir agir sur toute sorte de modifications pour le maintien de la qualité du service rendu.

1.7 Equipement d'un environnement ubiquitaire

Un environnement ubiquitaire est subdivisé principalement en deux parties : l'environnement matériel et l'environnement logiciel.

1.7.1 Environnement matériel

Les technologies croissantes des différents dispositifs informatiques et des réseaux sans fils ont fortement contribué à l'épanouissement de l'informatique ubiquitaire[9].

1.7.1.1 Les réseaux sans fils

Un réseau sans fil (Wireless network) est un réseau dans lequel au moins deux périphériques peuvent communiquer sans liaison filaire, en se basant sur des liaisons utilisant des ondes radios électroniques. Les réseaux sans fils peuvent être classés en :

1. **Les réseaux personnels sans fil (WPAN : Wireless Personal Area Network) :** Ce sont des réseaux de faible porté (quelques dizaines de mètres) qui servent à relier des périphériques (imprimante, téléphone portable, PDA, etc.) à un

ordinateur sans avoir recours à des liaisons filaires. L'une des meilleures technologies de ces réseaux et la norme IEEE 802.15.1 (Bluetooth).

2. **Les réseaux locaux sans fil (WLAN : Wireless Local Area Network) :** Ce sont des réseaux permettant de couvrir l'équivalent d'un réseau local d'entreprise, qui servent à relier des terminaux présents dans la zone de couverture. Plusieurs technologies courantes existent (exemple, wifi : IEEE 802.11).
3. **Les réseaux métropolitains sans fil (WMAN : Wireless Métropolitan Area Network) :** Ce sont des réseaux permettant de couvrir un rayon d'une vingtaine de kilomètres, basés sur la norme IEEE 802.16 et offrent un débit de 10Mbit/S. Ils sont destinés principalement aux opérateurs de télécommunication.
4. **Les réseaux étendus sans fil (WWAN : Wireless Wide Area Network) :** Ce sont des réseaux dédiés à la téléphonie mobile et sont connus sous le nom des réseaux cellulaires. Divers générations existent, on peut évoquer :
   - **1G(première génération de téléphonie mobile) :** c'est un téléphone qui possède un fonctionnement analogique rendu obsolète par l'avènement de la 2G.
   - **2G(deuxième génération de téléphonie mobile) :** c'est un téléphone marqué par le passage du fonctionnement analogique en numérique. Utilisables dans les transmissions de voix et de données numériques de faible volume Ces principaux standards sont : GSM, CDMA, TDMA.
   - **3G(troisième génération de téléphonie mobile) :** c'est un téléphone qui dispose d'un usage multimédia tel que la transmission des vidéos, la visio-conférence ou l'accès à Internet haut débit.

1.7.1.2 Les dispositifs mobiles

Ce sont des dispositifs de poche (handheld) pouvant être qualifiés d'ordinateurs et disposants d'une certaine forme de connectivité (Bluetooth, Wifi, Wimax, GSM, etc.). On peut citer [8,9] :

1. **Le PDA (Personnel Digital Assistant) :** c'est un ordinateur de poche de taille réduite mené d'un processeur, d'un écran tactile et de fonctionnalités réseaux praticable pour ses fonctions d'agenda, de bloc note et de lecture MP3.

2. **Le Smartphone (téléphone portable couplés au PDA) :** développe plus de fonctionnalités à savoir : la navigation Web, consultation de courriers électroniques, visualisation des fichiers PDF, etc.
3. **La tablette PC :** c'est un ultraportable disposant d'un stylet qui nous sert d'écrire ou de dessiner manuellement et directement sur l'écran.

1.7.2 Environnement logiciel

Des intergiciels (en anglais : middleware) ont été élaborés afin de remédier aux problèmes de communication issus de l'hétérogénéité des dispositifs qui se trouvent dans l'environnement ubiquitaire, ils désignent la couche logicielle intermédiaire qui fournit une alternative très prometteuse dans le sens où elle doit [7, 8] :

– Cacher la répartition en la rendant transparente à travers l'interconnexion des différents dispositifs ou applications distribués ;
– Cacher l'hétérogénéité des composants matériels, des systèmes d'exploitation et des protocoles de communication utilisés par les différentes parties d'une application ;
– Fournir un ensemble de services communs réalisant des fonctions d'intérêt général pour faciliter la coopération entre applications.

1.8 Domaines d'application

L'informatique ubiquitaire a vu son évolution se tisser dans divers domaines. Nous présenterons ci-dessous un petit aperçu de ces derniers.

1.8.1 Domaine d'apprentissage (E-learning)

Apprendre et enseigner dépendent de la création, le maintien et l'expansion d'une communauté de recherche pratique.

L'apprentissage omniprésent est basé sur la reconnaissance des environnements d'apprentissage ubiquitaires ULE (Ubiquitous Learning Environnement) qui sont des environnements intégrés à de multiples dimensions : physiques, milieux sociaux, informationnels et techniques, ils permettent une intégration et une coordination plus lisse entre

les établissements d'enseignement (les écoles, les universités, les instituts de recherche en éducation, etc.), des espaces de travail, des communautés et des familles[14].

Dans un ULE, de nombreux dispositifs distribués et une variété d'objets intelligents (capteur) avec des ordinateurs encastrés seront utilisés. Ces dispositifs vont de très petite taille comme les lecteurs RFID à puce et à capteurs, à la moyenne tels que les PDA, les téléphones portables, et à la grande taille comme les ordinateurs portables et les PC[14].

1.8.2 Domaine publique

Le principe est d'accompagner le grand public dans leur vie quotidienne, et de leur offrir une multitude de services selon l'instant et l'endroit où ils se trouvent grâce à des capteurs. La diversité des services existants dans le système ubiquitaire et le contexte dynamique des utilisateurs exigent la mise en place d'une méthode permettant de recommander des services à l'utilisateur en fonction de ses préférences. L'une des alternatives suggérées est DUPS (Dimension User Profil System), qui préjuge les services qui seront utilisés à l'avenir par les utilisateurs selon leurs profils[10].

1.8.3 Domaine sanitaire

Les soins de santé à l'époque de l'informatique ubiquitaire est une innovation médicale personnalisée à l'avenir (intégration future des équipements informatiques dans tous ce qui est relatif aux soins). En effet, elle joue un rôle important dans l'assistance des malades n'importe quand et n'importe où, afin qu'un meilleur niveau de vie puisse être atteint. En établissant une connexion à distance entre un médecin et son patient, les maladies seront ainsi détectées et diagnostiquées en temps réel ce qui permettra l'intervention rapide de l'équipe de soin (médecins, infirmiers, etc.) et augmentera les chances de vie des patients. [11]

De nombreuses recherches ont été menées par des instituts et des universités. Toshiba Corporation au Japon a développé un "LifeMinder" qui est un système de soins de santé qui peut fournir des informations en temps réel en fonction du contexte actuel du patient. Il est composé d'un module en forme de montre-bracelet (détecteur portable) et d'un PDA. Le module de capteur mesure les paramètres physiologiques sur 3 axes :

l'accélération du sang, le pouls et la température de la peau. Le PDA est utilisé pour reconnaître les activités du patient telles que marcher et manger en analysant les données recueillies à partir de l'interface Bluetooth[11].

1.9 Les dernières avancées technologiques de l'informatique diffuse

1.9.1 ClassRoom2000

Ce projet fut commencé en juillet 1995 et vise à impliquer l'informatique ubiquitaire dans le système d'éducation afin d'apporter un plus aux pratiques standards de l'enseignement dans le future.

Dans une salle contenant des dispositifs classiques, l'enseignant s'assois à la frontale de la class et écrit dans un tableau (exemple : whiteboard) et souvent efface les derniers points abordés pour en écrire d'autres. Durant le cours une variété de questions peuvent probablement être posées et des réponses doivent être fournisses. Cependant, à la fin de la séance du cours, la structure détaillée, la discussion, et le déroulement de la leçon sont souvent perdus, et ne demeure que les prises de notes des étudiants, les références et parfois les supports du cours de l'enseignant[15].

ClassRoom2000, offre la possibilité de capturer la leçon toute entière y compris son déroulement afin qu'elle servira de référence en elle-même. Cependant, la vidéo toute seule est en limite d'utilisation comme référence, le défi était alors de créer un point d'indexation permettant aux étudiants de passer à travers le block vidéo, pour se positionner sur un point au moment où une réponse à une question donnée peut être fournis[15].

En outre, ces points d'indexation doivent être générés automatiquement d'une manière claire pour tous ceux qui veulent les utiliser. Avec un tableau électronique (Xerox Liveboard a été utilize dans cette optique), il est possible d'estampiller toutes les annotations de l'enseignant pendant le cours, les coulisses de transitions durant la présentation, et d'autre entrées générées par les utilisateurs, et celles qui sont utilisées pour indexer les enregistrements audio-visuel de la leçon. Ainsi, la combinaison media, frise chronologique et indices représente un récapitulatif puissant qui peut être immédiatement disponible et

prêt à l'usage à la terminaison de la séance du cours.

ClassRoom2000 a prouvé son importance à travers sa fréquente demande, il a été répliqué cinq fois à Georgia Tech campus et même déployé dans trois autres universités[15].

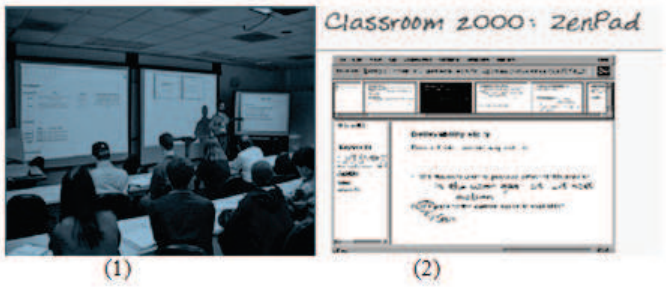


FIGURE 1.2 – (1) ClassRoom 2000 : Gregory Abowd enseigne pendant que (2) ZenPad UI capture les notes et indexe les points de la leçon en temps réel[15].

1.9.2 ContextNotePad

C'est une version sensible au contexte de l'application Notepad (bloc note, mémo, etc.) elle dispose des caractéristiques suivantes [12] :

- **Le dispositif est sensible au mouvement de faible amplitude :** il s'allume automatiquement lorsque l'utilisateur le prend dans la main et inversement, s'éteint lorsqu'il est reposé ;
- **Le dispositif est sensible au déplacement :** par exemple lorsqu'il est tenu par un utilisateur en marche, la taille des caractères est augmentée en sorte que l'utilisateur puisse lire le texte malgré son déplacement. Le système rétablit la taille normale de la police de caractères à l'arrêt du mouvement ;
- **Le dispositif est sensible à la lumière :** si l'éclairage de celle-ci baisse de manière significative, l'écran est rétro-éclairé pour faciliter la lecture ;
- **Le dispositif est sensible à l'utilisation :** s'il n'est pas utilisé, l'écran est masqué jusqu'à détection de présence d'étrangers à proximité en sorte de conserver le caractère privé des informations affichées.

1.9.3 Le système AURA

AURA système (Advanced User Resource Annotation) réalisé par Brush et al. en 2005 permet à une personne de scanner un code à barres des objets, y compris les CD, les DVD et les produits alimentaires emballés, en utilisant un ordinateur de poche sans fil avec un lecteur de code à barres ci-joint. Après le scan, les utilisateurs peuvent visualiser, stocker et partager les métadonnées et les annotations sur l'objet balayé[15].

FIGURE 1.3 – l'utilisation d'AURA système pour le scan des codes à barres dans un magasin de grosserie[15].

1.9.4 DUMMBO Meeting Board

DUMBO (Dynamic Ubiquitous Mobile Meeting BOard) est un tableau physique, blanc utilisé lors d'une réunion. Il est doté de fonctionnalités lui permettant de capturer le contenu et les informations audio-vidéo provenant de la réunion. La capture se lance automatiquement dès que deux personnes au moins se trouvent au voisinage du tableau[12, 16].

1.9.5 Context-Aware Office Assistant

Comme son nom l'indique, c'est un assistant de secrétaire de bureau. Il permet d'annoncer les nouveaux arrivants, les faire patienter en attendant leur tour et de leur proposer d'autres rendez-vous en cas d'arrivée en retard ou d'annulation de leur rendez-vous en cours[12].

1.9.6 Cyberguide

C'est une application développée par Long et al. de Georgia Tech en 1996. Elle est fondée sur la localisation et fournit à un utilisateur véhiculé sa position et le représente sur une carte. Elle produit automatiquement des informations sur les lieux alentours et permet de réserver et de payer la place de parking. Cette place est indiquée au conducteur en tenant compte des places disponibles et de la position courante de l'utilisateur[12, 15].

1.9.7 CyberDesk

C'est une application qui propose d'autres services de bureau comme le mail et le web en fonction du contexte. Par exemple, lorsqu'un utilisateur lit un mail dans lequel se trouve un lien http, l'application ouvre ce lien automatiquement dans un navigateur en préférence de l'utilisateur, et cela se fait en tâche de fond pour ne pas le gêner. Si l'utilisateur change de mail sans consulter le lien ouvert dans le navigateur, l'application le considère comme non intéressant et le ferme[12].

1.9.8 Information Display

C'est une application qui affiche les informations pertinentes sur la localisation de l'utilisateur sur l'écran du dispositif. Ce dernier s'allume automatiquement quand quelqu'un se trouve à proximité afin de lui donner la possibilité de lire les informations[12].

1.9.9 In/Out Board

C'est une application qui se présente sous forme d'un tableau qui indique la présence ou l'absence d'un utilisateur dans son lieu de travail, l'heure à laquelle ce dernier s'est présenté ou à laquelle il est reparti. [12]

1.9.10 Le bracelet GPS pour les malades d'Alzheimer

Le travail a été réalisé par l'opérateur orange en collaboration avec la société canadienne "Medical mobile", aidant à détecter la position d'un patient atteint d'Alzheimer qui se désoriente dès qu'il s'éloigne de son repère familial, et qui se retrouve incapable de

retracer son chemin. Avant d'équiper le malade du bracelet, la première étape consiste à le configurer : le tuteur du patient doit limiter un périmètre diurne et nocturne. Si le patient le dépasse un message d'avertissement automatique sera transféré vers l'opérateur de médical mobile indiquant la position exacte du malade grâce au récepteur GPS et aux antennes reliées. Ainsi le membre de la famille dont le numéro de téléphone est enregistré lors de la configuration sera contacté pour qu'il puisse récupérer son malade et même de lui parler directement car le bracelet est doté d'une carte SIM[9].

1.9.11 L'assistance médicale de SFR pour les diabétiques

Le Service Diabète a été réalisé par l'opérateur français SFR en collaboration avec l'association française des diabétiques. Le principe est d'embarquer dans les téléphones de poche ou PDAs une application permettant d'analyser, de lire et de sauvegarder les taux de glycémie mesurés par le malade via un lecteur connecté au mobile par une liaison Bluetooth[9].

Le médecin aura des relevés transférés automatiquement à son ordinateur. Ainsi, il aura une connaissance détaillée de l'historique des relevés de son patient, et pourra suivre les mesures de la glycémie à jeun ou non et le nombre d'insuline prise. Il a également la possibilité de prévenir et de mettre en garde son patient lors de la détection d'un éventuel danger, via un SMS[9].

1.9.12 L'assistance médicale de SAMU pour les diabétiques

Depuis l'an 2003, l'équipe du SAMU (Service Ambulatoire d'urgences) d'Avignon utilise le service "Mobile Urgence Medical", une solution pratique que France Télécom a mis en oeuvre en partenariat avec la société de distribution de matériel médical GardioGap. Les services du SAMU sont ainsi informés en temps réel de l'état de santé d'un patient embarqué dans une ambulance et de ses conditions de transport pendant son transfert vers les urgences grâce à l'utilisation successive des technologies bluetooth, GPRS, ADSL et VPN[9].

L'ambulance est équipée d'un module de faible encombrement pour le recueil des paramètres vitaux (comme l'électrocardiogramme, la pression artérielle, etc.) et d'une

tablette PC dotée d'une liaison GPRS, les deux sont reliés par une liaison Bluetooth. Au centre de régulation du SAMU, un ordinateur équipé d'une connexion ADSL permet de recevoir les données transmises par l'ordinateur à bord de l'ambulance[9].

1.10 Conclusion

Ce chapitre a mis l'accent sur les notions préliminaires de l'informatique ubiquitaire qui nous servirons de base pour les chapitres à venir.

Le bon fonctionnement des systèmes ubiquitaires est marqué par plusieurs contraintes dont la sensibilité au contexte, qui fait l'objet de second chapitre.

Chapitre

2 Contextes et sensibilité aux contextes

2.1 Introduction

Dans leurs interactions quotidiennes et afin d'atteindre un meilleur niveau de communication, les êtres humains utilisent et prennent en considération leurs contexte d'une manière implicite, ce qui n'a pas été pris en compte dans l'interaction entre humains et systèmes informatiques classiques. L'objectif de l'informatique ubiquitaire est donc d'intégrer cette approche dans les systèmes informatiques diffuse (SID), dans le but de les rendre sensibles au contexte et flexible.

Dans ce chapitre nous passerons en revu deux parties essentielles : le contexte et la sensibilité au contexte. La première définit toutes les notions relatives au contexte (définitions, caractéristiques, acquisition, catégorisation et modélisation), la seconde est dédiée à l'étude de la sensibilité des applications aux variations du contexte (définitions, l'architecture générique et les quatre plateformes principales des applications sensibles au contexte) et nous terminerons par une conclusion.

2.2 Le contexte

2.2.1 Définitions

La notion de contexte a été utilisée initialement dans le domaine de la linguistique, puis a été élargie à d'autres domaines comme la psychologie, la philosophie, l'informatique, etc. Vu que ce mot couvre une large gamme de domaines, une meilleure compréhension de ce terme est donc requise.

1. **Définition linguistique**

   Plusieurs définitions du terme contexte ont été retrouvées dans divers dictionnaires littéraires, à savoir [18, 19] :

   – **L'encyclopédie Larousse :** définit le mot contexte comme étant "l'ensemble des conditions naturelles, sociales, culturelles dans lesquelles se situe un énoncé ou un discours. Il est aussi l'ensemble des circonstances dans lesquelles se produit un événement où se situe une action" ;

   – **Grand Dictionnaire d'Office québécois de la langue française** [20] **:** "Énoncé dans lequel figure le terme étudié" ou encore "ensemble d'un texte précédant ou suivant un mot, une phrase, un passage qui éclaire particulièrement la pensée d'un auteur". Et, si l'on parle d'informatique : "Ensemble d'informations concernant l'action du stylet, en rapport principalement avec sa localisation à l'écran, qui permet au système d'exploitation de l'ordinateur à stylet de différencier les commandes et l'entrée des données, et de fonctionner en conséquence".

2. **Définition Informatique**

   La notion du contexte en informatique est ancienne et remonte aux années soixante où elle a été déjà exploitée par différents domaines, tels que les systèmes d'exploitation, la théorie des langages et l'intelligence artificielle. Dans les systèmes d'exploitation par exemple, un contexte caractérise l'ensemble minimal d'informations sur une tâche en exécution (processus) et permet de retourner à l'exécution de cette tâche après l'occurrence d'une interruption et l'exécution du programme de traitement de cette dernière [19, 21].

   Différents chercheurs se sont intéressés à éclaircir la notion de contexte pour l'informatique ubiquitaire. En effet, il est difficile de trouver d'une manière claire et unique, une définition valable dans tous les travaux impliquant cette notion. On cite ces définitions selon leurs ordres chronologiques comme suit [19,21] :

– En 1994, plusieurs approches au mot contexte ont été introduites par divers chercheurs : [19,21]

  ▪ Schilit et Theimer, [22] introduisent l'expression context aware pour désigner la localisation et l'identité des personnes et des objets à proximité ainsi que les modifications pouvant intervenir sur ces objets.

  ▪ Schilit et all [23], ont donné une définition plus vaste ; selon eux le contexte englobe plus que la localisation de l'utilisateur puisque d'autres centres d'intérêt peuvent aussi changer du contexte. Tels que les effets de la lumière, le niveau du bruit, la connectivité réseau, le coût de la communication et la bande passante de la communication.

– En 1997, une multitude de définitions ont été présentées :

  ▪ Brown, Bovey et Chen [24] présentent des éléments de base flexibles : la localisation, l'ensemble des objets dont l'utilisateur a besoin, le temps et l'orientation spatiale (direction) afin de caractériser le contexte.

  ▪ Ryan, Pascoe et Morse [25], annoncent que : "les éléments du contexte sont : la localisation de l'utilisateur, l'environnement, l'identité et le temps".

– En 2000, Chen et Kotz [26], déterminent le contexte comme un : "ensemble des états environnementaux et paramètres qui déterminent le comportement d'une application ou dans lequel un événement de l'application se déroule et ayant un intérêt pour l'utilisateur".

– En 2001, Dey [27], affirme qu'un contexte désigne toute information utile pour caractériser la situation d'une entité qui peut être un objet ou une personne.

Le mot contexte se définit selon deux axes, le premier se base sur l'énumération du contexte et le deuxième tente de formaliser les termes du contexte. L'importance du contexte dans le domaine de l'interaction homme-machine et les systèmes mobiles a généré des définitions centrées sur l'utilisateur ou sur les applications[19].

En 2004, Brezillon et all [29] ont conclu dans leur analyse que la plupart des définitions sont une suite d'enchainement de réponses aux questions : **qui ?** (entité qui peut être une personne ou un objet physique ou informatique.), **quoi ?** (percevoir et interpréter l'activité de l'utilisateur), **quand ?** (indexation temporelle

d'un évènement), **où ?** (localisation de l'utilisateur ou d'un évènement du système), pourquoi ? (consiste à savoir la raison du déclenchement de l'évènement ou de l'activité)et **comment ?** (la façon dont l'activité se déroule).

Dans le cadre de notre projet, la définition qui est plus appropriée est celle qui considère le contexte comme étant : **" Toute information dont le changement de sa valeur engendre un service ou un changement de comportement de l'application "** c'est ce qu'on appelle un contexte pertinent.

2.2.2 Acquisition du contexte

Les informations sont acquises indépendamment de l'application selon plusieurs méthodes. Mostefaoui [36] les a classées en trois catégories :

- **acquisition par profil**[1] **:** cette méthode consiste à récupérer des informations soit à travers une interface graphique soit par l'intermédiaire d'un fichier de profil ;
- **acquisition par sonde :** cette méthode consiste à utiliser des sondes (capteurs) pour récupérer les informations de contexte ;
- **acquisition par dérivation :** la dérivation ou l'interprétation du contexte consiste à utiliser un ou plusieurs observables (informations déjà acquises) pour déduire ou calculer en temps réel un contexte de plus haut niveau en utilisant des méthodes d'interprétation.

2.2.3 Les caractéristiques du contexte

Les informations contextuelles ont des caractéristiques variables de fait qu'elles sont collectées à partir de plusieurs sources hétérogènes. Un contexte peut être statique : informations non influençables au cours du temps, collectées au lancement de l'application (profil de l'utilisateur, taille de l'écran, etc.) ; ou dynamique : informations dont leurs valeurs changent continuellement, une observation de leur état peut devenir très rapidement obsolète. Il est très important d'avoir une description abstraite de ces informations afin de pouvoir les utiliser et les catégoriser [37, 30].

1. Un profil est un ensemble d'information personnel (Nom, prénom, âge, fonction, loisir, etc.).

2.2.4 Catégories du contexte

Les informations contextuelles sont généralement classées sous quatre catégories primaires et une catégorie secondaire, illustré dans le tableau ci-dessous [28, 37] :

Type de catégories	Catégories	Explication
primaire	L'identité	Se réfère a la capacité d'assigné un unique identifiant à une entité.
	La localisation	Concerne la position des entités (le lieu de leur présence) ainsi que l'orientation, l'altitude et les relations spatiales entre ces derniers.
	L'état (activités)	Encapsule les caractéristiques intrinsèques des entités intervenant dans le système.
	Le temps	Permet d'établir l'historique du contexte.
Secondaire		**Indique le profil de l'utilisateur y compris ses préférences.**

TABLE 2.1 – Catégories des informations contextuelles.

2.2.5 Modélisation des informations contextuelles

La première tâche de la conception des applications informatiques diffuses consiste à comprendre le contexte, qui doit être modélisé sous une forme appropriée qui, favorise son partage entre les équipements et fournit un niveau élevé d'abstraction pour faciliter la tâche d'adaptation.

Un certain nombre d'approches de modélisation du contexte ont été développées au cours de la dernière décennie, allant de très simples premiers modèles à l'état actuel de l'art des modèles contextuels et les plus intéressantes sont celles orientées ontologie parce qu'elles permettent le partage de contexte et un raisonnement efficace [18 , 19].

Nous dressons ci-dessous un panorama des approches les plus connues de modélisation du contexte.

2.2.5.1 Approches non ontologiques

1. **Modèle attribut/valeur**

   L'approche par associations **attribut/valeur** ou encore **clé/valeur** a été proposée pour la première fois pas Schilit, Adams et Want en 1994 [38]. Elle a pour objectif de modéliser les attributs contextuels par une structure de données simple de type associations attribut/valeur. L'attribut est une métadonnée qui représente le nom/type de la donnée contextuelle à laquelle on associe une variable d'environnement (valeur) [18, 19].

   Cette approche est très utilisée dans les services distribués où les services sont décrits en général avec une liste d'attributs sous forme d'attribut/valeur et la découverte de services est ensuite appliquée par des algorithmes qui utilisent ces paires attribut/valeur (Held,Buchholz et Schill, 2002)[39]. Les informations peuvent être sauvegardées dans une base de données sous forme de tables dont les colonnes correspondent à une information représentée par une même métadonnée. Chaque ligne (tuple) représente une suite d'informations pour caractériser les différents documents suivant le terminal de l'utilisateur (Kammanahalli, 2004) [40].

2. **Modèle de présentation par balises**

   Cette représentation est formée d'une structure de données hiérarchique constituée de balises avec des attributs et des contenus. Elle utilise des langages dérivés du SGML (Standard Generic Markup Language) en particulier, le XML (eXtended Markup Language). Les entités du contexte sont exprimées sous forme de pages web qui peuvent être extraites par un URL (adresse web). Les profils qui décrivent les données et les ensembles de données constituent des exemples typiques de cette approche [19].

3. **Modèle graphique**

   Cette approche consiste à modéliser les informations contextuelles selon un graphe conceptuel pour représenter les attributs contextuels et leurs relations. Indulska [42] propose un modèle graphique utilisé pour représenter un attribut contextuel (dispositif mobile, utilisateur, localisation, etc.). Les relations entre objets, sont représentées graphiquement par des arcs (type de relation entre les attributs contex-

tuels). Le schéma graphique de représentation du contexte est généralement défini lors de la phase de conception de l'application. La traduction de ce schéma, en une base de données relationnelle, vise ensuite à faciliter la gestion des attributs contextuels à partir d'une requête SQL. L'extension de ce modèle nécessite cependant de refondre l'application et la base de données du contexte.

4. **Modèle orienté objets**

   Cette approche tire profit des mécanismes offerts par les langages orientés objets (encapsulation, réutilisation, héritage, etc.). Les informations contextuelles sont groupées dans un ensemble d'entités (personne, réseau, dispositif, etc.) et les propriétés de ses entités sont représentées par des attributs. Chaque entité est reliée à son attribut par une relation d'association. Prenons comme exemple le langage ContextUML, basé sur UML (Unified Modeling Language) qui permet de décrire des relations entre les informations de contexte [8, 19, 30].

5. **Modèle logique**

   Dans un modèle logique, les informations contextuelles doivent être représentées d'une façon formelle comme des faits, il utilise l'algèbre booléenne [2] ou la logique du premier ordre [3] afin de définir un processus de raisonnement ou d'inférence offrant la possibilité d'abstraire les données contextuelles collectées [7,30].

2.2.5.2 Approche ontologique

Une **ontologie** est une description structurée de concepts. Ces derniers sont organisés dans un graphe dont les relations peuvent être de type sémantique, composition ou d'héritage. Son objectif est de modéliser un ensemble de connaissance dans un domaine donné (description de domaine sous une forme utilisable par la machine). Les ontologies ont comme grand avantage le partage de ressources, et elles ont généré un grand intérêt dans l'intelligence artificielle [28].

2. L'Algèbre booléenne ou calcul booléen, est la partie des mathématiques, de la logique et de l'électronique qui s'intéresse aux opérations et aux fonctions sur les variables logiques.

3. La logique mathématique, logique formelle est une discipline des mathématiques introduite à la fin du XIXe siècle, qui s'est donnée comme objet l'étude des mathématiques en tant que langage.

L'utilisation des ontologies a été mise en valeurs pour ses caractéristiques suivantes [7,28] :

- Une ontologie permet de partager les informations contextuelles dans un système distribué ;
- Une ontologie comprend des sémantiques déclaratives permettant d'élaborer des raisonnements sur les informations contextuelles ;
- Une ontologie assure l'interopérabilité des applications et des terminaux.

2.2.6 Synthèse sur les modèles

Les approches orientées objet utilisent un modèle formel pour décrire le contexte. Ce type d'approche permet de décrire le contexte de manière plus riche que les approches attribut/valeur. De plus, cela offre aux concepteurs d'applications la possibilité de réutiliser le modèle pour d'autres applications.

Les modélisations basées sur un langage de balises permettent de décrire des profils ou des contextes simples, sans offrir la possibilité de décrire des relations de dérivation et de dépendance entre ces informations.

Les modélisations existantes basées sur UML permettent de décrire des relations entre les informations de contexte mais ne prennent pas en compte la description des dépendances entre ces informations, ni la qualité ou la validité temporelle des données décrites. CML est venu remédier à certains de ces manques en proposant un modèle avec visualisation graphique qui permet de typer le contexte, et de décrire un ensemble de relations entre plusieurs contextes observables.

L'approche basée sur la logique ne permet pas de décrire la validité temporelle des informations ni les relations qui peuvent exister entre les informations de contexte mais peut être utilisée dans l'informatique sensible au contexte afin d'intégrer et d'interpréter les données collectées.

Les approches orientées ontologie sont des approches formelles qui tirent parti des caractéristiques des ontologies pour modéliser le contexte [19, 30].

2.3 La sensibilité au contexte

2.3.1 Définitions

La notion de sensibilité au contexte "context-awareness" concerne l'utilisation du contexte dans les applications. Elle caractérise la capacité d'un système à s'adapter aux changements du contexte [19, 28].

Plusieurs définitions ont été présentées par plusieurs chercheurs :

- Selon Schilit, Adams et Want [32], premiers chercheurs à avoir évoquer le terme sensibilité au contexte en 1994 dans leurs travaux sur un système de localisation, la sensibilité au contexte est définie comme l'aptitude d'une application à s'adapter au contexte de son exécution selon : la localisation, l'ensemble des personnes à proximité, les machines, les équipements accessibles, de même que les changements de ces objets dans le temps.
- En1995, Brown [33] a défini la sensibilité au contexte dans son travail relatif à un guide touristique comme toute application qui prend en compte le contexte de l'utilisateur.
- En couplant et en élargissant les définitions de Brow et Dey(2001) [34], on tire aussi la définition suivante : "les applications sensibles au contexte sont des applications dont la structure et le comportement varient en fonction du contexte. Elles utilisent les observations de contexte pour fournir des informations et des services pertinents pour l'utilisateur".

2.3.2 L'architecture d'un système sensible au contexte

Dey [43] a été le premier à proposer une architecture pour un système sensible au contexte pour atténuer les difficultés de la manipulation du contexte. Son architecture se présente sous Cinq couches (figure 2.1).

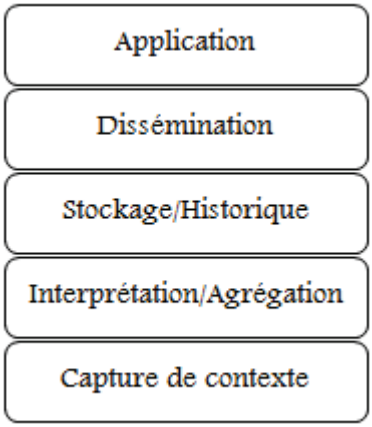


FIGURE 2.1 – Architecture général d'un système sensible au contexte [43].

2.3.2.1 La couche capture de contexte

Elle vise à détecter les informations contextuelles à l'aide des capteurs matériels ou logiciels. Il existe trois catégories de capteurs [28, 41].

- **Les capteurs physiques :** ce sont des dispositifs matériels susceptibles de fournir des données contextuelles. Citons comme exemples, un GPS qui renvoie la localisation et un thermomètre numérique qui pourvoit la température.
- **Les capteurs virtuels :** ce sont des capteurs qui offrent des informations contextuelles en se basant sur d'autres applications ou services logiciels. Par exemple il est possible de détecter l'emplacement d'un livreur de marchandise en consultant son carnet électronique [4] de rendez-vous sans avoir recours à des capteurs physiques.
- **Les capteurs logiques :** ce sont des capteurs qui utilisent plusieurs sources d'informations contextuelles pour fournir une autre information de synthèse plus précise, ils peuvent réutiliser les capteurs physiques et virtuels afin de fournir une information contextuelle de plus haut niveau. Par exemple, un capteur logique peut fournir l'emplacement d'un caissier dans un grand magasin en analysant les différentes sessions ouvertes sur les caisses du magasin.

4. Un carnet électronique est un logiciel ou une partie d'un logiciel contenant une fonctionnalité pour gérer son emploi du temps.

2.3.2.2 La couche Interprétation et Agrégation de contexte

Elle sert à analyser et à transformer les données contextuelles brutes en paramètres de haut niveau plus faciles à utiliser. En effet les capteurs fournissent généralement des données techniques qui ne sont pas appropriées pour une utilisation directe par l'application. La transformation peut s'effectuer par plusieurs opérations : extraction, raisonnement, etc.

2.3.2.3 La couche Stockage et Historique du contexte

Le principe de cette couche consiste à organiser les données déjà capturées et interprétées puis les stocker pour une utilisation ultérieure (gestion de l'historique).

2.3.2.4 La couche dissémination du contexte

Cette couche assure la transmission des différentes informations contextuelles à l'application et une transparence totale de la communication avec cette dernière. L'implantation de ces types de communication est nécessaire pour garantir la sensibilité au contexte dans une application.

2.3.2.5 La couche application du contexte

Cette couche est représentée par l'application qui offre ses services aux différents utilisateurs. Elle implémente les réactions nécessaires aux changements du contexte.

2.3.3 Principales plateformes des systèmes sensibles aux contextes

Au cours du temps, plusieurs plateformes ont été élaborées par divers chercheurs. Nous présentons ci-après les quatre principales implémentations de plateformes sensibles au contexte qui servent de références à l'implémentation des principales couches d'un système sensible au contexte.

2.3.3.1 Le context Toolkit

Cette plateforme a été proposée en 2001 par Dey, Abowd et Salber [17] pour appuyer le développement des systèmes sensibles au contexte, elle recouvre les différentes couches

du modèle générique présenté dans la (figure 2.1). C'est une infrastructure de type flux de données, fondée sur cinq composants logiciels et deux composants matériels :

- **les capteurs :** capture de contexte physique ;
- **les gadgets (contexte Widgets) :** est un composant autonome qui encapsule un capteur physique. Son rôle est de communiquer les informations perçues à un (ou plusieurs) serveur(s) ou interpréteur(s) en fonction des données reçues par le capteur. Il a aussi pour rôle de constituer un historique qui lui est propre, ce qui représente la couche capture de contexte dans le modèle générique ;
- **les interpréteurs :** transforment l'information du contexte dans le but d'élever le niveau d'abstraction, ils représentent la couche interprétation du contexte dans le modèle générique ;
- **les groupeurs (serveurs) :** font le lien entre les context-widgets et les applications sensibles au contexte et regroupent des informations du contexte relatif à un sujet ou à une situation ;
- **le découvreur :** c'est la liaison entre les serveurs, les widgets et les applications. Il maintient un registre des capacités existantes dans le cadre de travail (les composantes actuellement disponibles pour utilisation par des applications cherchant un tel composant et /ou service) ;
- **le service :** exécute des actions au profit des applications ;
- **Les actionneurs :** exécutent les tâches transmises par le service.

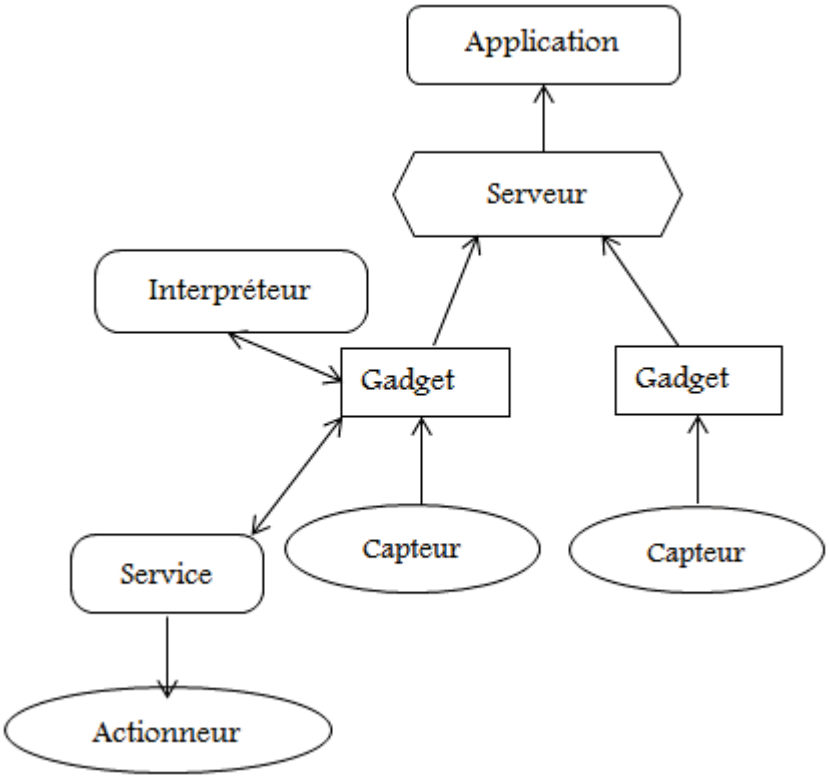


FIGURE 2.2 – Éléments de l'architecture du Contexte Toolkit. [19]

2.3.3.2 Le contexte Broker Architecture (CoBrA)

C'est une architecture orientée agents pour les systèmes sensibles au contexte, développée en 2004 par Chen et al [44] (Figure 2.3). L'élément central de cette architecture est un agent intelligent appelé le courtier de contexte (context broker) qui maintient une base de connaissances partagée pour une communauté d'agents, de services et de capteurs.

L'agent courtier possède une architecture en couche comprenant les éléments suivants :

- base de connaissance du contexte ;
- moteur de raisonnement sur le contexte ;
- module d'acquisition du contexte ;
- module de gestion de confidentialité.

Le courtier de contexte a les responsabilités suivantes [28] :

- Fournir un modèle centralisé de contexte qui peut être partagé par tous les dispositifs, services et agents dans l'environnement ;

– Acquérir les informations contextuelles provenant de sources inaccessibles par des terminaux à faible capacité ;
– Assurer l'interprétation de l'information contextuelle qui ne peut pas être directement acquise des capteurs (par exemple, des intentions, des rôles, des relations temporelles et spatiales, etc.) ;
– Détecter et résoudre les conflits d'interprétation du contexte stocké dans le modèle partagé ;
– Protéger la confidentialité des utilisateurs en appliquant leurs politiques de sécurité pour le partage et la communication de données (authentification par un mot de passe).

L'agent CoBrA repose sur un modèle de contexte appelé Soupa (Standard Ontology for Ubiquitous and Pervasive Applications). Ce dernier est composé d'un ensemble d'ontologies écrites en langage OWL (Ontologie Web Language), qui permettent à des applications hétérogènes de partager la même interprétation du contexte à travers un vocabulaire commun[19, 21, 45].

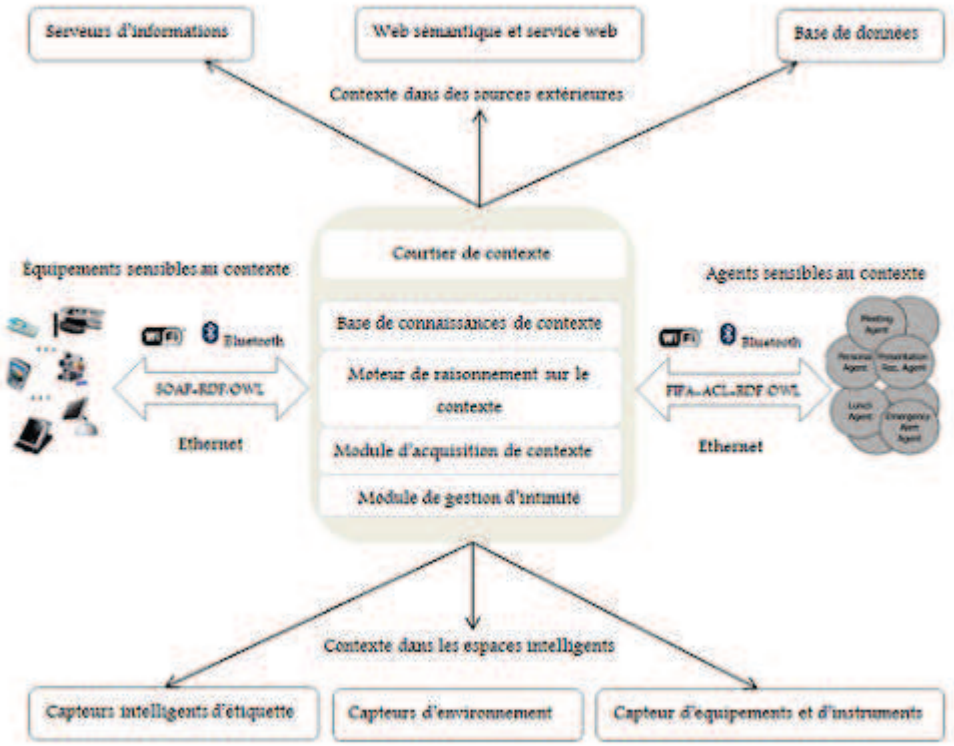


FIGURE 2.3 – Infrastructure d'un espace actif basé sur l'agent CoBrA[35].

2.3.3.3 Contexte Managment Framework (CMF)

Le CMF a été développé par Korpipää et al [46]en 2003, c'est une plateforme analogue au contexte toolkit de Dey qui contient cinq composants de base comme suit :

- **Ressource server (Le Serveur de ressources) :** acquière des informations de contexte des capteurs physiques et les interprète avant de les envoyer au gestionnaire de contexte ;
- **Context manager (Le Gestionnaire du contexte) :** permet de stocker des informations contextuelles dans un serveur et de les livrer aux clients selon différents mécanismes (requête/réponse, inscription/notification etc.) ;
- **Context recognition service (Le Service de reconnaissance du contexte) :** convertie le flux de données reçus en une représentation définie dans l'ontologie du contexte ;
- **Change detection service (Le Service de détection des changements) :** détecter les changements d'un service suite à un changement de contexte ;
- **Security (La Sécurité) :** vérifie et contrôle les informations de contexte. Ces entités communiquent entre elles suivant le modèle client/serveur.

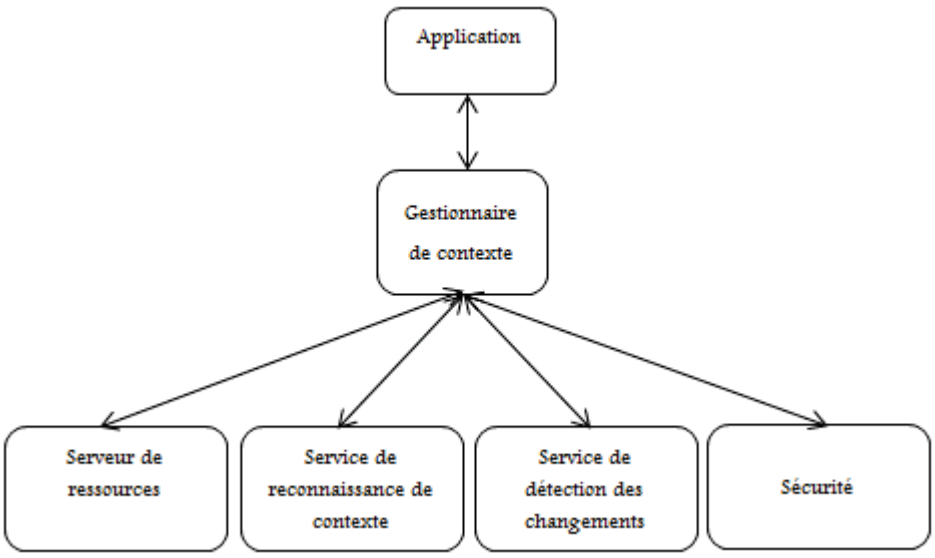


FIGURE 2.4 – L'architecture générale du Context Management Framework (CMF). [28]

2.3.3.4 Service Oriented Contexte-aware Middleware (SOCAM)

L'architecture de l'intergiciel SOCAM est constituée d'un ensemble de composants qui lui permettent de gérer automatiquement la collecte et l'interprétation du contexte [19, 28, 30] :

- **Context provider (fournisseur de contexte) :** se charge de collecter le contexte puis de le convertir en des représentations d'OWL pour que le contexte puisse être partagé. Ensuite, il le fournit au Context interpreter. Deux types de Context provider existent :
  - **Context providers externes :** les fournisseurs externes du contexte se chargent de collecter le contexte à partir des sources de données externes à la machine comme la température de la pièce par exemple ;
  - **Context providers internes :** les fournisseurs internes du contexte se chargent de collecter des données sur l'état interne de la machine comme la valeur de la bande passante par exemple.
- **Context interpreter (interprète de contexte) :** fournit un service de raisonnement logique sur les présentations OWL du contexte en appliquant une suite de règles d'interprétations ;
- **Service Locating Service (Service de localisation de services) :** il est utilisé par le Context Interpreter et le Context Provider pour s'enregistrer et par les services sensibles au contexte pour trouver des services d'interprétation du contexte et pour collecter des données ;
- **Contexte DataBase (base de données de contexte) :** elle stocke les différents éléments décrivant l'environnement de l'application ;
- **Contexte Aware service (services sensibles au contexte) :** il utilise les différentes informations sauvegardées dans la base de données de contexte pour modifier son comportement selon le contexte courant.

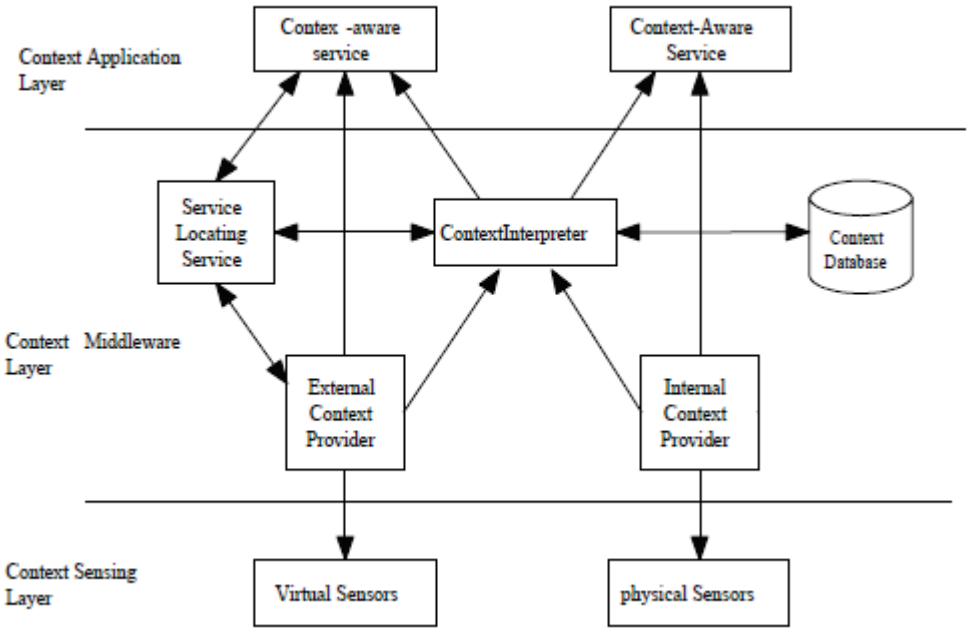


FIGURE 2.5 – Plateforme SOCAM [30].

2.4 Conclusion

Ce chapitre s'est focalisé en premier lieu sur deux éléments principaux : le contexte et la sensibilité au contexte. Son but est d'affiner la définition de ses deux notions afin de pouvoir les utiliser d'une manière simple ultérieurement et en second lieu sur les différentes approches de modélisation qui nous servirons de référence pour le choix du modèle et de langage de conception de notre application dans le but de faciliter la tâche de programmation des applications sensibles au contexte.

Dans la suite de notre travail, nous envisageons d'élaborer un chapitre intitulé "les applications mobiles" afin d'enrichir nos connaissances et de pouvoir utiliser l'ensemble de ces acquis pour le développement de notre application.

Chapitre

3 Appareils et applications mobiles

3.1 Introduction

Actuellement, les périphériques mobiles sont les appareils les plus utilisés pour communiquer ou surfer sur Internet par rapport à un ordinateur personnel. L'augmentation importante de l'utilisation de ces périphériques mobiles est suivie par le développement d'applications mobiles dédiées à ce type d'appareils. Ces dernières sont donc un abandon progressif des systèmes logiciels, que l'on trouve généralement sur des ordinateurs.

Avant qu'on se lance dans le développement d'une application mobile, il est nécessaire de définir un périphérique mobile pour pouvoir introduire par la suite ce qui est une application mobile.

Ainsi, ce chapitre sera divisé en deux grandes parties. La première constituera une revue des notions de base relatives aux périphériques mobiles, tandis que la seconde sera dédiée aux applications mobiles.

3.2 Les appareils mobiles

3.2.1 Définition

Un appareil mobile, traduction littérale du terme anglophone "mobile device", est un ordinateur portatif utilisable de manière autonome lors d'un déplacement. L'appareil est de taille et de poids réduits pour permettre un usage mobile. Il est doté d'une batterie et bénéficie souvent d'une connexion Bluetooth ou même wifi pour fonctionner de manière autonome. Il se présente typiquement comme un écran possédant une interface tactile ou un clavier miniaturisé. Il peut s'agir de tablettes tactiles, de Smartphones, de téléphones

mobiles, d'assistants personnels (PDA), d'assistants de navigation personnel, etc. [48].

3.2.2 Caractéristiques

Selon leurs utilisations et leurs importance, on distingue deux familles d'appareils mobiles sur lesquels une application mobile peut s'exécuter [49] :

- **Les PDAs :** Un PDA récent (défini dans la section I.7.1.2 du chapitre I) dispose, en moyenne d'un processeur de 500 MHz, de 64 MO de RAM et d'une carte-mémoire. Côté connectivité, Bluetooth et Wi-Fi sont de rigueur.
- **Les téléphones mobiles :** Leurs performances sont plus modestes, même si la tendance actuelle des constructeurs est de proposer des appareils tout-en-un de plus en plus puissants disposant des mêmes fonctionnalités qu'un PDA, à l'image du Nokia 9500.

FIGURE 3.1 – Le téléphone Nokia 9500. [50]

3.2.3 Limites

Le développement d'applications pour périphériques mobiles se confronte à de différentes limites relatives au périphérique lui-même. On peut citer : la taille d'écran, la saisie des données et la stabilité du réseau[49].

3.2.3.1 La taille d'écran

Elle est réduite et varie d'un appareil à un autre. Ce qui limite largement le nombre d'informations que l'on peut afficher[49].

3.2.3.2 La saisie des données

La saisie des données dans un périphérique mobile semble être relativement pénible. Dans le cas des téléphones mobiles, il est impératif d'évaluer la difficulté rencontrée lors d'une simple opération de saisie (exemple : la saisie d'un SMS) sans citer même des cas plus complexes comme le remplissage d'un formulaire. Certains PDAs disposent de mini claviers ou de logiciels de reconnaissance de caractères, n'empêche que la saisie reste lente et fastidieuse[49].

3.2.3.3 L'instabilité du réseau

Les périphériques étant dotés d'une connexion Wi-Fi, les utilisateurs sont confrontés à de fréquentes déconnexions et risquent,par conséquent, de ne pas être toujours connectés[49].

3.2.4 Quelques exemples de périphériques mobiles

Nous exposons comme exemple de téléphones mobiles, le Smartphone Samsung Galaxy Grand I9082 qui possède les caractéristiques suivantes : une résolution d'écran de 480x800 pixels, une mémoire de 8GB, une connectivité Bluetooth et wifi, un processeur Dualcore de 1,2 GHZ et qui utilise Androïde comme système d'exploitation[51].

FIGURE 3.2 – Exemple d'un téléphone portable (Samsung Galaxy Grand I9082)[51].

Nous montrons comme exemple de tablette PC sans support GSM, la HP TouchPad qui possède les caractéristiques suivantes : une résolution d'écran de 768x1024 pixels, une mémoire de 16/32GB, une connectivité Bluetooth et wifi, un processeur Dualcore de 1,2 GHZ et qui utilise HP webOS comme système d'exploitation[52].

FIGURE 3.3 – Exemple d'une tablette PC (HP TouchPad)[52].

Nous présentons comme exemple de PDA, le i-mate PDA2 qui possède les caractéristiques suivantes : une résolution d'écran de 240x320 pixels, une mémoire de 128MB, une connectivité Bluetooth et wifi, un processeur intel de 520 MHZ et qui utilise Microsoft Windows mobile comme système d'exploitation[57].

FIGURE 3.4 – Exemple d'un PDA (i-mate PDA2) [57].

3.3 Les applications mobiles

3.3.1 Définition

Une application mobile est un type de logiciel d'application téléchargeable, conçu pour fonctionner sur un appareil mobile, tel qu'un Smartphone ou une tablette Pc. Elle vise à fournir des services similaires à ceux accessibles sur ordinateurs et sont généralement de petites unités de logiciels offrant une fonctionnalité limitée comme un jeu, une calculatrice ou la navigation Web mobile[53, 54].

Plusieurs appellations peuvent identifier une application mobile : application Web, application en ligne ou application Smartphone.

3.3.2 Le mode de fonctionnement des applications mobiles

Une application mobile fonctionne en trois modes : le mode connecté, le mode non connecté et le mode hybride.

3.3.2.1 Le mode connecté

C'est un mode qui impose la disponibilité permanente du réseau. Il est généralement utilisé lorsque la fraîcheur des informations est essentielle ou lorsqu'un utilisateur ne peut se satisfaire de simples synchronisations[49].

Ce mode repose principalement sur deux serveurs. Un Serveur Web de type Apache/PHP et un serveur de base de données (par exemple : MySQL).

1. **Les avantages du mode connecté**
   - Le développement et l'administration sont très simples puisque l'application se situe sur un serveur central ;
   - Nécessite de faibles ressources matérielles.
   - Possibilité de fournir des informations en temps réels ;
   - Facilité de mise à jour du contenu des applications.
2. **Les inconvénients du mode connecté**
   - La connexion réseau est obligatoire ;
   - Interruption de service suite à une défaillance du réseau ;

- Le trafic réseau engendré est important ;
- Vulnérabilité du système, vue que les informations ne résident pas directement sur le client ;
- Echec de l'envoi de données volumineuses (grande photos, vidéo) ;
- Plein de petits chargements qui peuvent être gênants au cours de l'utilisation.

3.3.2.2 Le mode déconnecté

C'est un mode avec des synchronisations périodiques, il s'utilise quant à lui lorsque l'application doit être disponible en tout temps, et que la fraîcheur des informations n'est pas essentielle (agenda, calculatrice, etc.)[49].

1. **Les avantages du mode déconnecté**
   - L'application est toujours disponible, quel que soit l'état du réseau ;
   - L'interface utilisateur est plus conviviale ;
   - L'application est toujours disponible ou que l'on soit ;
   - L'accès aux données est très rapide, d'autant plus que les téléphones disposent d'une mémoire de type SSD, avec des débits de lecture assez élevés ;
   - Lors du téléchargement de l'application, tout son contenu est immédiatement disponible.
2. **Les inconvénients du mode déconnecté**
   - La multiplication des types d'appareils induit souvent des incompatibilités ;
   - Les ressources consommées sont importantes.

3.3.2.3 Le mode hybride

C'est un mode issu de la combinaison des deux modes (connecté et déconnecté)[49].

1. **Les avantages du mode hybride**
   - Un bon mélange entre la disponibilité et la flexibilité du contenu ;
   - Application légère à télécharger ;
   - Seules les nouvelles informations sont téléchargées (ou celles mises à jours évidemment), les autres sont stockées dans le téléphone.

2. **Les inconvénients du mode hybride**
   - Les données peuvent ne plus être valables si l'on oublie de faire la mise à jour ;
   - Le premier lancement implique le téléchargement de tout le contenu. Même en Wifi, cela peut prendre du temps dans certain cas.

3.3.3 Les types d'applications mobiles

Selon leurs modes de fonctionnement, les applications mobiles se réparties en trois catégories : les applications web (connectées), les applications natives (non connectée) et les applications hybrides.

3.3.3.1 Les applications web (connectées)

1. **Présentation**

   Une application web fonctionne en mode connecté. Elle se différencie d'une application native par son point d'accès : elle peut être uniquement lancée depuis un navigateur. Cette solution est également basée sur des langages web (HTML, XML, etc.). On peut alors les qualifier de site web pour appareil mobile. Á chaque opération effectuée ou accès à des données, les informations sont lues sur Internet. Elles sont codées en Objective C sur iPhone, JAVA pour Android et Blackberry.

   Une application mobile web est accessible par tous les Smartphones quelques soient leur marque et système d'exploitation[61, 62, 65].

2. **Fonctionnement**

   Le principe du fonctionnement des applications web est la lecture de données. En effet, une fois lancée, l'application se connecte au serveur web afin de télécharger un fichier, en général au format XML.

   Les flux de ce fichier sont générés en temps réel par un script écrit en PHP, qui interprète la demande reçue et interroge la base de données en question. Dès que les données sont prêtes l'application les prend en charge, les interprète et les place dans l'interface.

   Grace à cette technique de lecture de données, l'ensemble des applications (iPhone, Android, etc.) se connectent à la même source de données ce qui simplifie

aussi la maintenance et les mises à jour.

Ce type d'application est recommandé pour des personnes qui utilisent des services connectés à un site web, tels que l'utilisation des réseaux sociaux, où généralement quand l'accès aux informations doit être rapide (exemple : Dépanne Moi)[61].

3.3.3.2 Les applications natives (non connectées)

1. **Présentation**

Ce sont des applications où toutes leurs fonctions et données médias se trouvent dans le téléphone ou la tablette. Les deux éléments essentiels pour bénéficier des fonctionnalités d'une application non connectée consistent à avoir un téléphone avec l'ensemble des logiciels embarqués. Ce type d'applications est basé sur un langage propre à la plateforme ("langage natif") : codées en Objective C sur iPhone, JAVA pour Android et Blackberry[61,62].

2. **Fonctionnement**

Le téléphone mobile lit tous les fichiers qu'il lui faut directement dans la mémoire de stockage du téléphone, ou dans une base de données embarquée.

Ce type d'application est destiné à remplir une fonction bien précise,sans pour autant avoir besoins d'évoluer dans le temps par leur contenu[61].

3.3.3.3 Les applications hybrides

1. **Présentation**

Une application hybride est une application qui combine des éléments HTML5 sous forme d'application web mobile et des éléments d'une application native. Elle permet d'utiliser les fonctionnalités natives des smartphones et d'être distribuée en tant qu'application sur les plateformes d'applications (App Store, Android Market, etc.).

Le principe d'une application hybride est de réduire les coûts et les délais de développement nécessaires par rapport au développement d'application native spécifique à une seule système d'exploitation.

Elles reposent à la fois sur :

- Un serveur web/php alimenté par une base SQL ;
- Un système de stockage Interne du téléphone ;
- L'application en elle-même.

L'application en utilisation simple vient lire les données sur son disque de stockage, ou sa base de données interne, mais dispose d'une fonction qui permet de synchroniser son contenu[61, 62, 65].

2. **Fonctionnement**

Une fois le téléphone est allumé, l'application vérifie tout d'abord sa connectivité puis envoie une requête de date de dernière mise à jour vers le serveur. Le serveur répond via un flux XML contenant une liste de données à récupérer, (URL(s) de fichiers médias, requêtes SQL pour synchroniser avec la base de données embarquée, etc.). L'application télécharge toutes les informations de manière séquentielles (via HTTP ou FTP), ou exécute les requêtes SQL. Une fois toute la séquence terminée, l'interface est rechargée.

Ce type d'application se prête bien à l'usage professionnel, quand la fiabilité est importante, tout en gardant la possibilité d'échanger des informations rapidement[61].

3.3.4 Exemples d'applications installées sur les appareils mobiles

3.3.4.1 Dépanne-moi

"Dépanne-moi" est un nouveau service mobile créé en fin 2010. C'est une révolution pour chercher le service ouvert le plus proche de l'endroit où on se trouve.

Comme son nom l'indique, "Dépanne-moi" mis à la disposition de l'utilisateur une multitude de services de proximité (commerces, artisans, dépanneurs,etc.) ouverts et les plus proches de l'endroit où ce dernier se trouve. C'est l'annuaire mobile nouvelle génération. En quelques clics, on peut appeller directement le professionnel dont on a besoin et ce depuis l'application [63].

3.3.4.2 Firefox mobile

Firefox Mobile est le nom de la version pour périphériques mobiles de Mozilla Firefox adaptée aux caractéristiques des plateformes mobiles(petit écran, utilisation du clavier malaisée, puissance de calcul faible) [64].

3.3.4.3 Google Maps

Google Maps est un service gratuit de cartographie en ligne, créé par Google. Il propose des images satellitaires et des cartes numériques à différentes échelles d'un pays. Deux types de vue sont disponibles : une vue en plan classique, avec les nom des rues, quartier, villes et une vue panoramique, qui couvre aujourd'hui le monde entier [66].

3.3.4.4 Gps

Le GPS est un système de positionnement par satellites, capable de donner la position géographique avec une précision allant de l'ordre de centaines de mètres jusqu'à quelques centimètres. En plus de la longitude et de latitude, il peut fournir des informations supplémentaires comme la vitesse de déplacement et l'heure locale.

Pour les civils, il sert essentiellement à pouvoir se repérer dans un milieu inconnu, savoir où est le nord géographique et eventuellement comment joindre un lieu précis à partir de ses coordonnées géographique [47].

3.3.5 Plateformes

3.3.5.1 Définition

Une plateforme en informatique est une base de travail à partir de laquelle on peut lire, écrire, développer et utiliser un ensemble de logiciels. Elle est composée de [59] :

1. Matériel (exemple : microprocesseur) ;
2. Système d'exploitation mobile : Un système permettant, entre autres la gestion de la connectivité sans fil et l'interfaçage avec d'autres applications externes [60] ;
3. Des outils logiciels :
   – de développement :

- Bibliothèque logicielle ;
- Editeur de texte, compilateur ;
- Environnement de développement intégré.

– de gestion de projet :

- Gestionnaire de bugs.

– de gestion de base de données (SGBD) :

- MySQL ;
- Oracle.

– de serveur web :

- Apache.

3.3.5.2 Les principales plateformes

Les appareils mobiles sont des appareils à ressources limitées ou faibles. C'est pourquoi, une adaptation du système d'exploitation à leurs limites est nécessaire. Pour se faire, plusieurs plateformes issues de différentes compagnies sont destinées au développement de systèmes d'exploitations et d'applications mobiles. Parmi elles on peut citer : Symbian OS de Nokia, iOS de Appel's, BlackBerry OS de RIM's, Windows Phone de Microsoft, ubuntu MID Edition d'Ubuntu, Bada de Samsung, Palm, webOS et Android de Google.

1. **Symbian OS**

   Symbian OS est le système d'exploitation mobile développé par la société Symbian , pour le système de messagerie électronique. Symbian est soutenu par plusieurs technologies comme BlackBerry via BlackBerry Connect [1] et Microsoft via Exchange ActiveSync [2], etc.

   La position du Symbian dans le marché des systèmes d'exploitation pour les Smartphones est garantie du fait que Nokia, l'une des plus grande firme de téléphones dans le monde soit son principal actionnaire.

   Cette réduction dans le marché des systèmes d'exploitation a poussé la compagnie

1. Le logiciel BlackBerry Connect permet d'accéder aux comptes de courriels professionnels et personnels, d'envoyer des courriels, d'en recevoir et d'y répondre directement à partir d'un téléphone.

2. ActiveSync est un logiciel de synchronisation développé par Microsoft, il permet à un périphérique portable d'être synchronisé avec un PC de bureau ou bien un serveur d'hybèrgement.

Nokia en janvier 2008 à envisager des plans afin de passer Symbian en une plateforme open source. Selon le calendrier de lancement du Symbian, le code source est lancé en juin 2010. Il est publié sous la licence " la Licence publique Eclipse 1.0. ".

Symbian est extrêmement puissant, il offre un OS de soutien au niveau de la plupart des fonctionnalités que l'on peut trouver dans Palm, Windows Mobile et BlackBerry. C'est une plateforme flexible permettant aux développeurs d'ajouter facilement leurs technologies et leurs infrastructures à la plateforme Symbian. De plus, il est soutenu par de grandes compagnies de l'industrie mobile telles que Nokia, Sony Ericsson, Motorola, etc. [54].

2. **iOS**

iOS est le système d'exploitation mobile développé par Apple pour l'iPhone, l'iPod touch, et l'iPad. C'est l'un des systèmes d'exploitation mobiles les plus avancés au monde.

La plateforme iOS est dotée d'une interface utilisateur conviviale, élégante et intuitive : il suffit d'avoir l'iPhone, l'iPad et l'iPod touch en main pour aussitôt les maîtriser.

Les mises à jour iOS sont gratuites et faciles à télécharger du fait qu'elles sont notifiées automatiquement par l'appareil dès leurs apparitions[55].

3. **Blackberry OS**

BlackBerry OS est un système d'exploitation multitâche propriétaire conçu pour les téléphones mobiles de la gamme BlackBerry, par la société canadienne Research In Motion (RIM) [56].

BlackBerry OS fut créé à l'origine pour le marché des hommes d'affaires. Il est doté donc de différentes applications pour le business tels que : les messages électronique, les messages texte (SMS), les messages MMS, le BlackBerry Messenger, le web navigateur, le mémo, etc. Ce système est surtout connu pour son support natif des courriels à travers le protocole Mobile Information Device Profile (MIDP). En effet, ce système permet un service de messagerie électronique pour les entreprises en utilisant BlackBerry Enterprise Server. Il soutient aussi la technologie de la pièce jointe de différents types tels que les fichiers avec les extensions .zip, .html, .doc,

.dot, .ppt, .pdf, etc. Dotés d'une molette, d'un clavier QWERTY et d'une interface utilisateur, les BlackBerry sont d'excellents appareils pour faciliter l'utilisation du système de courriel [54].

Maintenant, BlackBerry devient de plus en plus populaire dans le monde et offre de multiples fonctionnalités pour le grand publique comme le multimédia (l'appareil photo, le camera, etc.).

4. **Windows Phone**

Windows phone, dit aussi Windows Mobile est développé par la firme Microsoft. C'est une plateforme propriétaire et non open source, crée à l'origine pour être une version mobile de windows avec une interface utilisateur conforme à la version actuelle de windows.

Windows Mobile supporte beaucoup de types d'audio, vidéo et apporte même la possibilité de télécharger, de jouer de la musique et de regarder le TV en ligne.

La messagerie électronique du Windows Mobile n'est pas aussi performante que celle du Blackberry. En effet, la messagerie électronique est moins sécurisée dans le windows mobile.

Windows mobile utilise la même collection d'APIs que Windows ce qui facilite le développement d'applications pour les développeurs du Microsoft et l'exécution pour les applications de Microsoft, ce qui n'est pas avantageux pour des applications et des développeurs autres que ceux du Microsoft.

Dans ces nouvelles versions (exemple : version 6), Windows Mobile a amélioré la capacité de connexion et de compatibilité avec Microsoft Office. Il a intégré plusieurs applications de communication comme Windows Live, hotMail, Messenger, etc. Maintenant, il y a plusieurs appareils qui marchent sous Windows Mobile dans le monde comme quelques modèles du Motorola, Samsung, T-Mobile, etc. [54, 57].

5. **Ubunto MID Edition**

Ubuntu MID [3] Edition est un système en code open source, développé par la communauté Ubuntu et parrainé par les sociétés Canonical [4] et Intel. Les MIDs (Mobile Internet Device) peuvent être des terminaux mobiles multimédia dotés ou non de fonctions de téléphonie, créée sous l'idée de "mini - ordinateur portable ". L'usage principal des MIDs est la navigation Internet en mobilité. Comme Android, grâce au noyau Linux, Ubuntu MID Edition est sûr.

Il est très souple, personnalisable et simple d'utilisation. Il supporte aussi des variétés d'application de Web 2.0 comme le navigateur Web, la messagerie électronique, la caméra, le VoIP, la Messagerie instantanée, le GPS, le blog, la TV, les jeux électroniques, etc. Pour la technologie de communication, il soutient le Wi-Fi, Bluetooth, GPS et WiMAX.

Maintenant, Ubuntu MID Edition est utilisé dans quelques modèles des appareils utilisant une plate-forme Intel Atom, Nokia N800 web tablette, etc.

Au final, Ubuntu MID Edition inclue les mêmes services d'Internet qu'on trouve sur ordinateurs, mais utilisée généralement pour les MIDs et non pas pour les téléphones portables [54].

6. **Bada OS**

Bada est le système d'exploitation pour smartphones, propriètaire de Samsung. Il est développé au début de l'année 2010, après avoir été dévoilé au Mobile World Congress (le congrès mondial du mobile) de Barcelone dans la même année.

Bada peut fonctionner d'une manière variée selon l'appareil sur lequel il est utilisé. Les périphériques utilisant Bada comportent toujours trois touches : la touche centrale, la touche Répondre et la touche Raccrocher.

La première sert à afficher la liste des applications présentes dans l'appareil et/ou à quitter une application, en la laissant en arrière-plan.

La deuxième sert à répondre uniquement aux appels. Il y a aussi une touche

3. MID sont généralement de petites tablettes portables conçues pour les activités de navigation web et du multimédia.

4. Canonical Ltd est une société fondée (et financée) par l'entrepreneur sud-africain Mark Shuttleworth, et dont l'objet est la promotion de projets open source (code source libre).

volume, présente à gauche. Quant à La troisième, elle sert à raccrocher lors d'un appel, à afficher les widgets présents sur l'appareil ou à fermer l'application, bien que certaines applications natives, comme Musique, restent actives [58].

7. **Palm OS**

   Palm OS est le système d'exploitation développé par la société Palm. Il est facile à utiliser et simple à apprendre. Il permet de minimiser les étapes pour naviguer entre les écrans et choisir les applications. Par exemple, pour lancer un programme il suffit d'appuyer sur son icône et dès qu'on passe à un autre, l'application en cours prenne fin automatiquement.

   Palm OS a vu son évolution passée des versions mono-tâches (5.0 et ultérieur) en version multitâche (6.0). Le fondement de cette nouvelle version se base sur les versions mono-tâches précédentes et offre plusieurs améliorations comme la communication et le multimédia [54].

8. **WebOS**

   webOS est un système d'exploitation entièrement open source basé sur un noyau Linux, développé comme une dernière version de Palm. Il est très différent des anciennes versions, il se base sur les applications utilisant HTML, JavaScript et CSS.

   Une application webOS est lancée soit par un utilisateur ou par une autre application. Elle supporte le multitâches, mais peut être conçue pour fonctionner uniquement en arrière-plan. Cette applications d'arrière-plan interagit avec l'utilisateur principalement en affichant des alertes et des notifications.

   En 2013, Samsung annonça l'arrêt du développement de Bada pour se concentrer sur le développement d'une nouvelle plateforme baptisée Tizen [57].

9. **Android**

   Android est un système d'exploitation développé par Google pour téléphones portables de nouvelle génération, PDA et autres terminaux mobiles (tablette). Il se base sur un noyau Linux 2.6. Cette plateforme met à disposition un kit de développement (SDK) basé sur le langage Java. Il possède donc une bibliothèque de plusieurs classes java servant de base pour plusieurs type d'application (exemple :

SSL pour les protocoles de sécurité).

Android a été conçu pour intégrer au mieux les applications existantes de Google, pour permettre un accès rapide aux services d'internet comme le service de courrier Gmail, YouTube, Google Talk, l'agenda Google Calendar et encore la cartographie Google Maps.

Afin de promouvoir ce nouveau système d'exploitation ouvert, Google a su fédérer autour de lui un consortium d'une trentaine d'entreprises : l'Open Handset Alliance (OHA). Le but de cette alliance est de mettre en place des normes standards dans le domaine de la téléphonie mobile. Ce qui permet aux développeurs d'applications Android d'obtenir la permission d'intégrer, d'agrandir et de remplacer des composants existants. De plus, il ne fait aucune distinction entre les applications natives et les applications développées par des tierces parties, car ils accèdent aux mêmes APIs. L'utilisateur peut donc personnaliser facilement son appareil, adapter des applications à ses besoins ou les remplacer entièrement [58].

En conclusion, Android se montre à la tête des systèmes d'exploitation vue la flexibilité de sa plateforme et offre la possibilité aux développeurs de créer des applications extrêmement riches et innovantes.

3.4 Conclusion

On assiste à une évolution d'applications plus orientés périphériques mobiles où les fabricants d'appareils vont au-delà du marché d'ordinateurs classiques et prennent part à une nouvelle création de valeur basé sur les services et le contenu.

Le but de ce chapitre était de présenter les différentes plateformes mobiles afin de pouvoir identifier leurs forces et leurs faiblisses pour enfin déterminer la plateforme qui conviendra le mieux à notre cas d'étude. Suite à l'analyse faite après la présentation de différentes plateformes, Android s'avère le choix qui nous est le plus approprié.

Chapitre

4 Conception de l'application

4.1 Introduction

Le recours à la modélisation est une pratique indispensable au développement logiciel. Elle a pour rôle de cerner les problèmes : les identifier, trouver leurs solutions, schématiser ces dernières, puis enfin préparer le terrain d'action.

Un modèle est, en effet, une représentation abstraite d'un système afin d'en faciliter son étude et sa documentation. C'est un outil majeur de communication entre les différents intervenants au sein d'un projet.

En outre, les systèmes devenant de plus en plus complexes, leur compréhension et leur maîtrise globale dépassent les capacités d'un seul individu. La construction d'un modèle abstrait aide à y remédier.

Ce chapitre sera consacré à la conception de notre application. D'abord, nous introduisons le langage UML et ses différents diagrammes. Ensuite, nous présenterons la modélisation proprement dite de notre projet en utilisant cinq types de diagrammes.

4.2 Présentation d'UML

Dans le cadre de notre projet, nous avons opté pour un langage de modélisation unifié, baptisé UML.

4.2.1 Définition

UML (Unified Modeling Language) est le résultat d'une opération d'unification d'un ensemble de concepts pris à partir des méthodes orientées objets dans le but de

modéliser d'une manière claire et précise la structure et le comportement d'un système, indépendamment de toute méthode et tout langage de programmation [68].

UML est un langage standard de modélisation graphique. Il permet d'appréhender visuellement la complexité d'un produit logiciel en termes de composants, d'architecture et de liens entre les éléments de conception.

Il unifie à la fois les notations et les concepts orientés objet. En effet, il ne s'agit pas d'une simple notation graphique. Les concepts transmis par un diagramme ont une sémantique précise et sont porteurs de sens au même titre que les mots d'un langage parlé [69].

L'utilisation d'UML est un choix à faire compte tenu des opportunités qu'il offre et des exigences du travail entamées. Il nous propose un nombre de diagrammes qui nous assistent durant l'analyse des besoins, la conception de données, l'implémentation et le déploiement du système [79].

4.2.2 Présentation générale des Diagrammes d'UML

La version UML 2.0 s'articule autour de treize (13) diagrammes différents et compte, par conséquent quatre nouveaux diagrammes par rapport à la version antérieure. Chaque diagramme est destiné à la représentation d'un système logiciel suivant une vision spécifique. UML modélise le système suivant deux modes de représentation : l'un concerne la structure du système pris "au repos" et l'autre concerne sa dynamique de fonctionnement. Les deux représentations servent de référence pour schématiser les composants du système et l'interaction fonctionnelle entre elles [70].

4.2.2.1 Les diagrammes structurels

Ces diagrammes, au nombre de six, servent à représenter l'aspect statique d'un système (classes, objets, composant, etc.) [70] [79] [80] [81].

1. **Diagramme de classes :** considéré comme le plus utilisé des diagrammes, il donne la description des structures de données d'un système sous forme de classes (schéma d'objet).

2. **Diagramme d'objets :** est basé sur les éléments du diagramme de classe, il permet la représentation d'instances des classes et des liens entre instances.
3. **Diagramme de composants :** représente les différents constituants du logiciel au niveau de l'implémentation d'un système (base de données, fichier).
4. **Diagramme de déploiement :** décrit l'architecture technique d'un système et il sert à représenter les éléments de l'environnement où le système sera déployé (machine, réseau, etc.).
5. **Diagramme de paquetage :** donne une vue d'ensemble du système structuré en paquetage. Chaque paquetage représente un ensemble homogène d'éléments du système (classes, composants, etc.).
6. **Diagramme de structure composite :** permet de décrire la structure interne d'un ensemble complexe composé par exemple de classes ou d'objets et de composants techniques. Ce diagramme met aussi l'accent sur les liens entre les sous-ensembles qui collaborent.

4.2.2.2 Les diagrammes de comportement

Ces diagrammes représentent la partie dynamique d'un système réagissant aux évènements et permettant de produire les résultats attendus par les utilisateurs. Sept diagrammes ont été proposés par UML [66] [71] [80] [81] :

1. **Diagramme des cas d'utilisation :** est destiné à représenter les besoins fonctionnels des utilisateurs par rapport au système en faisant abstraction de tout aspect technique. Il constitue un des diagrammes les plus structurants dans l'analyse d'un système.
2. **Diagramme d'état-transition :** montre les différents états (états simples, les transitions et les états composites imbriqués) et comportements des objets en réaction aux évènements. L'état d'un objet à un instant T peut changer à l'instant T+1.
3. **Diagramme d'activités :** donne une vision des scénarios qui se déroulent dans le système. Ceux-ci sont propres à une opération ou à un cas d'utilisation. Il permet aussi de représenter les flux de contrôle et les flux de données.

4. **Diagramme de séquences :** permet de décrire les scénarios de chaque cas d'utilisation en mettant l'accent sur la chronologie des opérations en interaction avec les objets. [71]
5. **Diagramme de communication :** c'est une autre représentation simplifiée des scénarios des cas d'utilisation qui met plus l'accent sur les objets et les messages échangés entre eux.
6. **Diagramme global d'interaction :** ce diagramme fournit une vue variante de l'ensemble des changements de l'exécution d'un diagramme d'activité. Les noeuds sont des interactions qui permet d'associer les notations du diagramme de séquence à celle du diagramme d'activité, ce qui permet de décrire une méthode complexe.
7. **Diagramme de temps :** ce diagramme permet de représenter les états des interactions où l'aspect temporel est mis en valeur. Il permet de modéliser les contraintes d'interactions entre plusieurs objets, comme le changement d'état en réponse à un évènement extérieur.

4.2.3 Processus associés à UML

Des processus ont été définis et associés à UML vue qu'il n'existait aucune démarche unifiée pour construire des modèles et conduire un projet qui le met en oeuvre.

Un processus de développement est un ensemble d'étapes partiellement ordonnées, qui définit **qui** fait **quoi**, **quand** et **comment** dans le but de produire des logiciels :

– De qualité (qui répondent aux besoins de leurs utilisateurs) ;
– Dans des temps et des coûts prévisibles et raisonnable.

Et à chaque étape, on produit :

– Des modèles ;
– De la documentation ;
– Du code.

Il en existe plusieurs processus pour UML. On peut citer UP (Unified Process), XP (eXtreme Programming) et RUP (Rational Unified Process) qui n'est d'autre qu'une dérivée d'UP [71].

Nous avons choisi le processus RUP qui est un processus de développement logiciel itératif et incrémental, centré sur l'architecture, conduit par les cas d'utilisation.

4.2.3.1 Le processus unifié de rational (RUP)

En son temps, la société Rational Software (rachetée par IBM) avait développé une version spécifique d'UP sous le nom de RUP (Rational Unified Process). Dans la présentation qui suit, nous avons surtout mis l'accent sur les principaux apports de RUP.

RUP (Rational Unified Process) est un processus de développement logiciel qui couvre tout le cycle de vie du projet et guide l'équipe de développement dans les activités de gestion du projet ainsi que les activités techniques.

RUP est un processus basé sur une approche disciplinée afin de bien maîtriser l'assignation des tâches et la responsabilisation des différents acteurs participant au cycle de développement du logiciel [70].

4.2.3.2 Les principaux apports de RUP

1. **Les bonnes pratiques**

   RUP adhère à six bonnes pratiques de développement. Sur ces six bonnes pratiques, trois sont issues des principes d'UP [70] :

   – Développement itératif et incrémental ;
   – Développement piloté par les cas d'utilisation ;
   – Développement centré sur l'architecture 4+1.

   Trois autres bonnes pratiques ont été introduites par RUP [67,70] :

   – Modélisation visuelle ;
   – Vérification continue de la qualité (Livraison de qualité) ;
   – Adaptable aux changements.

2. **Développement itératif et incrémental**

   Le développement d'un produit logiciel est une vaste opération qui peut prendre beaucoup de temps. Il est donc utile de découper le travail en plusieurs parties qui sont autant de mini-projets. Chacun d'eux représente une itération qui prend en compte un certain nombre de cas d'utilisations et qui donne lieu à un

incrément. Les itérations désignent des étapes de l'enchaînement d'activités, tandis que les incréments correspondent à des stades de développement du produit [74].

Parmis les avantages du développement itératif, on peut citer [70] :

- Les risques sont évalués et traités au fur et à mesure des itérations ;
- Les premières itérations permettent d'avoir un feed-back des utilisateurs ;
- Les tests et l'intégration se font de manière continue ;
- Les avancées sont évaluées au fur et à mesure de l'implémentation.

3. **Développement guidé par les cas d'utilisation**

   Le but étant de montrer que le système à construire se définit d'abord avec les utilisateurs. Les **cas d'utilisation** permettent d'exprimer les interactions du système avec les utilisateurs, donc d'identifier les besoins. Ils constituent un vecteur structurant pour le développement et les tests du système ce qui permet une décomposition du développement par cas d'utilisation. De même, la conception du logiciel sera elle aussi séparée par cas d'utilisation [70].

4. **Développement centré sur l'architecture et la vue 4+1**

   Il est important de définir l'architecture type qui sera retenue pour le développement, l'implémentation et ensuite le déploiement du système dès le début des travaux d'analyse et de conception. Une architecture type constitue le lien et motive la création de tous les diagrammes d'architecture. C'est pourquoi le processus RUP donne de l'importnace à cette architecture [70].

   RUP adopte alors le modèle de Philippe Kruchten [75] dit modèle des 4+1 vues. Une vue de l'architecture est la description d'un système d'un point de vue particulier, couvrant certains points et omettant certains autres. Cette architecture identifie 4 vues +1 [70,76] :

   - **La vue logique :** concerne les exigences fonctionnelles du système, elle décrit de façon statique et dynamique le système en termes d'objets et de classes. Elle est représentée principalement, par des diagrammes statiques de classes et d'objets enrichis par des descriptions dynamiques : diagrammes d'activités, de séquences, diagrammes de communications ou d'états-transitions ;

- **La vue implémentation (réalisation) :** décrit l'organisation du logiciel. Elle permet de visualiser l'organisation des composants (bibliothèque dynamique et statique, code source, etc.).
- **La vue déploiement :** montre comment les différents exécutables sont structurés dans la plateforme. Les diagrammes de cette vue sont ceux de composants et de déploiement ;
- **La vue du processus :** concerne les aspects concurrents du système à exécuter. Elle permet d'exprimer la synchronisation et l'allocation des objets. Les diagrammes utilisés dans la vue des processus sont exclusivement dynamiques : diagrammes d'activités, de séquence, diagrammes de communication ou d'états-transitions ;
- **La vue de cas d'utilisation :** contient les principaux scénarios qui sont utilisés pour faire fonctionner l'architecture de logiciel. Chaque cas d'utilisation peut être décrit par un ou plusieurs diagrammes dynamiques.

5. **Modélisation visuelle**

   RUP préconise l'utilisation du langage de modélisation standard UML qui permet de gérer la complexité et rendre transparente la communication entre les intervenants dans le projet.

   Il recommande l'utilisation d'outils de modélisation visuelle qui permettent de [70] :

   - Modéliser l'architecture et ses composants à l'aide de diagrammes ;
   - Faciliter la gestion des modèles de RUP et contribuer à la maintenance de la cohérence entre les différentes phases du processus : de l'expression des besoins à l'implémentation ;

6. **Vérification continue de la qualité**

   RUP met en valeur l'évaluation de la qualité d'un système, en termes de fonctionnalités, de fiabilité et de performance. Pour cela, il nous assiste dans la planification, la conception, l'implémentation et l'exécution des tests adéquats [70].

7. **Contrôle des changements du logiciel**

   Combinée au développement itératif, RUP permet de contrôler les changements de sorte qu'il soit possible de découvrir rapidement les éventuels problèmes et d'y remédier. Il propose une coordination des activités et des livrables afin de gérer activement les changements du logiciel [70].

4.2.3.3 Les étapes de processus RUP

Le processus unifié de Rational, organisé en fonction du temps, est divisé en quatre phases successives [70].

- Lancement (Inception) ;
- Elaboration ;
- Construction ;
- Transition.

1. **Lancement (Inception)**

   Cette phase correspond à **l'initialisation du projet** où l'on mène une étude de faisabilité du système à construire et ce n'est qu'à son issue que l'on peut considérer le lancement du projet. C'est donc à ce niveau que **les objectifs sont fixés.**

   Au cours de cette phase, les principaux scénarios accompagnés d'une description générale doivent être identifiés et modélisés dans un diagramme de cas d'utilisation [70].

2. **Elaboration**

   Cette phase reprend les résultats de la phase de lancement et élargit l'évaluation de **la faisabilité** sur la quasi-totalité des scénarios (cas d'utilisation) qu'on retrouve dans le diagramme des cas d'utilisation.

   Elle a pour but d'analyser le domaine technique du système à développer afin d'aboutir à une architecture stable. Ainsi, toutes les exigences non recensées dans les cas d'utilisation, seront prises en compte dans la conception et l'élaboration de l'architecture.

   Au cours de cette phase, les cas d'utilisations sont modélisés dans des diagrammes de séquences. L'architecture logique du système est dégagée sous forme

d'un diagramme de classe [70].

3. **Construction**

   Cette phase correspond à **la production** d'une première version exploitable du produit. Elle est donc fortement centrée sur les activités de conception, d'implémentation et de test. En effet, les composants et les fonctionnalités non implémentés dans la phase précédente le seront ici.

   Au cours de cette phase, la gestion et le contrôle des ressources ainsi que l'optimisation des coûts représentent les activités essentielles pour aboutir à la réalisation du produit [70].

4. **Transition**

   C'est la phase **de livraison de produit** pour une exploitation réelle où les actions liées au déploiement sont traitées.

   De plus, des "bêta tests" sont effectués pour valider le nouveau système auprès des utilisateurs [70].

   Remarque : *Cette phase ne pourra être vue, car ce projet ne peut être considéré comme étant commercial.*

4.3 Modélisation de l'application

On associe à chaque phase du processus RUP une description associée à notre application qui montrera les diagrammes qu'offre UML qui pourront être utilisés.

4.3.1 Première phase (Inception)

Cette première phase consiste à lancer notre application et notamment de spécifier les besoins et les différents scénarios.

4.3.1.1 Problématique

Le travail se focalise particulièrement sur la gestion de contexte et son utilisation pour la découverte de services dans un environnement ubiquitaire (musée). Les travaux antérieurs se focalisent seulement sur la découverte des services pour répondre à la requête de l'utilisateur et cela sans prendre en compte les informations contextuelles telles que la localisation, le temps, le réseau, l'environnement et les préférences de l'utilisateur.

La faiblesse de la quasi-totalité des architectures vues en chapitre II, réside dans le fait qu'elles se basent toutes sur des serveurs centralisés. Les fréquents changements des informations contextuelles rendent leur gestion très compliquée. De ce fait l'utilisation d'un seul serveur central dans un environnement ubiquitaire entrave la disponibilité des informations contextuelles et ne permet pas le passage à l'échelle et la tolérance aux pannes.

4.3.1.2 Solution proposée

La proposition faite par BOUANANI.Z et GUANEM.S en 2012 [8] a comme principal objectif de développer une architecture de gestion de contexte décentralisé et de l'utiliser pour le développement d'une architecture de découverte de services sensibles au contexte dynamique, automatique et avec un minimum d'intervention de la part de l'utilisateur (figure 4.1).

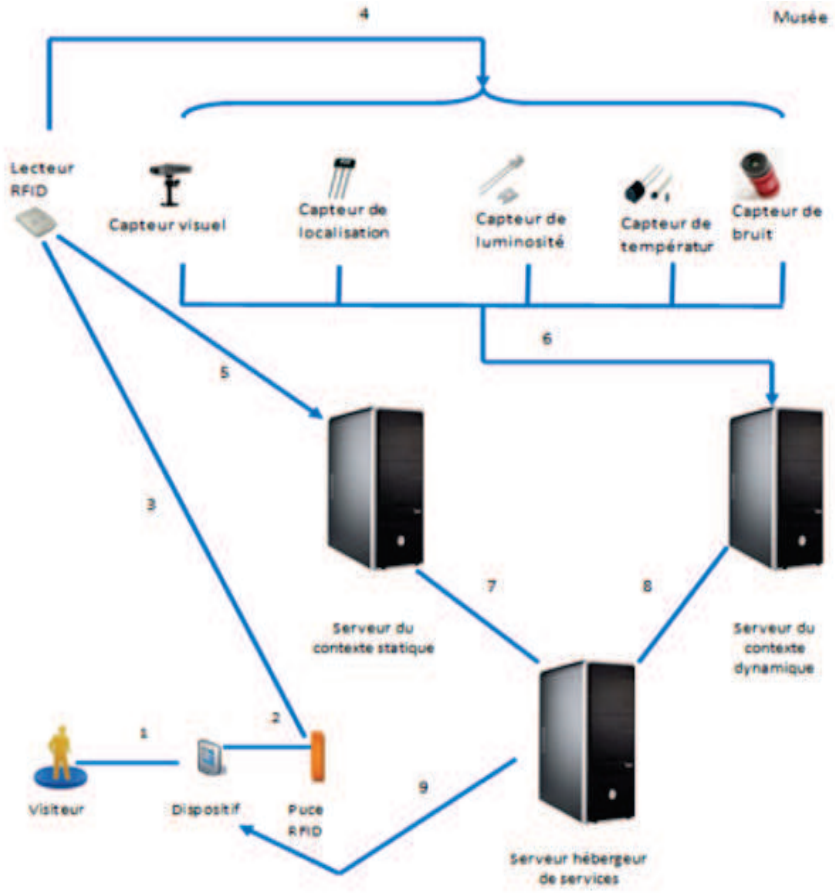


FIGURE 4.1 – Architecture proposée par BOUANANI.Z et GUANEM.S. [8]

Une fois le visiteur est dans le musée, il sera représenté par son dispositif qui est composé d'une puce RFID. Le lecteur RFID détecte sa présence et collecte son contexte statique grâce au formulaire installé dans ce dernier.

Le lecteur RFID envoie l'identifiant de cet utilisateur à l'ensemble des capteurs physiques qui se trouvent à sa proximité pour initier la collecte des informations du contexte dynamique et envoie les informations du contexte statique collectées au serveur du contexte statique.

Quant aux capteurs physiques, ils envoient les informations du contexte dynamique au serveur du contexte dynamique. Une fois que ces informations sont collectées le serveur hébergeur des services les récupère du serveur du contexte statique et depuis le serveur du contexte dynamique et d'adapter le service au contexte de l'utilisateur selon ses besoins.

Comme indiqué dans l'introduction, le processus contient deux parties. La première consiste en la gestion des informations contextuelles et la deuxième concerne découverte des services existants dans l'environnement, en se basant sur le résultat de la gestion de contexte.

4.3.1.3 Faisabilité du système

Après avoir analysé l'architecture proposée, nous estimons que cette solution est réalisable. Néanmoins, des changements indispensables sont requis en tenant compte des moyens soumis à notre disposition (exemple : un capteur physique sera représenté par un simple programme logiciel vu la non disponibilité des capteurs matériels).

4.3.1.4 Description du projet

1. **Architecture du musée**

   Notre projet consiste à concevoir et à réaliser une application sensible au contexte pour un bâtiment de type musée que l'on suppose constitué d'un rez-de-chaussée et de quatre étages :

   – Rez-de-chaussée ;
   – **Etage 1 :** histoires et civilisations ;
   – **Etage 2 :** Arts ;
   – **Etage 3 :** Education ;
   – **Etage 4 :** Informatique.

   Chaque étage se divise en un ou plusieurs départements comme suit :

   – **Histoire et civilisations :** département d'histoire et département de civilisations ;
   – **Art :** département de sculptures et statues, et un département de peintures ;
   – **Education :** département formations et conférences, départements librairie ;
   – **Informatique :** département informatique.

   Chaque département se divise en une ou plusieurs salles :

   – **Histoire :** ce département se divise en une grande salle dont on trouve des ouvrages, des multimédia et des photos en relation avec l'histoire du musée ;

- **Civilisations :** ce département se divise selon les thèmes : antiquités égyptiennes, grecs et romaines, chaque salle contient des oeuvres qui sont localisées chacune d'elles dans une cellule ;
- **Sculptures :** ce département contient une variété de sculptures et de statues placées dans une grande salle, chaque sculpture ou statue est dans une cellule ;
- **Peintures :** ce département comprend une large salle exposant une multitude de tableaux et de toiles ;
- **Formations et conférence :** ce département se divise en deux salles, une salle d'attente et une grande salle de conférence là où les formations et conférences sont animées ;
- **Librairie :** ce département propose des milliers d'ouvrages portant sur les collections et les expositions du musée, ainsi que des revues spécialisées, des cédéroms et des films vidéo. Elle dispose d'une riche sélection d'éditions sur le musée, son histoire, ses collections et ses expositions ;
- **Informatique :** contient une salle serveurs et un bureau pour l'administrateur ;
- **Le rez-de-chaussée :** il se divise en quatre salles, salle d'accueil, salle d'archive, une salle d'attente et un magasin de souvenir.

2. **Le contexte utilisateur**

Les informations contextuelles de notre application sont centrées sur le visiteur et se répartient comme suit :

- **Le contexte statique :** regroupe les informations en relation avec le profil de l'utilisateur : date de naissance, centre d'intérêt, profession, etc.
- **Le contexte dynamique :** concerne principalement la localisation du visiteur.

3. **Architecture de l'application à réaliser**

L'architecture de l'application à réaliser est illustrée dans la figure ci dessous :

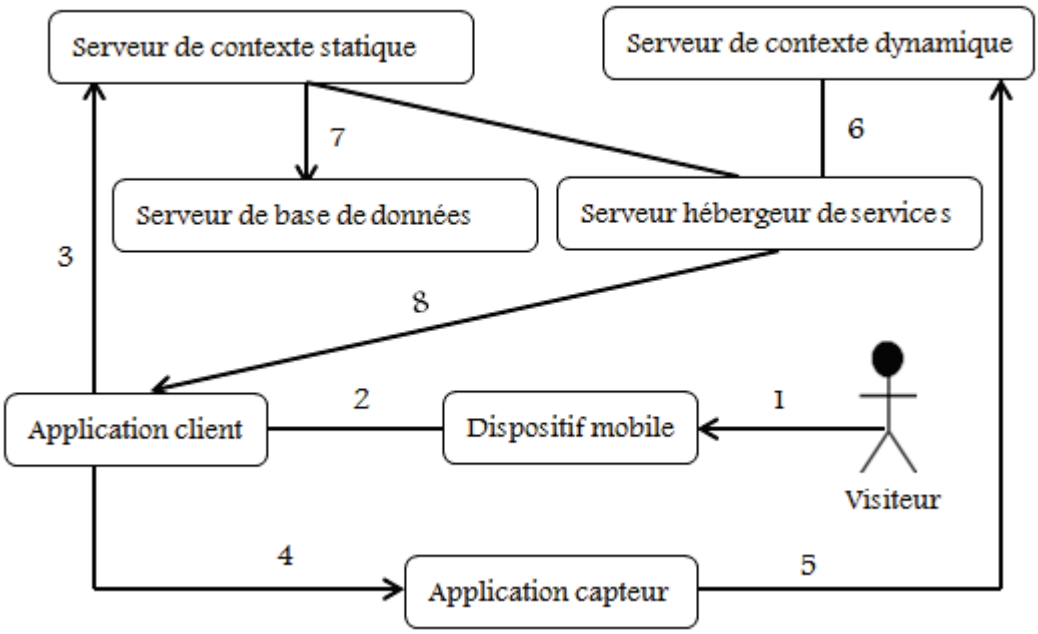


FIGURE 4.2 – Architecture de notre application.

4. **Déroulement de l'application**

A l'entrée au musée, le visiteur se rend à l'accueil. Si ce dernier dispose d'un téléphone à base d'un système d'exploitation "Android", l'agent du musée se contente de lui donner l'application à installer. Sinon il met à sa disposition un dispositif muni de l'application.

Une fois l'application est installée, le visiteur démarre cette dernière en cliquant sur l'icône qui lui est associée. Une page d'accueil s'affiche et propose à l'utilisateur de lancer l'application, de s'inscrire ou de quitter.

Le visiteur doit d'abord s'inscrire avant de pouvoir utiliser l'application. Pour se faire, il remplit un formulaire puis il le valide. A ce moment-là, les informations introduites seront enregistrées dans la base de données afin de pouvoir les exploiter ultérieurement par le serveur de contexte statique.

Pour lancer l'application, le visiteur s'authentifie en introduisant son identifiant et son mot de passe. Une interface sera donc affichée contenant un message qui demande de choisir le type de visite : libre ou guidé (que nous allons définir ci-dessous). Ensuite, une nouvelle interface lui sera affichée selon son choix.

L'application capteur sera appelée en faisant passer l'identifiant du visiteur comme paramètre et le serveur du contexte statique récupère le profil du visiteur à partir de la base de données, il effectuera en suites les tests adéquats pour détecter d'éventuels services et interrogera le serveur hébergeur de services pour lancer le service et l'afficher dans le périphérique du visiteur. L'application capteur quant à elle se charge de générer des positions du visiteur dans le musée, ces dernières seront envoyées au serveur du contexte dynamique qui va à son tour les tester en les comparant aux positions des cellules pour savoir devant quelle oeuvre le visiteur se trouve-t-il, pour enfin interroger le serveur hébergeur de services pour lancer le service.

5. **Processus détaillé**

L'application doit proposer des services concordants avec les préférences des utilisateurs. Ces utilisateurs ont des préférences, des souhaits relatifs à ce qu'ils vont pouvoir faire avec cette application. L'application doit leur fournir ces services. C'est pourquoi la première étape de cette démarche consiste à définir tous les services que l'application peut fournir. Cette description est faite en langage naturel (manière déclarative) à la façon d'un cahier des charges.

(a) **Acteurs**

Un acteur est un rôle joué par l'utilisateur du système logiciel. Les acteurs peuvent être des personnes physiques ou des systèmes automatisés. Ils se trouvent obligatoirement à l'extérieur du système [78].

Dans notre application de visite d'un musée, nous identifions tout d'abord trois types d'utilisateurs :

- **Visiteur libre :** il souhaite visiter librement le musée au gré de ses préférences et obtenir des informations sur les oeuvres ;
- **Visiteur guidé :** il sera assisté dès son entrée dans le musée afin de ne pas rater une oeuvre incontournable ;
- **Administrateurs (le personnel en charge du musée) :** ils souhaitent pouvoir gérer au mieux les visites dans le musée en termes de nombre de visiteurs par jour/mois en temps réel afin d'évaluer le taux de fréquenta-

tions. Ils souhaitent également connaître le type d'oeuvres qui intéressent chaque visiteur afin de les notifier en cas de présence de nouveaux services concernant ces derniers.

(b) **Les services**

Afin de répondre à l'ensemble des besoins d'un visiteur, notre application est en mesure de fournir les services suivants :

- **Service d'inscription**

  Avant qu'un visiteur puisse utiliser l'application, il est indispensable de passer par le service d'inscription. C'est à ce niveau que l'on aura le profil des visiteurs, donc leurs préférences et leurs souhaits.

- **Service de Description**

  Permet d'envoyer aux visiteurs des descriptions en temps réel des oeuvres devant les quelles ils se sont arrêtés pour une durée bien déterminée. Ces descriptions indiquent les informations relatives à l'oeuvre (nom, date et le lieu de création et d'autres détails).

- **Service de Guidage**

  Ce service est destiné aux utilisateurs qui ont choisi d'être des visiteurs guidés. Ce dernier va assister les visiteurs guidés dès leurs entrée dans le musée en leurs indiquant par où commencer leurs visites et le parcours à suivre.

  Pour les visiteurs libres, le système se contente de leur fournir un plan simple du musée pour qu'ils puissent se repérer dans le musée.

- **Service de Statistiques**

  Propose aux administrateurs du musée une série de données concernant le nombre de visiteurs dans le musée en temps réel, ainsi que le nombre de visiteurs dans la journée.

- **Service d'actualité**

  Il se déclenche automatiquement lors de l'ajout d'une nouvelle actualité de la part de l'administrateur. Il permet, donc, de transmettre les actualités du musée aux visiteurs afin de les inciter à revisiter le musée selon leurs centres d'intérêt.

- **Service de consigne**

  Ce service se déclenche automatiquement lors de la détection d'un nouveau visiteur suite à son inscription par exemple. Il permet donc de transmettre la réglementation à l'intérieur du musée.

- **Services de tarification**

  Transmet aux utilisateurs suite à leurs inscription, les différents tarifs tenant compte de leurs catégories (enfant, étudiant, handicapé, abonné, etc.)

- **Service conférence**

  Renvoie au visiteur des informations sur les conférences qui seront animées au sein du musée lors de sa visite ainsi que celles des jours avenir.

- **Service notification**

  Informe les visiteurs qui ne sont pas de la ville (touristes) de la liste des hôtels, restaurants, parcs d'attraction qui se trouvent à proximité du musée.

6. **Cas d'utilisations**

Un cas d'utilisation est une façon d'utiliser le système. Le terme "cas d'utilisation" est explicite : dans quel cas tel acteur utilise-t-il le système ? Chaque réponse à cette question est donc par définition un cas d'utilisation.

Cette technique d'expression de besoins a été formalisée par jacobson [73]. Les cas d'utilisation font aujourd'hui partie intégrante d'UML et constituent d'ailleurs le premier critère à prendre en compte dans le choix de la méthode associée à UML. Ils font référence aux acteurs, c'est-à-dire aux "objets" externes qui communiquent avec le système [76,77].

Dans notre projet, nous avons les cas d'utilisations suivants :

(a) **Coté visiteur**

- **Créer un compte :** avant qu'un visiteur puisse utiliser l'application, il devra s'inscrire et remplir un formulaire. L'inscription est nécessaire pour avoir les informations contextuelles statiques du visiteur ;
- **Se connecter :** une fois un visiteur est inscrit, il aura son propre identifiant et son propre mot de passe, qui lui permettront de se connecter et d'utiliser

l'application ;

- **Mise à jour du compte :** un visiteur peut également modifier son compte après une authentification afin de mettre à jours son contexte statique (profils).

(b) **Coté administrateur**

Après une authentification, l'administrateur aura accès à sa page de gestion pour effectuer les tâches suivantes :

- **Consulter les statistiques :**l'administrateur peut y voir les statistiques en nombre de visiteurs ;
- **Gérer les oeuvres du musée :**Il aura également la possibilité d'ajouter, de modifier et de supprimer une oeuvre qui peut être soit une peinture, une sculpture ou une statue ;
- **Gérer les services du musée :** l'administrateur pourra ajouter de nouveaux services (conférences, consignes, notifications, etc.), de modifier ceux déjà existants et de supprimer ceux qui sont dépassés afin de préserver le musée et son fonctionnement ;
- **Créer un compte administrateur :**l'administrateur pourra créer un compte afin d'avoir les privilèges associée à un administrateur ;
- **Imprimer le contenu des tableaux :** Le gérant du musée a la possibilité d'avoir les données affichées dans les tableaux dans l'application de gestion sous format papier. Ces tableaux contiennent des informations relatives aux services, aux oeuvres et aux comptes des visiteurs ;
- **Exporter le contenu des tableaux :** Le gérant peut aussi exporter les informations des services, oeuvres et comptes des visiteurs qui se trouvent dans les tableaux vers Excel ;
- **Récupérer le mot de passe :** en cas d'oubli du mot de passe, l'administrateur peut le récupérer.

4.3.2 Deuxième phase (Elaboration)

Cette phase implique une forte utilisation d'UML. Dans ce qui suit nous allons représenter dans Par des diagrammes tous les cas d'utilisation et aussi les séquences d'activités prises en charge pas l'application, puis le séquencement de chaque cas par un diagramme de séquence et/ou d'activité .
Afin de montrer la répartition physique des matériels du système (processeurs, périphériques) et leurs connexions on donnera aussi le diagramme de déploiement.

Ces digrammes seront représentés par le Visual Paradigm For UML qui permet la création des diagrammes UML et des modèles qui en sont à l'origine.

Le paradigme visuel renforce le développement des applications de qualité. Il permet aux clients de comprendre leurs organisations et de traduire les besoins en logiciels de qualité avec des outils qui sont à la fois riche en fonctions et simple à utiliser. Sa modélisation visuelle offre un excellent moyen de communication, surtout en période de resserrement du projet. Il rend la collaboration entre les gens qualifiés dans différents domaines à la fois efficace et fructueuse.

4.3.2.1 Diagrammes des cas d'utilisation

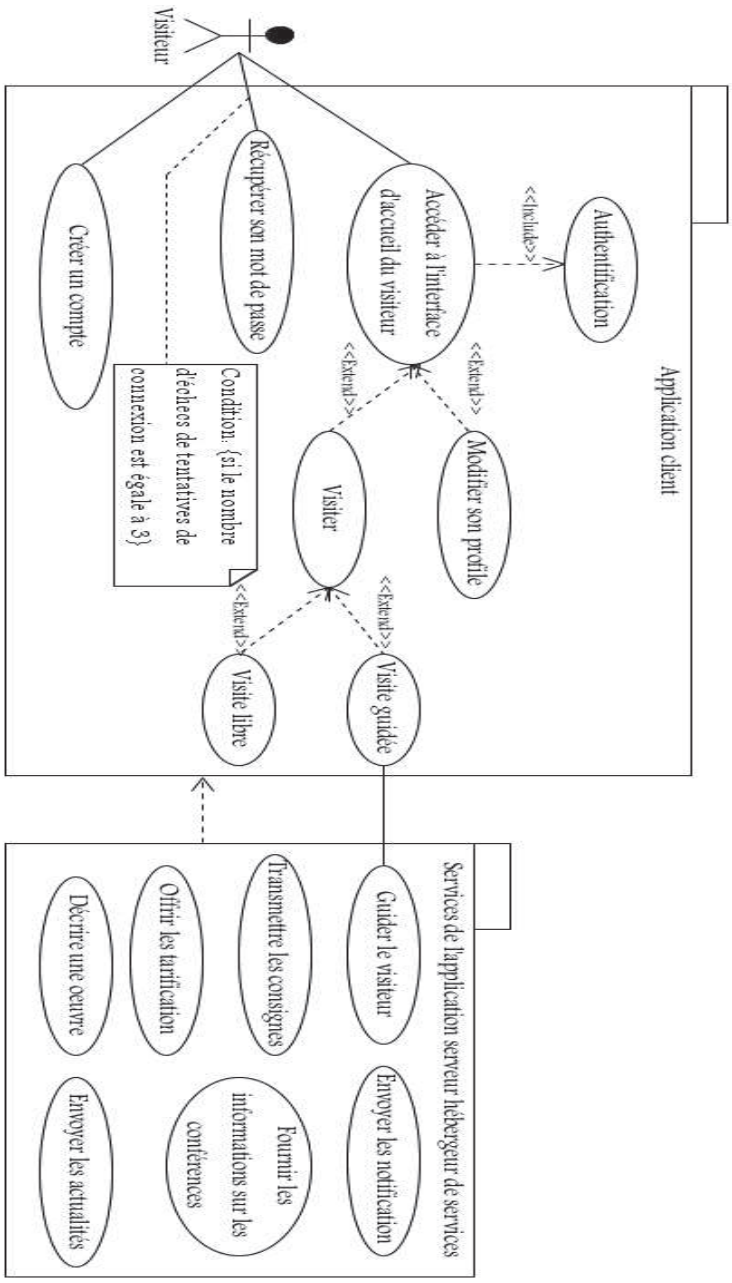


FIGURE 4.3 – Diagramme des cas d'utilisation (Visiteur).

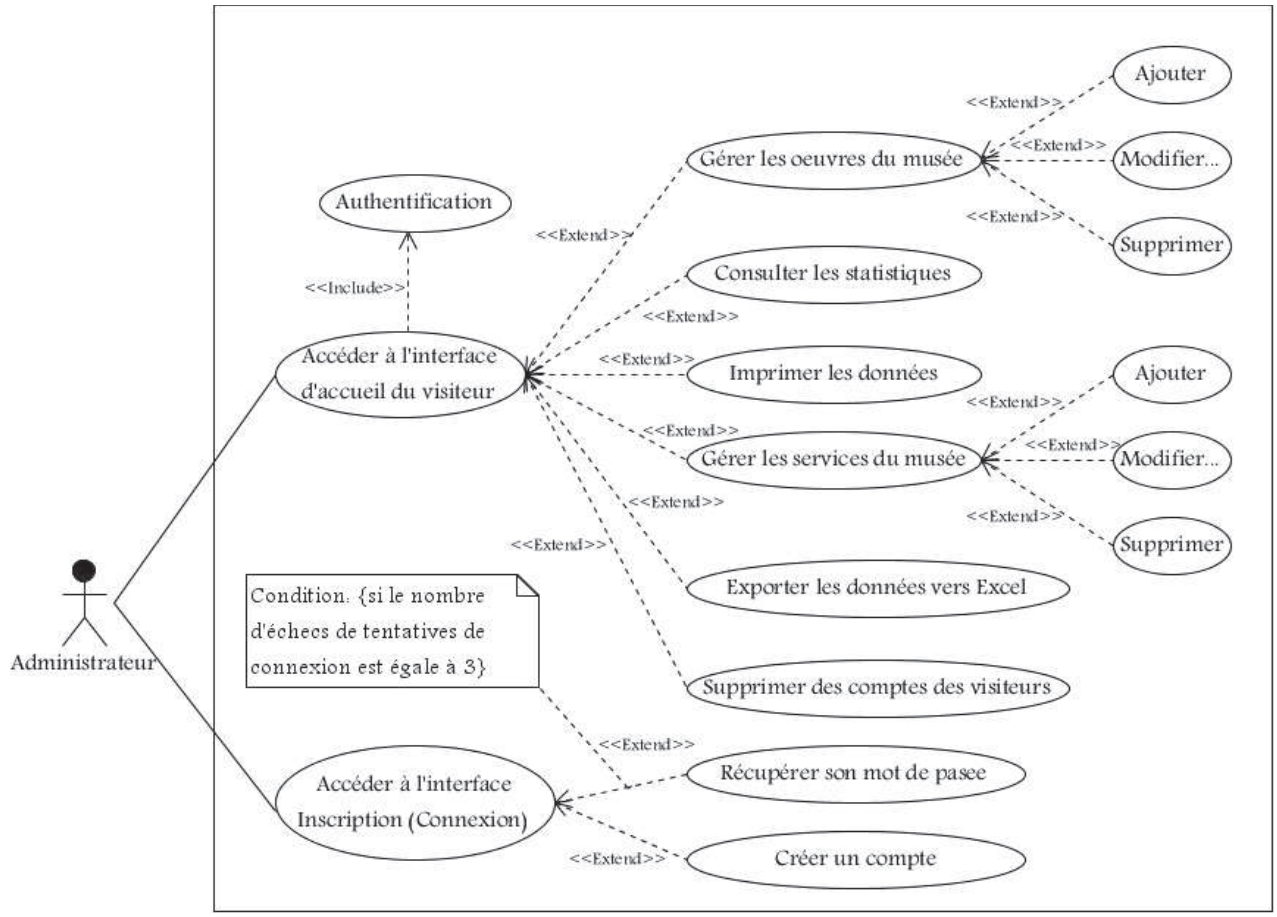


FIGURE 4.4 – Diagramme des cas d'utilisation (Administrateur).

4.3.2.2 Diagrammes de séquences

La description des cas d'utilisation se fait par l'identification des scénarios possibles. Chauqe scénario peut être représentés par un diagramme de séquences .

1. **Diagrammes de séquence des cas d'utilisation "Authentifier"**

   Lorsqu'un visiteur veut accèder à son compte et activer son application, il doit s'authentifier auprés du système en saisissant son identifiant et son mot de passe associé.

   Le système fait appel à une vérification de remplissage et d'authentification. On distingue alors deux cas :

   – Si tous les champs sont remplis une autre vérification se fera au prés de la base de données afin de tester l'existence du compte selon les champs introduits dans la base, on aperçoit deux cas :

   . Le cas ou les paramètres sont corrects, il l'on résulte un accès directe vers la page d'accueil du visiteur ;

   . Le cas opposé où les paramètres ne figurent pas dans la base de données. Et là, la page d'authentification sera réaffichée pour une réintroduction des paramètres. Aprés trois tentatives , un message sera affiché indiquant au visiteur que son mot de passe est oubliée et lui donnnat la possibilité de le récuperer aprés avoir répondu correctement à la question secrète.

   – Sinon un message d'erreur du manque de remplissage sera affiché (Figure 4.5).

   L'administrateur à son tour doit s'authentifier (en raison de sécurité) afin d'avoir les prévilèges qui lui ont été attribués : manipulation de la base de données (ajout, suppression, etc.). le principe d'authentification étant le même que celui du visiteur (Figure 4.6).

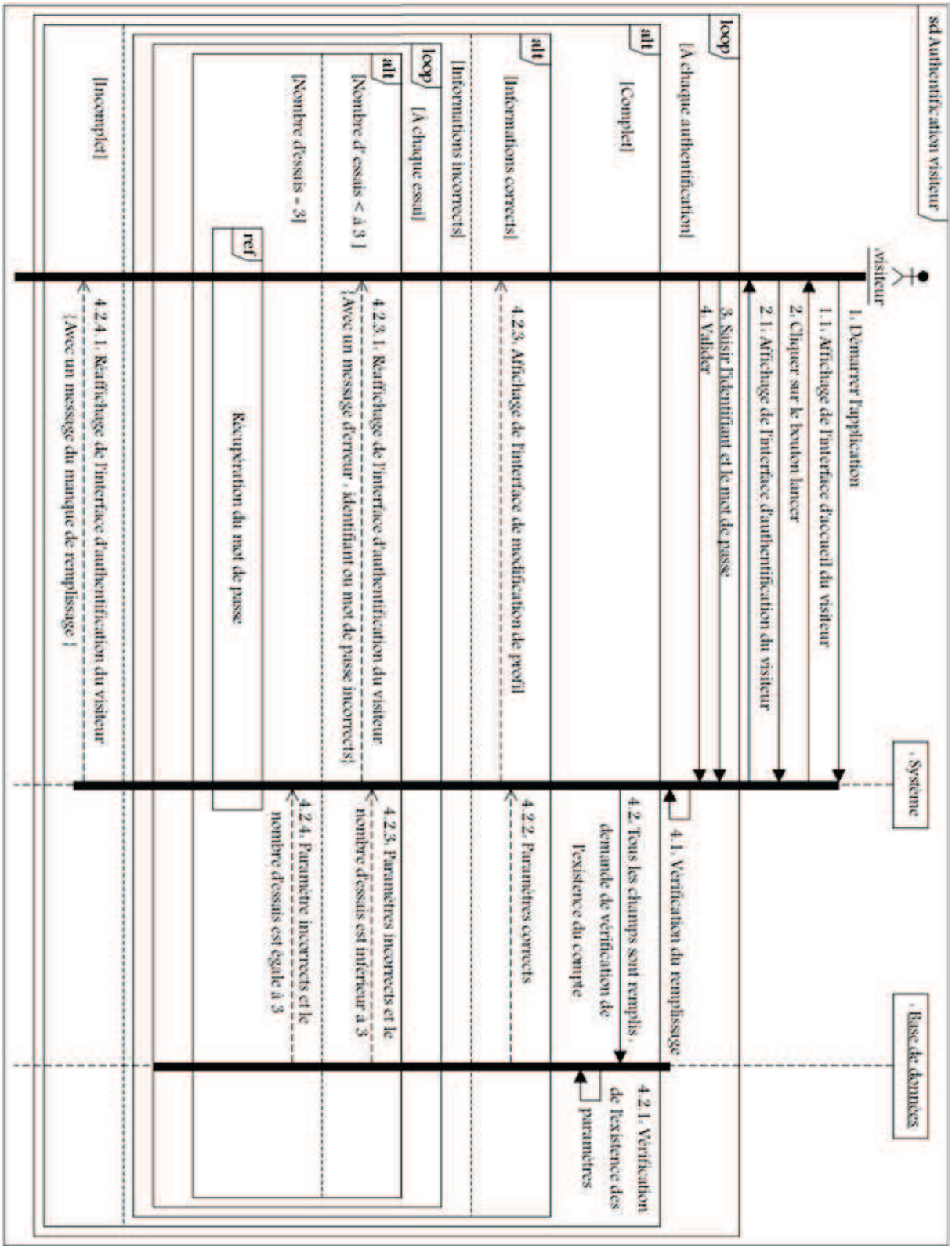


FIGURE 4.5 – Diagramme de séquences du cas d'utilisation "Authentification visiteur".

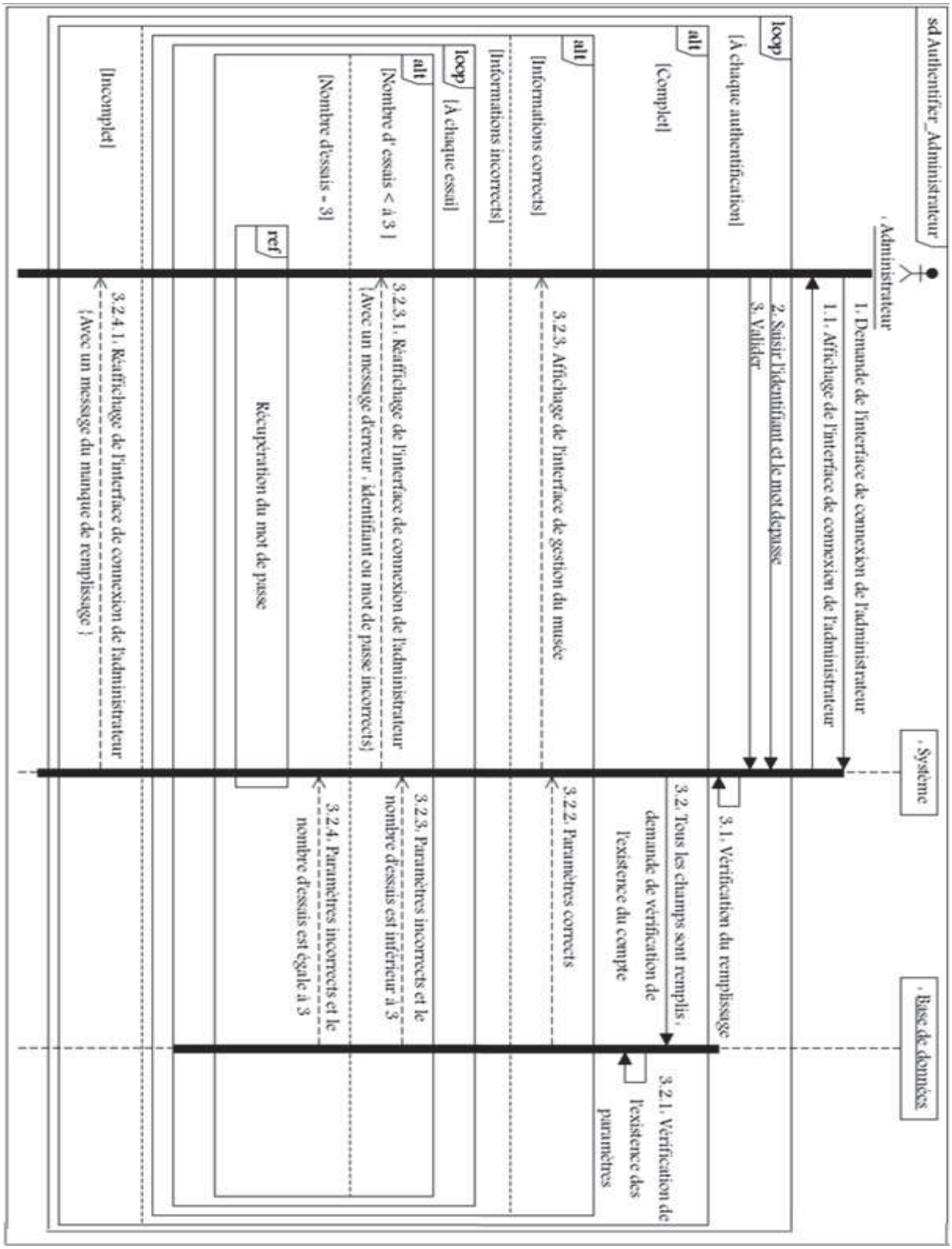


FIGURE 4.6 – Diagramme de séquences du cas d'utilisation "Authentification administrateur".

2. **Diagramme de séquences pour les cas d'utilisation "Récupération du mot de passe"**

   Une possibilité de récupérer le mot de passe est offerte aux administrateurs et aux visiteurs, cette récupération se fait grâce à la question secrète posée lors de la création d'un compte qu'il en soit un compte administrateur ou visiteur. Nous présenterons ci-dessous le diagramme de séquence associé à la récupération du mot de passe coté administrateur. Le principe étant le même pour le visiteur.(Figure 4.7)

3. **Diagrammes de séquences pour les cas d'utilisation "ajouter"**

   Pour ajouter des objets (oeuvres et services) dans la base de données, la procédure est simple. L'administrateur n'a qu'à remplir tous les attributs associés à l'objet puis cliquer sur ajouter. Par exemple si l'on veut ajouter une oeuvre à la base de données, l'administrateur accède à l'interface de connexion puis s'authentifie. Quand l'interface d'administration lui sera affichée, il choisit l'onglet "oeuvre" puis il clique sur le bouton ajouter. À ce moment-là une interface d'ajout d'une oeuvre s'affiche contenant le formulaire que l'administrateur doit remplir avant de presser le bouton valider. Le système vérifie que tous les champs sont remplis puis envoie une requête d'ajout a la base de données (table oeuvres, auteurs et cellules) et un message de confirmation de l'ajout sera affiché. Dans le cas où il y a un manque de remplissage un message d'erreur est prévu (Figure 4.8).

4. **Diagrammes de séquences pour les cas d'utilisation "supprimer"**

   Pour supprimer un objet (oeuvres, services et comptes visiteurs) à partir de la page de gestion. L'administrateur doit aussi s'authentifier, puis dans la page administration, il choisit l'onglet approprié à l'objet. Ensuite il sélectionne l'objet à supprimer et enfin le supprime en appuyant sur le bouton **supprimer**. Nous présenterons ci-dessous les diagrammes de séquence pour la suppression (Figure 4.10).

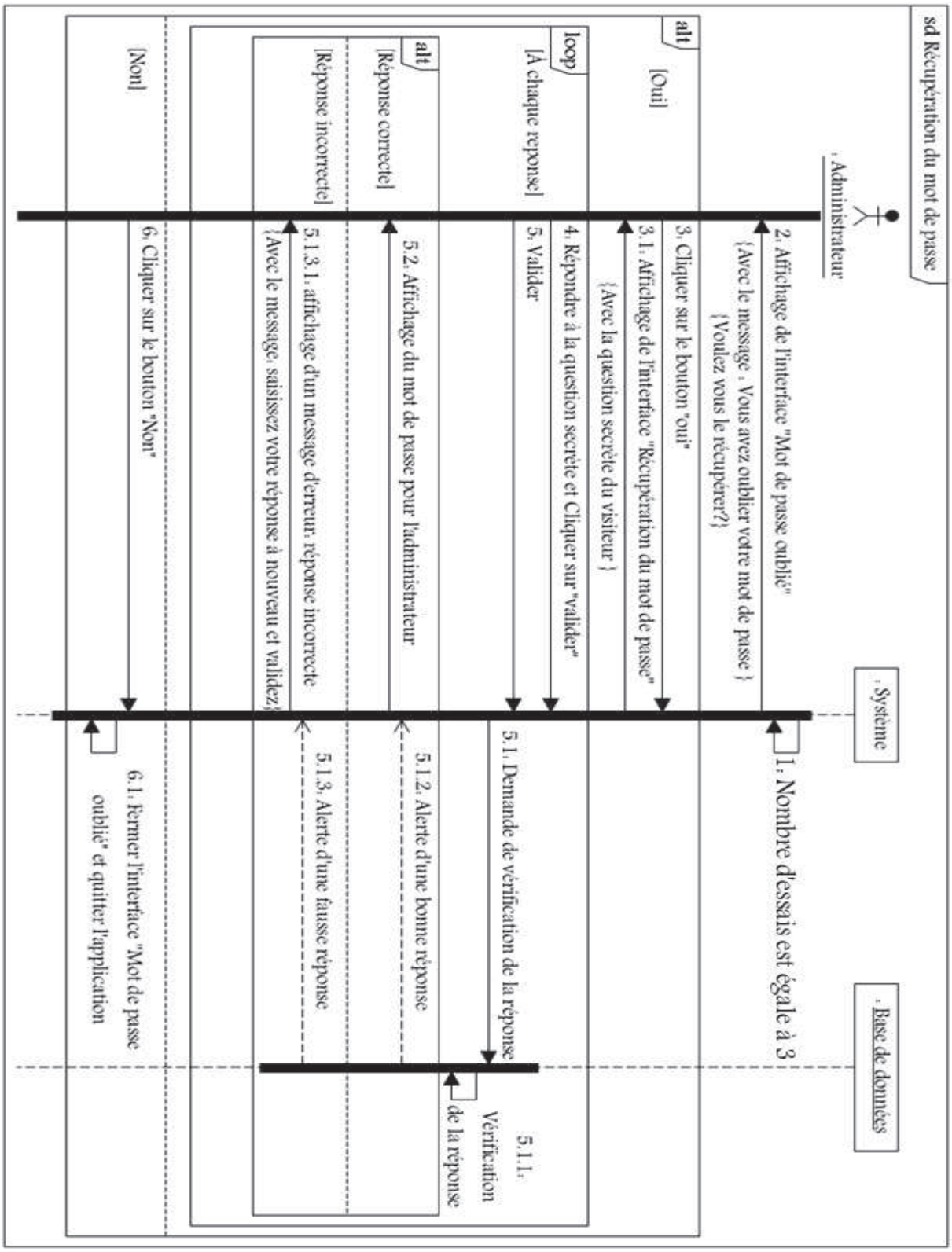


FIGURE 4.7 – Diagramme de séquence du cas d'utilisation " Récupération du mot de passe"

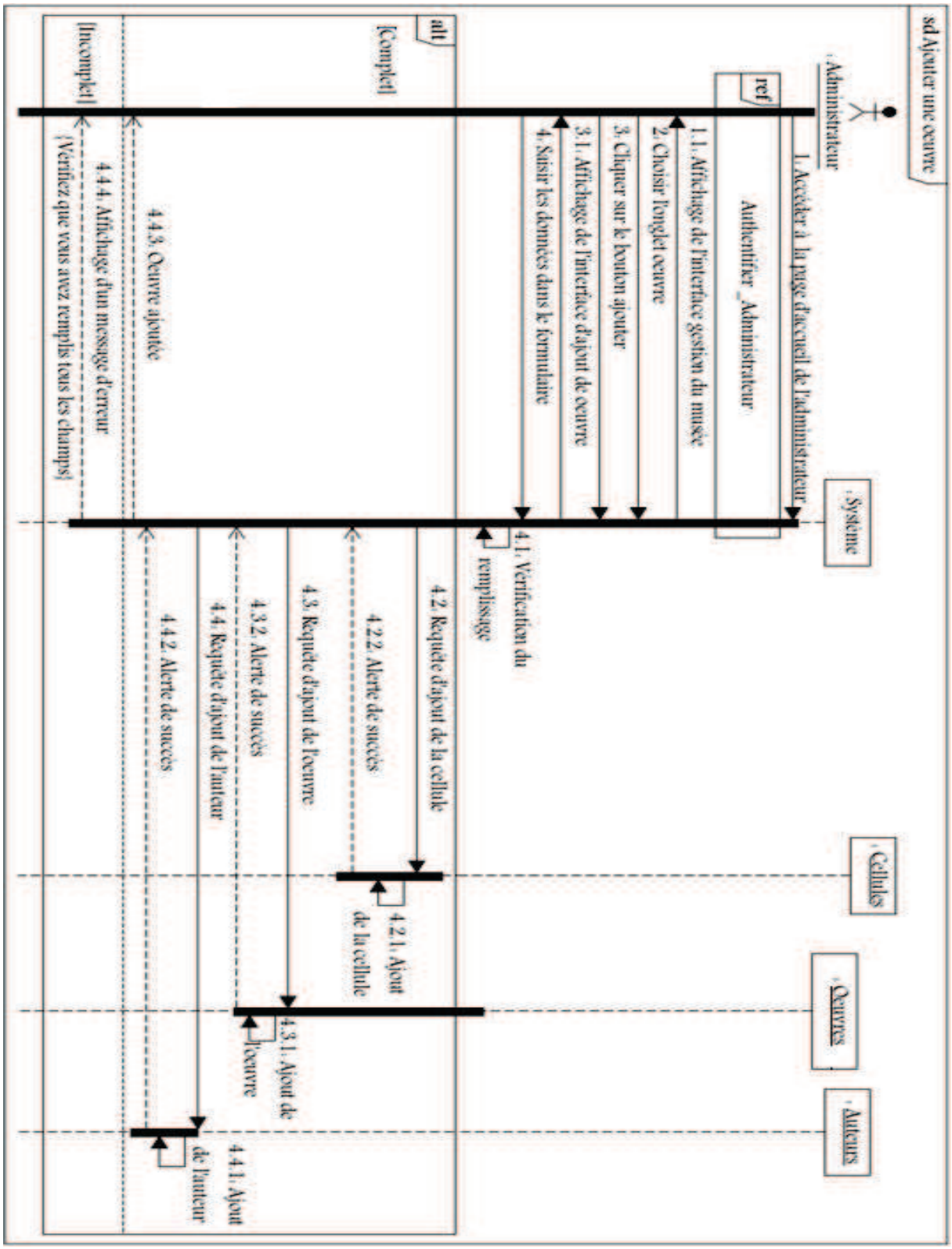


FIGURE 4.8 – Diagramme de séquences du cas d'utilisation "ajouter une oeuvre".

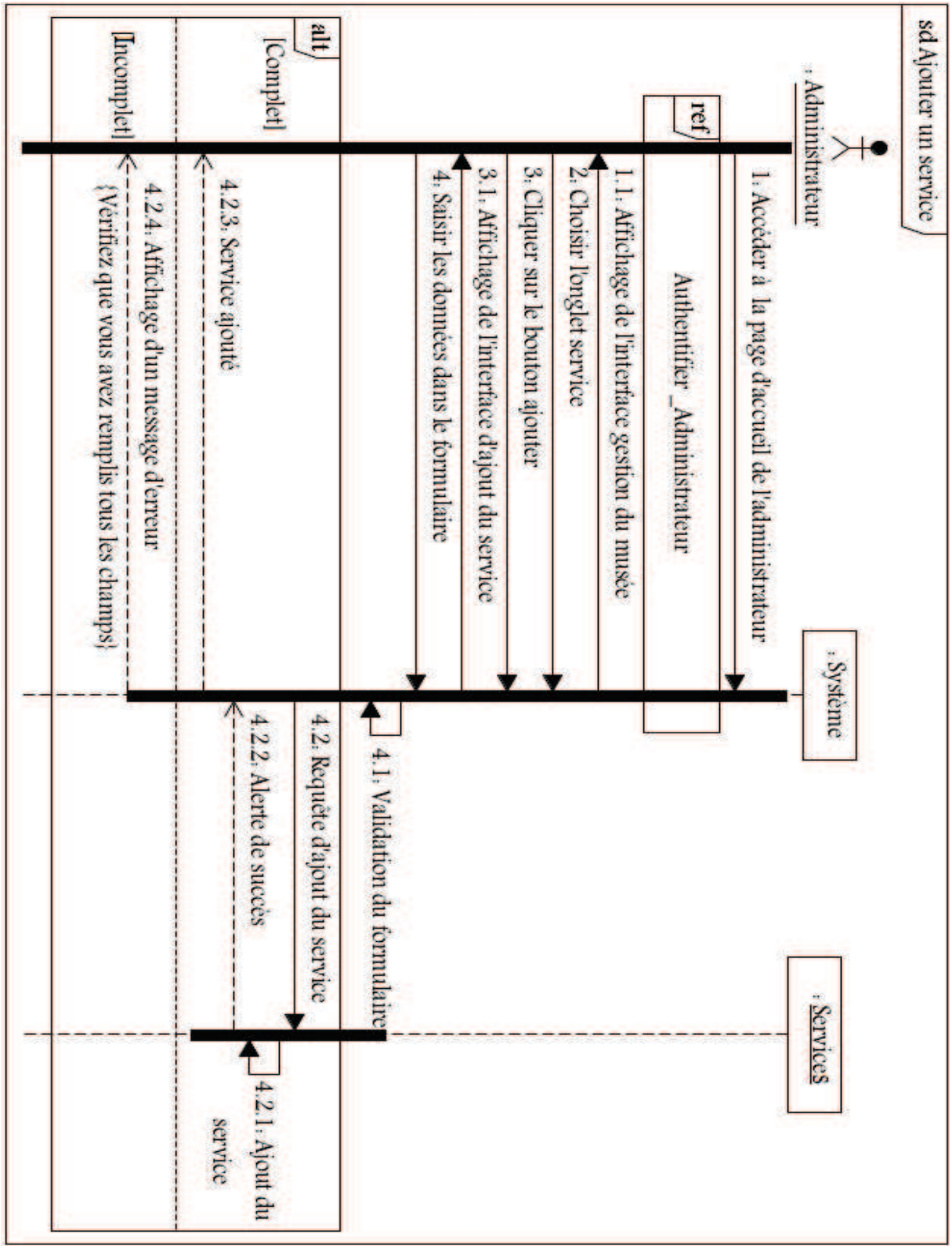


FIGURE 4.9 – Diagramme de séquence du cas d'utilisation "ajouter un service".

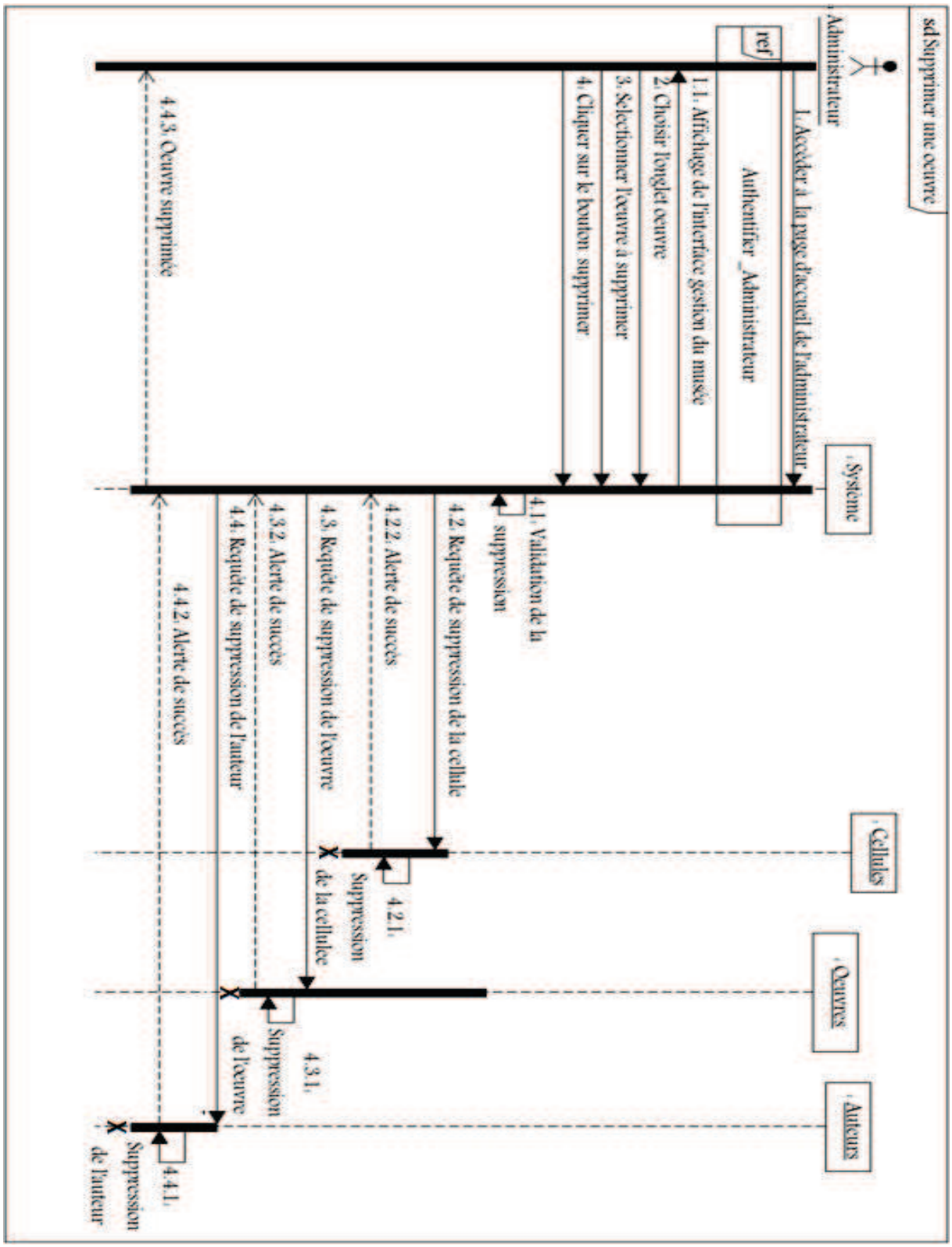


FIGURE 4.10 – Diagramme de séquence du cas d'utilisation "Supprimer une oeuvre".

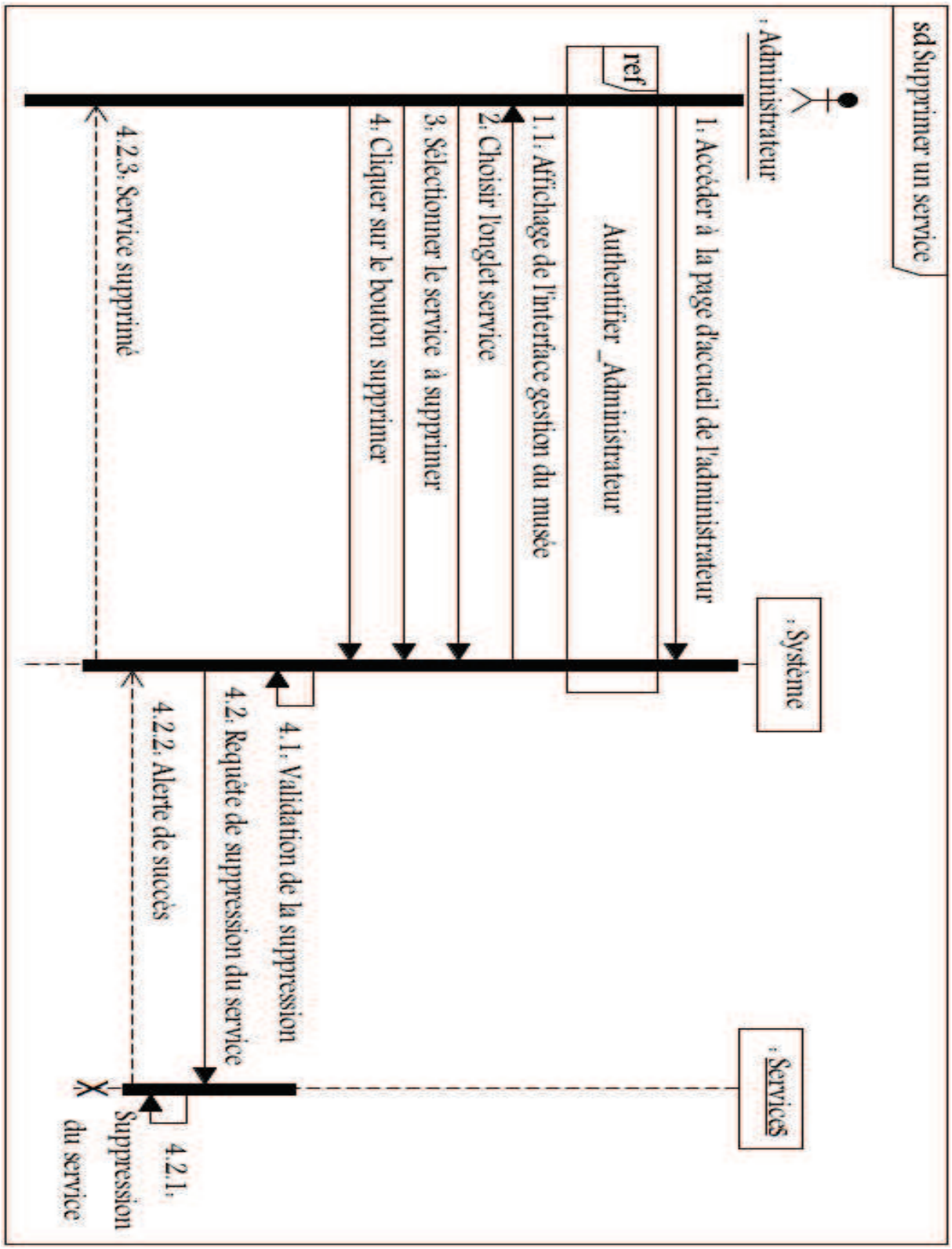


FIGURE 4.11 – Diagramme de séquence du cas d'utilisation "Supprimer un service".

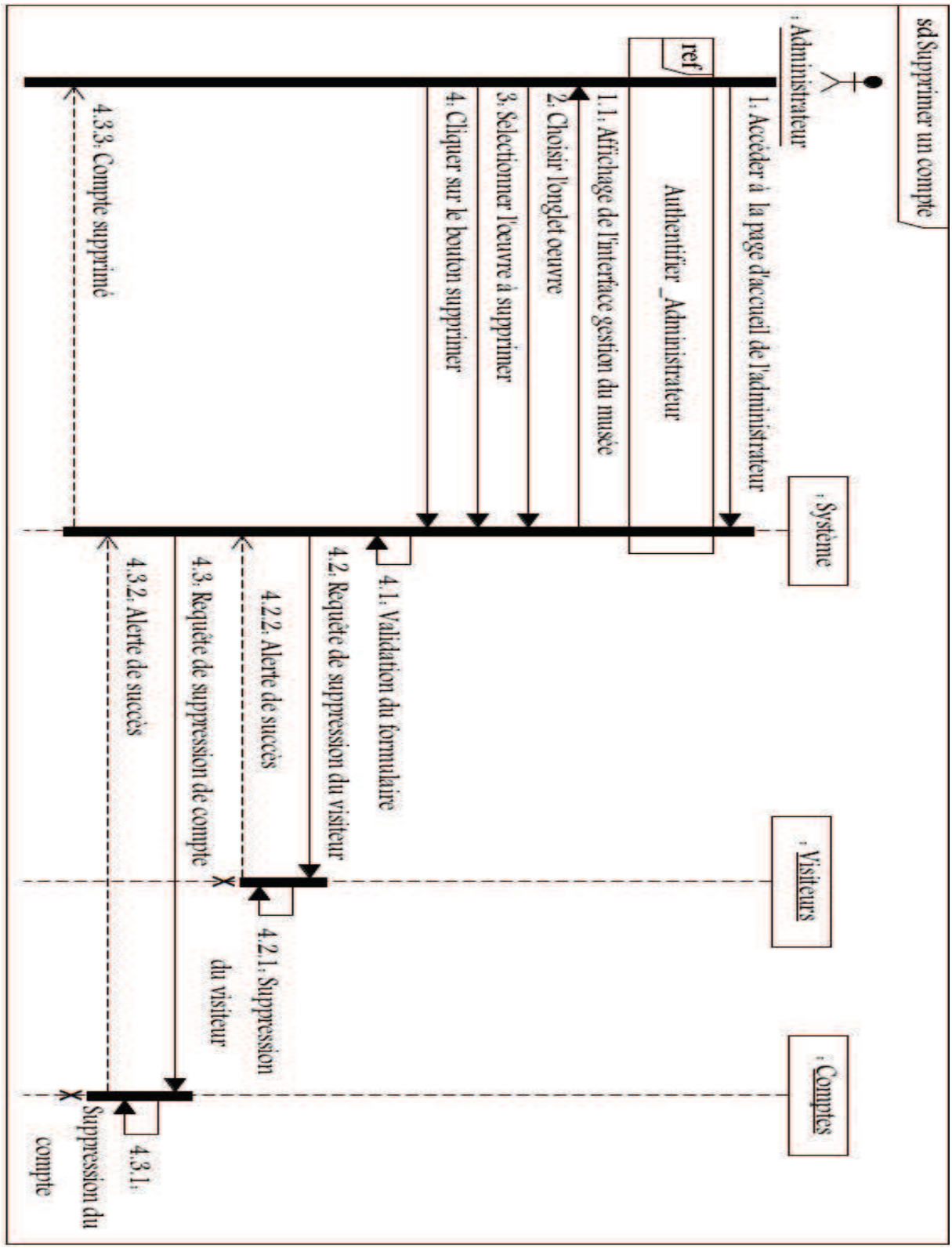


FIGURE 4.12 – Diagramme de séquence du cas d'utilisation "Supprimer un compte".

5. **Diagrammes de séquences pour les cas d'utilisation "Modifier"**

Pour modifier une oeuvre ou un service, l'administrateur doit d'abord s'authentifier. Une fois l'interface d'administration est affichée, il choisit l'onglet en question puis sélectionne l'objet à modifier et clique sur le bouton **modifier**. Une interface de modification lui sera affichée, contenant les informations de l'objet choisit. Il ne reste alors qu'à porter les modifications souhaitées et valider en appuyant sur le bouton **sauvegarder**. Á ce niveau-là, une requête de modification sera transmise à la base de données (les tables appropriées) afin de tenir compte des changements et un message de confirmation s'affichera (Figure 4.13).

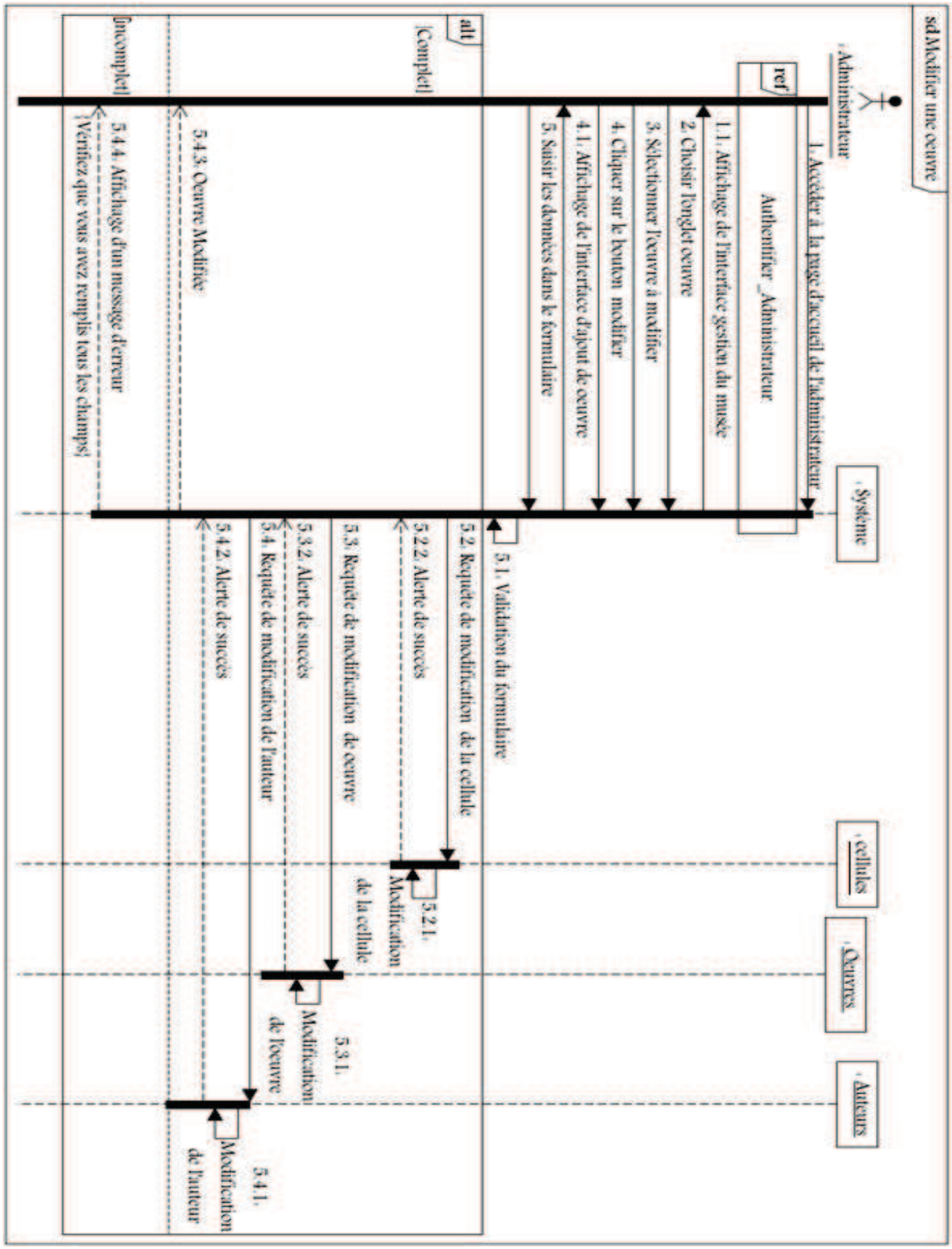


FIGURE 4.13 – Diagramme de séquence du cas d'utilisation "Modifier une oeuvre".

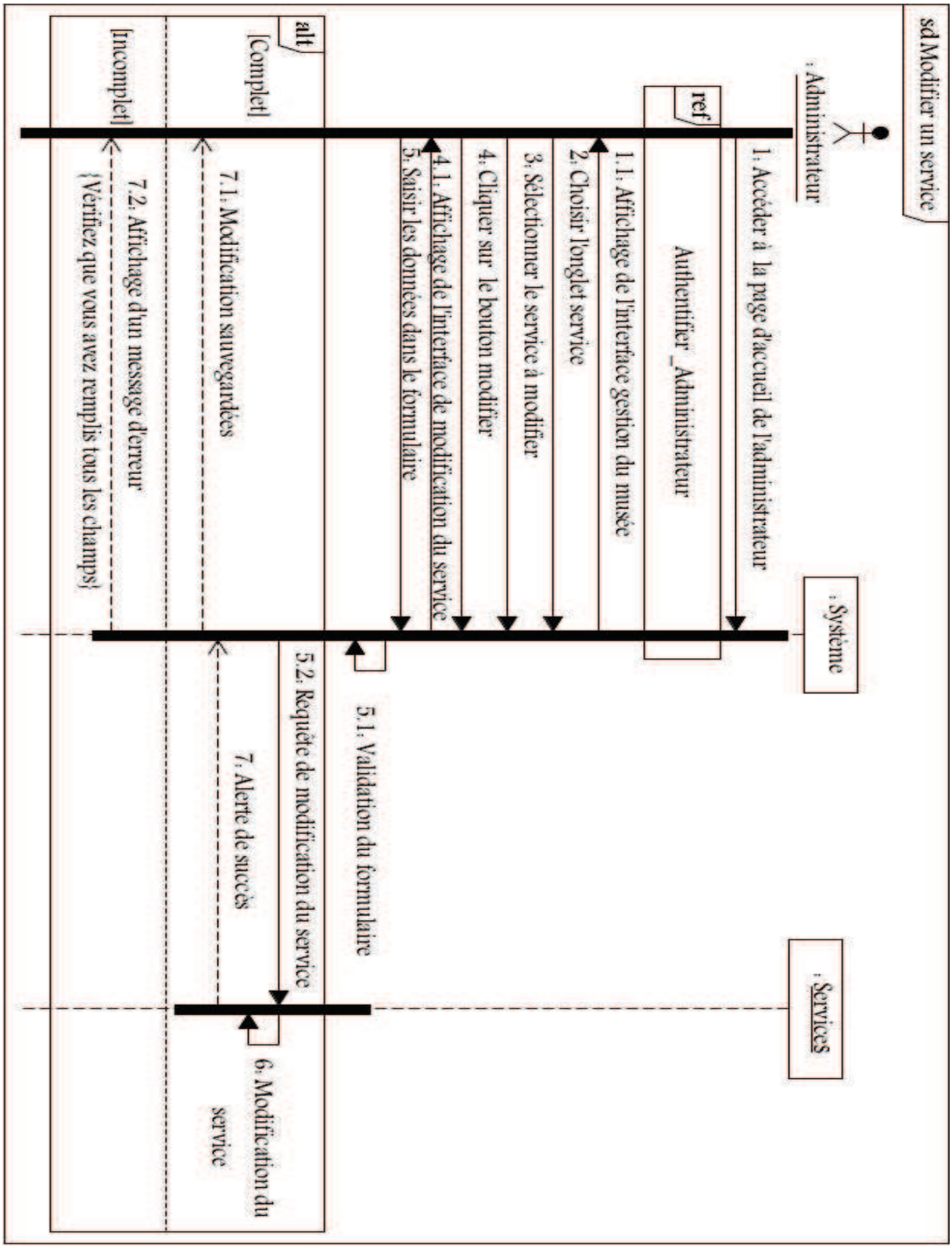


FIGURE 4.14 – Diagramme de séquence du cas d'utilisation "Modifier un service".

6. **Diagrammes de séquences pour les cas d'utilisation "inscription"**

L'inscription est une étape indispensable pour le visiteur. En effet, chaque visiteur doit avoir son propre compte pour utiliser l'application musée et pour pouvoir l'identifier à l'intérieur du musée. Pour ce faire le visiteur doit remplir un formulaire d'inscription contenant les champs : nom, prénom, date de naissance, etc.

Le système inspecte le remplissage des champs :

- Il affiche un message d'erreur dans le cas où l'un/tous des champs est/sont à vide(s)
- Il demande la vérification de l'existence de l'identifiant dans la base de données. Deux cas de figure s'apparaissent :
  - Le cas où l'identifiant est disponible, le système envoie une requête d'ajout du compte à la base de données. suite à l'ajout, une interface d'accueil du visiteur s'affichera.
  - Le cas contraire (identifiant déjà utilisé), un message d'erreur préviendra le visiteur.(Figure 4.15).

L'administrateur à son tour a la possibilité de créer un compte afin de pouvoir gérer la partie administration du musée. Le principe de l'incription étant le méme que celui du visiteur (Figure 4.16).

7. **Diagramme de séquence pour les cas d'utilisation "mise à jour du compte"**

Un visiteur a également la possibilité de modifier son compte afin de mettre à jour son profile. Cette procédure de gestion du compte s'illustre dans la figure 4.17.

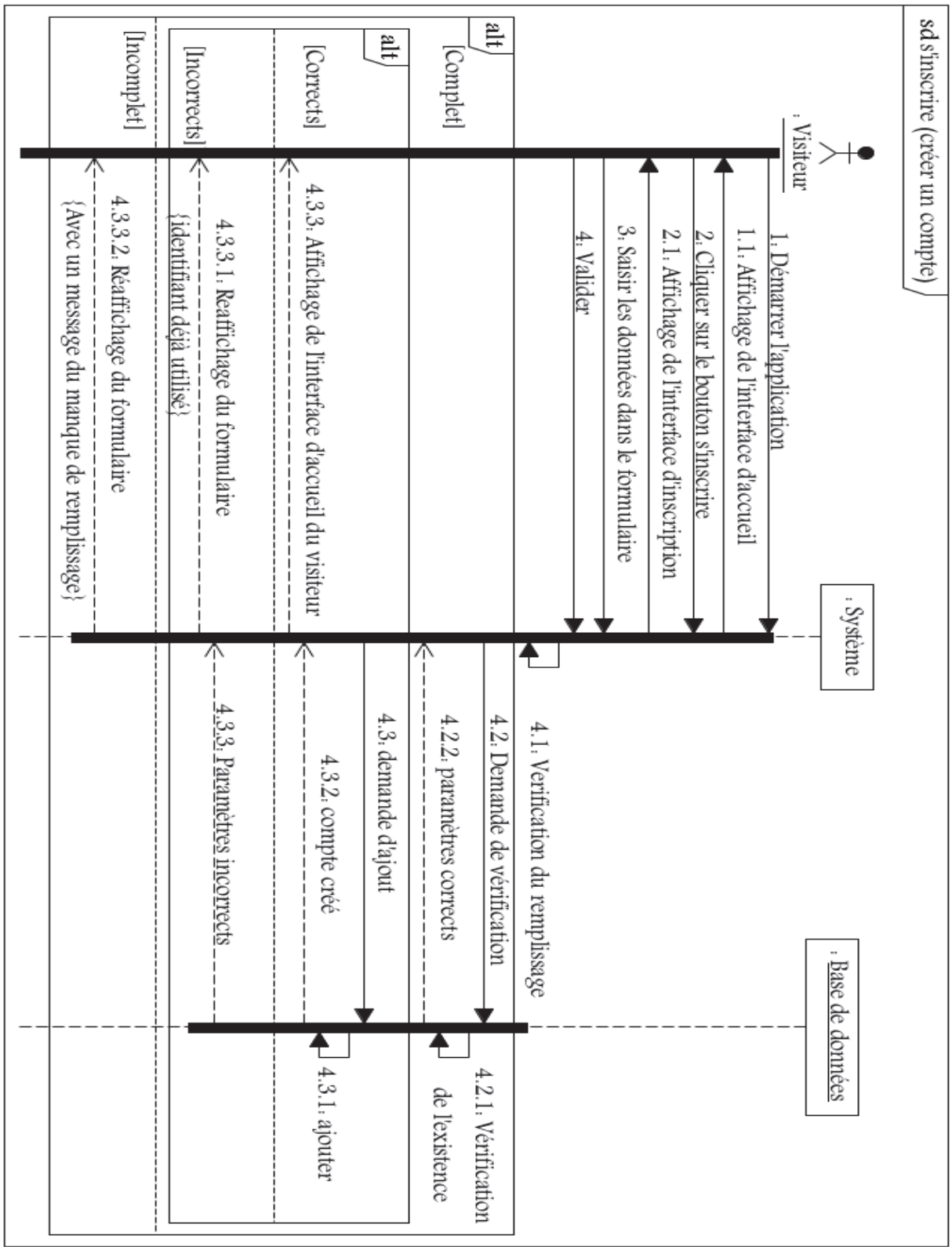


FIGURE 4.15 – Diagramme de séquence du cas d'utilisation "s'inscrire"(créer un compte visiteur).

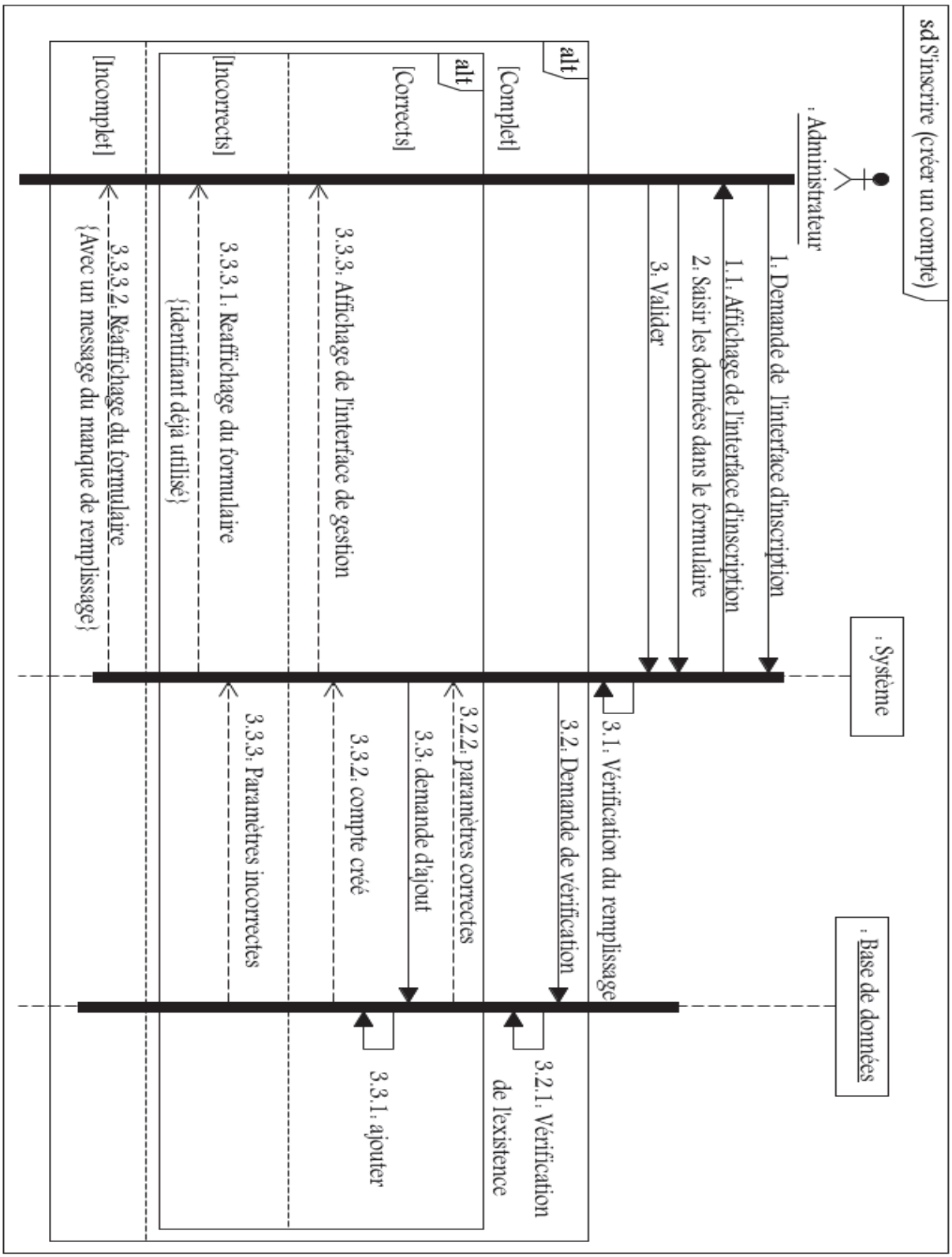


FIGURE 4.16 – Diagramme de séquence du cas d'utilisation "s'inscrire" (créer un compte) administrateur.

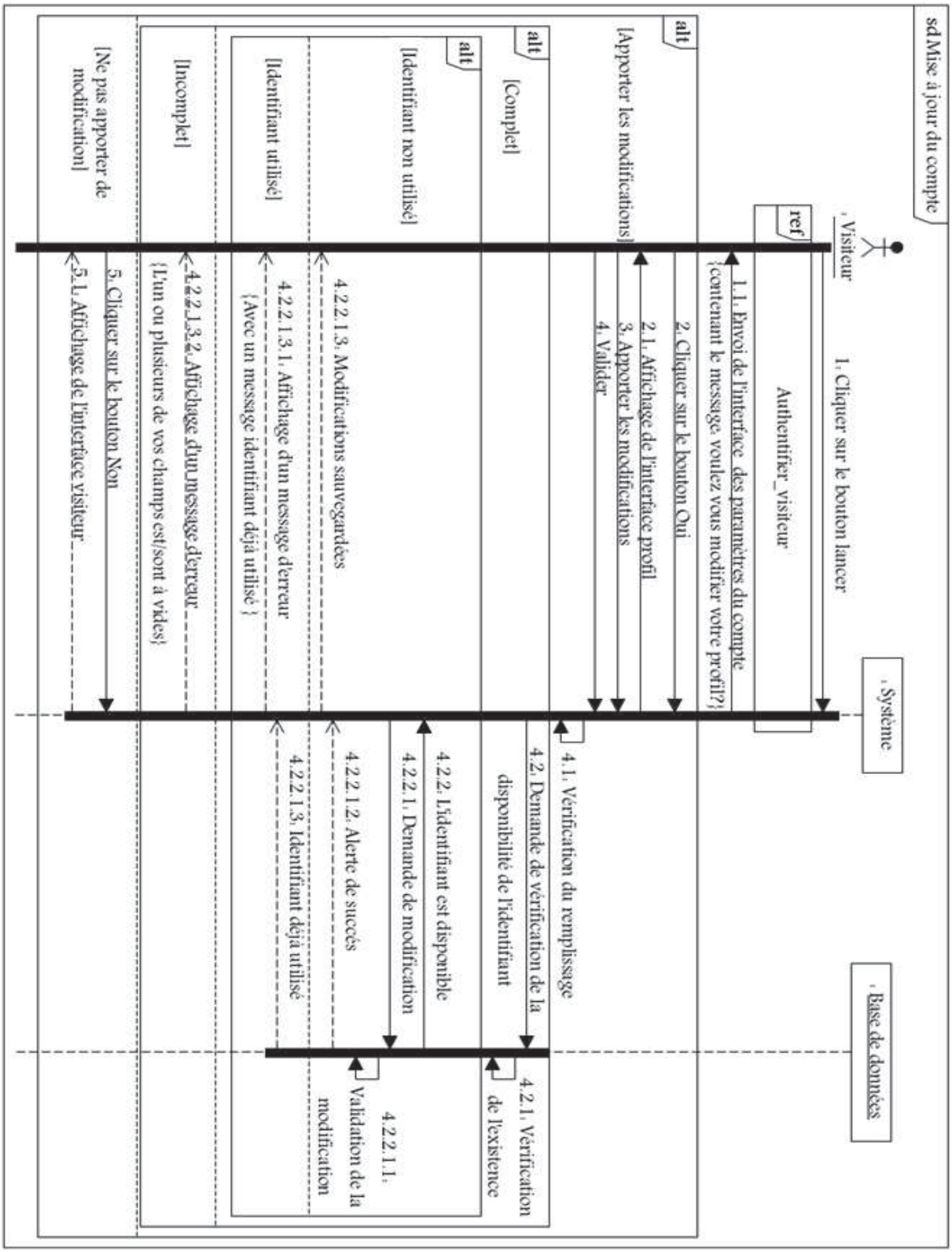


FIGURE 4.17 – Diagramme de séquence du cas d'utilisation "Mise à jour du compte".

4.3.2.3 Diagrammes d'activités

Le déclenchement de chaque service nécessite le lancement des serveurs (serveur de base de données, serveur de contexte statique, serveurs de contexte dynamique et serveur hébergeur de services) et l'application du visiteur.

Pour le service de description, le principe consiste à collecter les positions du visiteur au sein du musée grâce à la localisation fournit par l'application capteur. En effet, quand le visiteur lance l'application une alerte automatique se déclenchera vers l'application capteur de location et ça en utilisant l'identifiant de ce dernier. L'application capteur transmet ces positions au serveur de contexte dynamique, celui-ci vérifie est ce que la position P1 renvoyée à l'instant T (s) est égale à la position P2 renvoyée à l'instant T+5 (s) puis teste si cette dernière est l'une des positions associées aux oeuvres. Si c'est oui, le serveur hébergeur de service sera interrogé et le service de description sera lancé. Ainsi le visiteur va recevoir une description de l'oeuvre (Figure 4.18).

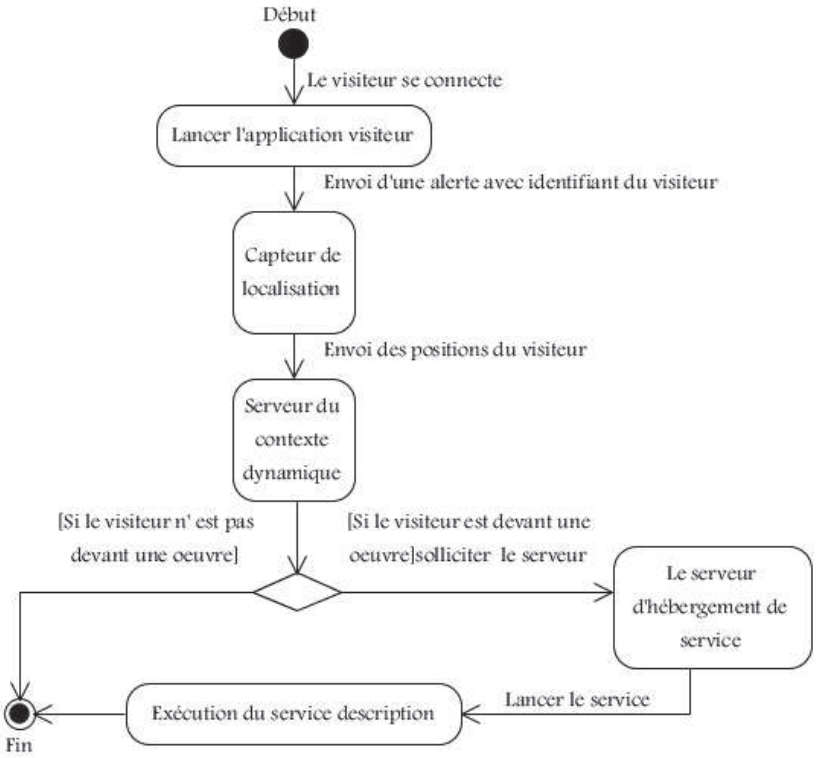


FIGURE 4.18 – Diagramme d'activité "service description".

Pour le service consigne, le principe consiste à collecter le profil du visiteur. Quand ce denier lance l'application, une alerte automatique se déclenchera vers le serveur de contexte statique. Celui-ci vérifie si le visiteur se rend dans le musée pour la première fois (nombre de visite est égale à 1). Si c'est oui, le serveur hébergeur de service sera interrogé et le service consigne sera lancé (Figure 4.19).

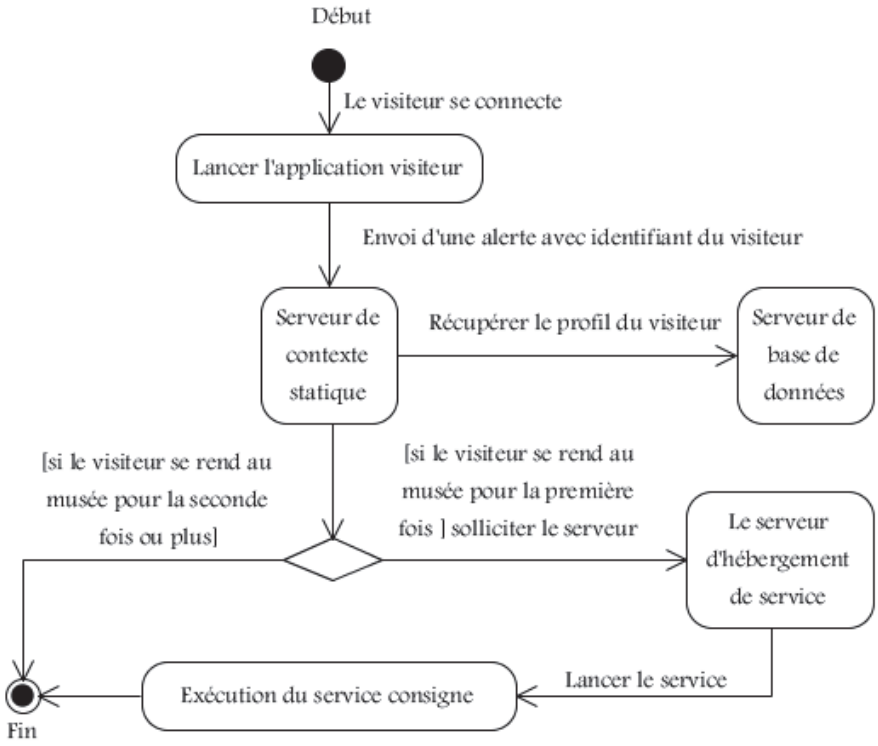


FIGURE 4.19 – Diagramme d'activité "service consigne".

4.3.2.4 Architecture logique

Afin de réaliser la conception logique, UML propose le diagramme de classes. Ce dernier montre la structure de base statique : ce sont les classes et les relations qui les relient [12].

1. **Règles de gestion**

   Avant de présenter le diagramme de classe, nous indiquerons les différentes règles de gestion associées à notre application :

R1 : Un administrateur peut maintenir (ajouter, modifier ou supprimer) une ou plusieurs oeuvres.

R1` : une oeuvre est maintenue par un ou plusieurs administrateurs.

R2 : un administrateur peut gérer (ajouter, modifier ou supprimer) un ou plusieurs services.

R2` : un service est gérée par un ou plusieurs administrateurs.

R3 : un visiteur peut créer et mettre à jour (modifier) un et un seul compte.

R3` : un compte X est créé et mis à jour par un et un seul visiteur.

R4 : un administrateur peut avoir un et un seul compte.

R4` : un compte Y appartient à un et un seul administrateur.

R5 : un visiteur utilise un à plusieurs services.

R5` : un service est utilisé par un ou plusieurs visiteurs.

R6 : une oeuvre est exposée dans une et une seule salle.

R6` : dans une salle sont exposées de zéro à plusieurs oeuvres.

R7 : à chaque oeuvre est associé un et un seul auteur.

R7` : un auteur peur être associé à zéro ou plusieurs oeuvres.

R8 : un visiteur se rend dans un à plusieurs étages.

R8` : un étage est visité par zéro à plusieurs visiteurs.

R9 : un étage est composé de un à plusieurs départements.

R9` : un département se trouve dans un et un seul étage.

R10 : un département est composé de un à plusieurs salles.

R10` : une salle se trouve dans un et un seul département.

R11 : une cellule contient une et une seule oeuvre.

R11` : une oeuvre est exposée dans une et une seule cellule.

R12 : une salle contient une à plusieurs cellules.

R12` : une cellule se trouve dans une et une seule salle.

2. **Dictionnaire de données**

Code de donnée	Désignation	Type	Taille	Observation
ADMINISTRATEURS				
numero-adm	Numéro de l'administrateur	INT	11	
nom_adm	Nom de l'administrateur	CHAR	40	
prenom_adm	Prénom de l'administrateur	CHAR	40	
AUTEURS				
numero_aut	Numéro de l'auteur	INT	11	
nom_aut	Nom de l'auteur	CHAR	40	
prenom_aut	Prénom de l'auteur	CHAR	40	
biographie	Biographie de l'auteur	TEXT		
CELLULES				
numero_cellule	Numéro de la cellule	INT	11	
x	Position selon l'axe des abscisses (x)	DOUBLE		
y	Position selon l'axe des coordonnées (y)	DOUBLE		
COMPTES				
numero_compte	Numéro du compte	INT	11	
id	Identifiant	VARCHAR	40	
mdp	Mot de passe	VARCHAR	40	
ques	Question	CHAR	40	
rep	Réponse	VARCHAR	40	
type_compte	Type du compte	CHAR	10	
last_connection	Date de la dernière connexion	DATE		
DEPARTEMENTS				
numero_dep	Numéro du département	INT	11	
nom_dep	Nom du département	CHAR	40	

OEUVRES				
numero_oeuvre	Numéro de l'œuvre	INT	11	
nom_ouvre	Nom de l'œuvre	CHAR	40	
categorie_oeuvre	Catégorie de l'œuvre	CHAR	15	
description_oeuvre	Description de l'œuvre	TEXT		
SALLES				
numero_salle	Numéro de la salle	INT	11	
Nom_salle	Nom de la salle	CHAR	40	
SERVICES				
numero_serv	Numéro du service	INT	11	
categorie_serv	Catégorie du service	CHAR	15	
titre_serv	Titre de service	CHAR	40	
description_serv	Description du service	TEXT		
date_ajout	Date d'ajout	DATE		AAAA-MM-JJ
VISITEURS				
numero_vs	Numéro du visiteur	INT	11	
nom_vs	Nom du visiteur	CHAR	40	
prenom_vs	Prénom du visiteur	CHAR	40	
date_nais	Date de naissance	DATE		AAAA-MM-JJ
ville_resid	Ville de résidence	CHAR	20	
profession	Profession du visiteur	CHAR	40	
centre_interet	Centre d'intérêt du visiteur	CHAR	40	
etat_sante	Etat de santé du visiteur	CHAR	20	
nbr_visites	Nombre des visites	INT	11	

TABLE 4.1 – Dictionnaire de données.

3. **La représentation graphique du diagramme de classes**

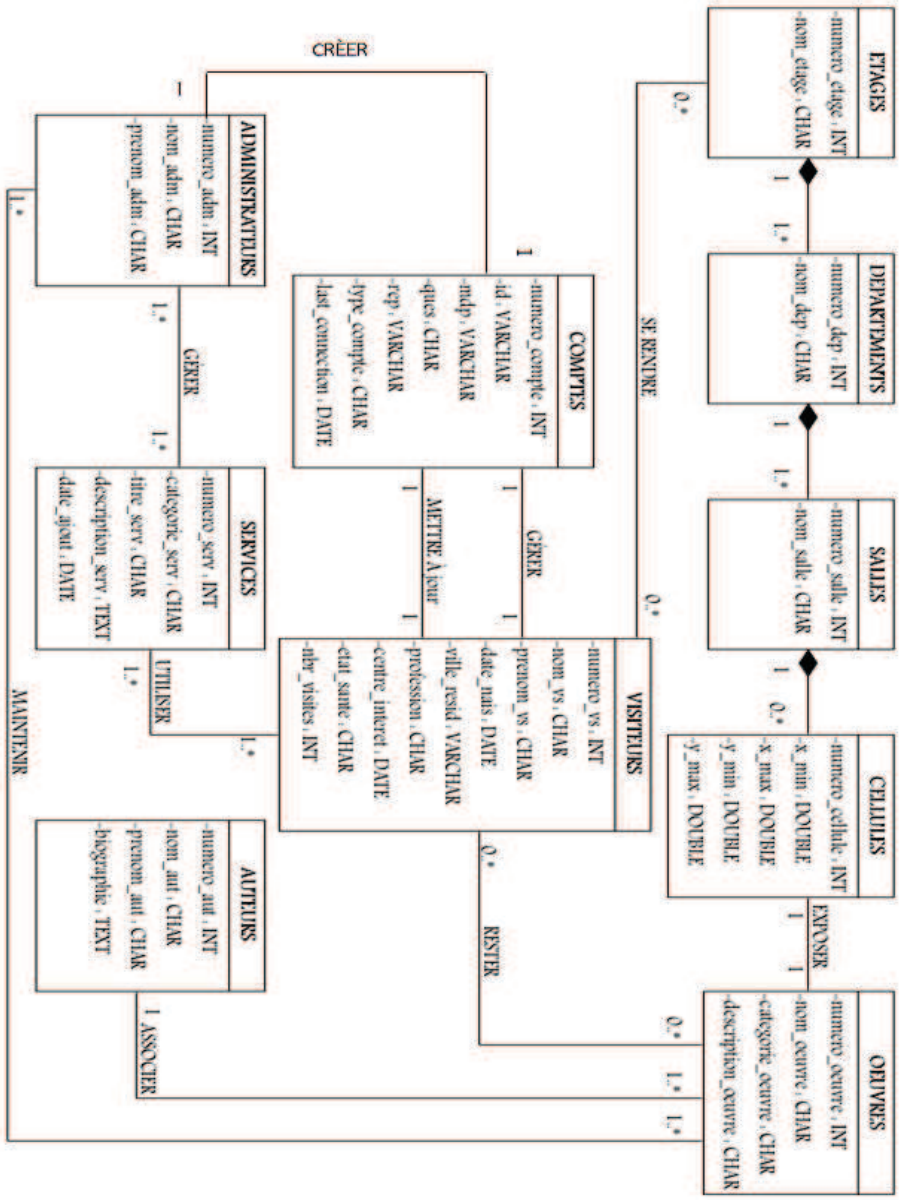


FIGURE 4.20 – Diagramme de classes.

4. **Passage au modèle relationnel**

Le modèle relationnel est une manière de modéliser les informations contenues dans une base de données et qui repose sur des principes mathématiques définis par l'informaticien britannique Edgar F. Codd [82]. Il est souvent considéré comme le plus simple et le plus élégant des modèles.

En 2010 le modèle de données relationnel est utilisé dans la grande majorité des bases de données [9, 10].

La traduction du diagramme de classes en modèle relationnel consiste à appliquer les règles de passage suivantes :

– La traduction des classes en tables :

. Les attributs deviennent des colonnes ;

. L'identifiant naturel est transformé en une clé primaire unique et non nulle.

– La traduction des associations en relations :

. Chaque association "plusieurs à plusieurs" est traduite par une table associative où la clé primaire est formée de la concaténation des colonnes de clés étrangères des tables sources ;

. Chaque association "un à plusieurs" disparait, la clé primaire du père va migrer vers le fils ;

. Chaque association où les deux cardinalités valent 1, chacune des deux tables peut devenir indifféremment source ou cible de la relation.

Après le passage du diagramme de classes au modèle relationnel, nous présentons ci-dessous les principales relations :

ADMINISTRATEURS (numero_adm, nom_adm, prenom_adm, numero_compte*).

AUTEURS(numero_aut, nom_aut, prenom_aut, biographie).

CELLULES(numero_cellule,numero_salle*,numero_dep*,numero_etage*, x_min, x_max, y_min, y_max numero_oeuvre*).

COMPTES (numero_compte, id, mdp, ques,rep,type_compte,last_connetion, numero_vs*, numero_adm*).

DEPARTEMENTS (numero_dep, numero_etage*,nom_dep).

MAINTENIR (numero_adm*, numero_oeuvre*).

ETAGES (numero_etage, nom_etage).

GERER (numero_adm*, numero_srev*).

OEUVRES (numero_oeuvre, nom_oeuvre, categorie_oeuvre, description_oeuvre, numero_cellule*, numero_salle*, numero_dep*, numero_etage*, numero_aut*).

RESTER (numero_vs*, numero_oeuvre*).

SALLES (numero_salle, numero_dep*, numero_etage*).

SE RENDRE (numero_vs*, numero_etage*).

SERVICES (numero_serv, categorie_serv, titre_serv, description_serv, date_ajout).

UTILISER (numero_vs*, numero_serv*).

VISITEURS (numero_vs, nom_vs, prenom_pers, date_ nais, ville_resid, profession, centre_interet, etat_sante, nbr_visites, numero_compte*).

5. **Architecture physique**

Nous présentons ci-après, le diagramme de déploiement de notre système. Ce dernier nous permet de représenter l'architecture physique supportant l'exploitation du système. Cette architecture comprend des noeuds correspondant aux supports physiques (serveurs, routeurs, etc.) ainsi que la répartition des artefacts logiciels (bibliothèques, exécutables, etc.) sur ces noeuds. C'est donc un véritable réseau constitué de noeuds et de connexions entre ces noeuds qui modélise cette architecture.

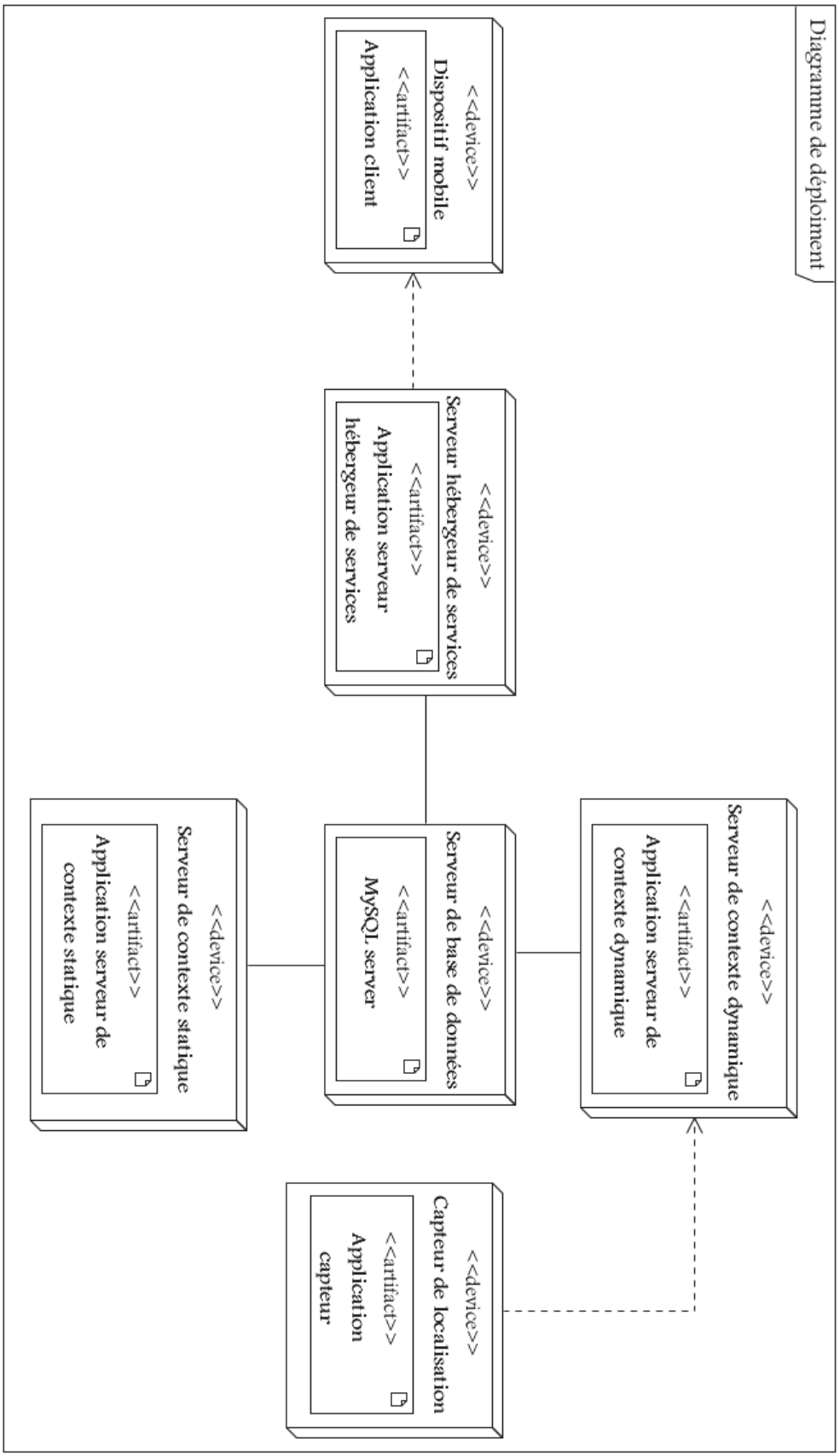


FIGURE 4.21 – Diagramme de déploiement "Configuration matériel du système".

4.4 Conclusion

Tout au long de ce chapitre, nous avons détaillé la conception du système à réaliser selon une approche orientée objet, et ce afin de garantir la fiabilité et l'efficacité de la phase de réalisation. Nous avons, donc, présenté les différents diagrammes UML utilisés, à savoir : le diagramme de classes, le digramme de cas d'utilisation et les diagrammes de séquences représentés avec le paradigme visuel.

Le chapitre suivant sera consacré à la réalisation proprement dite de l'application ; où l'on présentera les outils de développement utilisés et quelques interfaces de cette dernière ainsi qu'une explication sur le fonctionnement de l'application. C'est d'ailleurs l'objet de la phase construction (réalisation) du RUP.

Chapitre

5 Réalisation

5.1 Introduction

La troisième phase du processus RUP est la réalisation. Elle représente l'étape qui suit immédiatement la phase de conception. C'est l'aboutissement final de notre projet et c'est la raison d'être du projet lui-même.

La réalisation de notre application se répartie en quatre (04) sous-applications principales :

- L'application de gestion du musée : c'est l'application qui permettra aux administrateurs du musée de gérer la base de données ;
- L'application client : c'est l'application qui sera installée sur le périphérique mobile du visiteur ;
- L'application serveurs : c'est la partie implémentation des serveurs qui gèrera le contexte des visiteurs et qui se chargera de fournir les services du musée à ces derniers ;
- L'application capteur : c'est un petit programme qui remplacera le capteur matériel.

Dans ce chapitre nous présenterons les outils, langages et environnements de développement retenus pour la réalisation,le fonctionnement de l'application puis nous passerons en revue l'ensemble de ses interfaces.

5.2 Choix des outils et langages de développement

5.2.1 Java

Java fut l'un des outils de programmation idéal à usage général, évolué et orienté objet.

Il intègre les concepts les plus intéressants des technologies informatiques récentes dans une plate-forme de développement riche et homogène [83].

Nous avons choisi l'outil java pour l'ensemble de ses caractéristiques qui ont largement contribué à son énorme succès, à savoir [84] :

- **Java est interprété :** la source est compilée en pseudo code ou byte code puis exécutée par un interpréteur Java "la Java Virtual Machine (JVM)".
- **Java est portable :** il est multiplateformes (ne dépend d'aucune plate-forme). Il est possible d'exécuter des programmes Java sur tous les environnements qui possèdent une JVM.
- **Java est orienté objet :** Chaque fichier source contient la définition d'une ou plusieurs classes qui sont utilisées les unes avec les autres pour former une application.
- **Java est simple :** il est simple d'utilisation.
- **Java est fortement typé :** toutes les variables sont typées et il n'existe pas de conversion automatique qui risquerait une perte de données.
- **Java assure la gestion de la mémoire :** l'allocation de la mémoire pour un objet se fait automatique à sa création et Java récupère automatiquement la mémoire inutilisée grâce au garbage collector (GC) qui restitue les zones de mémoire laissées libres suite à la destruction des objets.

5.2.2 PHPMyAdmin

PHPMyAdmin est un outil d'administration du serveur de base de données MYSQL à travers une interface web. Il permet de[93] :

- Créer ou supprimer des bases de données ;
- Créer, copier, modifier ou supprimer des tables ;
- Ajouter, modifier ou supprimer des champs des tables ;
- Exécuter des commandes SQL ;
- Exporter/Importer des tables et leurs données.

PHPMyAdmin est écris en PHP, ce sont en fait des pages PHP qui permettent l'administration du serveur. Pour le faire fonctionner, il faut, donc, avoir un serveur Web et PHP installé.

5.2.3 MYSQL

MYSQL est un Système de Gestion de Base de Données(SGBD) qui permet d'entreposer des données de manière structurée (Base, tables, champs, enregistrements, etc.). Le noyau de ce système permet d'accéder à l'information entreposée via un langage spécifique qu'on appelle SQL (Structured Query Language) [85].

Les principales instructions SQL de MySQL sont classifiées dans le tableau suivant [86] :

Ordres SQL	Aspect du langage
CREATE – ALTER – DROP - RENAME – TRUNCATE	Définition des données
INSERT – UPDATE – DELETE - LOCK TABLE	Manipulation des données
SELECT	Interrogation des données
GRANT – REVOKE – COMMIT – ROLLBACK – SAVEPOINT – SET TRANSACTION	Contrôle des données

TABLE 5.1 – Les principales instructions SQL de MySQL.

MySQL doit sa popularité à sa rapidité, sa robustesse, la facilité de son utilisation, ses performances et son caractère open source. Il propose un ensemble de fonctionnalités riches et extrêmement utiles.

5.2.4 PHP

PHP (officiellement : Hypertext Preprocessor) est un langage de développement Web créé pour les développeurs web. Ce langage de développement est un langage de script qui est principalement utilisé pour être exécuté par un serveur http, mais il peut fonctionner comme n'importe quel langage interprété en utilisant les scripts et son interpréteur sur un ordinateur. On considère parfois PHP comme une plate-forme plus qu'un simple langage. Sa syntaxe et sa construction ressemblent à celles des langages C++, à la différence que le PHP peut être directement intégré dans un code HTML [87, 88].

Dans notre cas l'utilisation de ce langage se limitera à la création des scripts PHP qui nous permettrons de connecter l'application client à la base de données.

5.2.5 Le serveur Apache

Apache est le serveur le plus répandu sur Internet, il s'appuie sur les protocoles http (HyperText Transfer Protocol) ou HTTPS (la version sécurisée de HTTP). Son rôle est de recevoir les requêtes émises par les navigateurs, de rechercher la page demandée et de la renvoyer. Il s'agit d'une application fonctionnant à la base sur les systèmes d'exploitation de type Unix mais il a désormais été porté sur de nombreux systèmes, tels que Microsoft Windows [90].

Parmi ces caractéristiques [91] :

- Apache est open source ;
- Il dispose d'un code source disponible et modifiable permettant un développement rapide ;
- Il est flexible, grâce à sa structure modulaire. La jonction d'un nouveau module permet d'ajouter des fonctionnalités supplémentaires ;

La flexibilité du serveur combinée à sa stabilité, à ses performances ainsi qu'à la disponibilité du code source, ont fait du logiciel Apache le serveur web le plus répandu sur Internet.

5.3 Choix de l'environnement de développement

5.3.1 Eclipse

L'Eclipse IDE (Integrated Development Environment)est un environnement de développement libre permettant de créer des programmes dans de nombreux langages de programmation (Java, C++, PHP, etc.) [89].

Pour l'application gestion du musée, nous avons, donc, utilisé l'environnement Eclipse IDE pour lequel nous avons ajouté le pluging 'Visual Editor' afin de pouvoir manipuler d'une manière fluide l'aspect graphique de l'application de gestion.

5.3.2 Android SDK

1. **Présentation**

   Android a été développé afin de permettre aux développeurs de bénéficier au maximum de tous ce que doit offrir un appareil mobile. Android est entièrement ouvert, par exemple une application peut utiliser des fonctionnalités principales d'un téléphone comme lancer un appel, envoyer un SMS, envoyer des e-mails, utiliser le GPS ou l'appareil photo intégrée du téléphone. Un développeur peut accéder à toute les applications du téléphone ou de créer des applications plus riches. Android est basé sur le noyau linux qui utilise une machine virtuelle personnalisée qui était conçu pour optimiser la mémoire et les ressources matériels d'un environnement mobile. Android étant open source, il peut être librement adapté pour intégrer de nouvelles technologies dès sa sortie. Android est en constante évolution grâce aux applications innovantes que réalisent les développeurs [92].

2. **Création et utilisation de l'émulateur**

   L'AVD (Android Virtual Device) est utilisé pour avoir une imitation du comportement physique d'un matériel (téléphone) par un logiciel. Nous pouvons ainsi tester notre application sur plusieurs plateformes matérielles sans avoir à acheter les téléphones correspondants.

   Le SDK Android n'inclut aucun appareil virtuel préconfiguré et nous devrons en créer au moins un (Figure 5.1).

   Android cible un très large ensemble de terminaux. Afin de gérer au mieux cette diversité, Google a développé un système permettant d'émuler un large panel de terminaux. Un AVD est, en réalité, un terminal virtuel disposant de plusieurs propriétés qui lui sont propres : taille d'écran, version du système, présence ou non d'une carte SD, etc. (Figure 5.2).

   Lorsque l'émulateur démarre, il utilise un AVD et s'adapte aux différentes propriétés de ce dernier. Lorsque nous débutons le développement sur Android, il peut être nécessaire de créer plusieurs AVD (notamment pour les développeurs ne disposant pas de terminal Android).

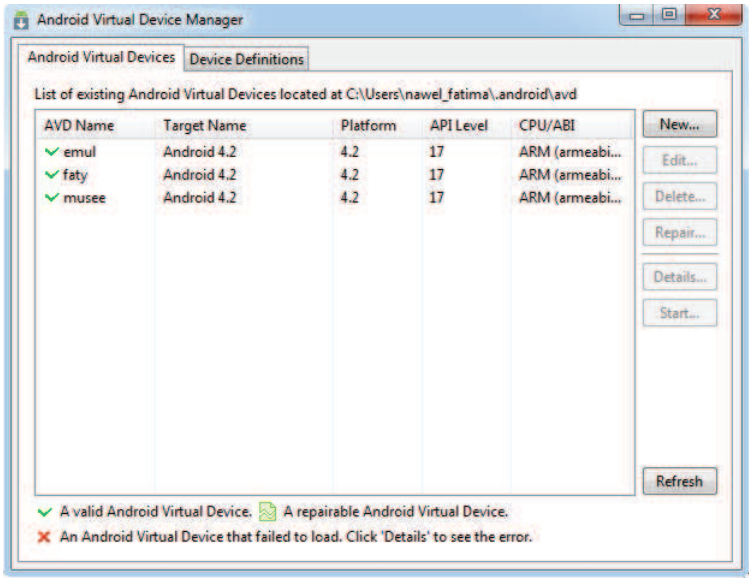


FIGURE 5.1 – Gestionnaire des AVDs.

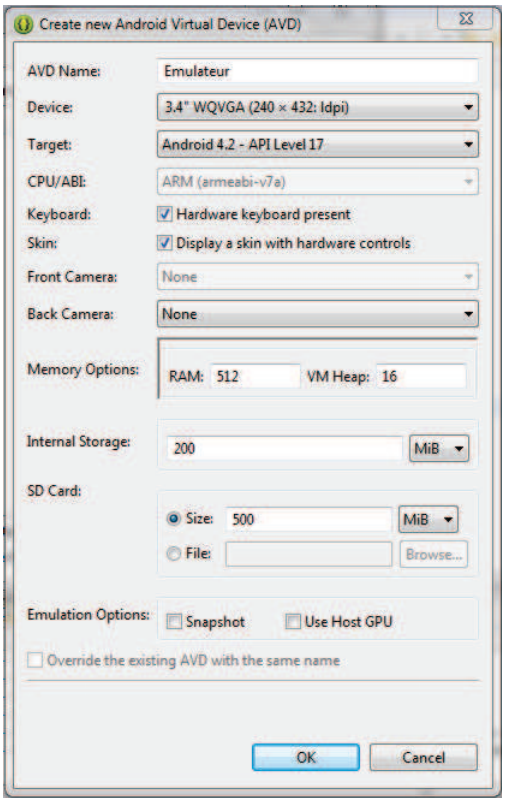


FIGURE 5.2 – Création d'un AVD.

L'émulateur est le parfait outil de test et de débogage des applications. Il est une implémentation de la machine virtuelle Dalvik/footnoteDalvik : c'est la machine virtuelle open-source utilisée dans les appareils Android., faisant de celle-ci une plateforme exécutant les applications Android comme le ferait n'importe quel téléphone Android. Etant découplé de tout matériel, c'est une excellente base de test de notre application (Figure 5.3).

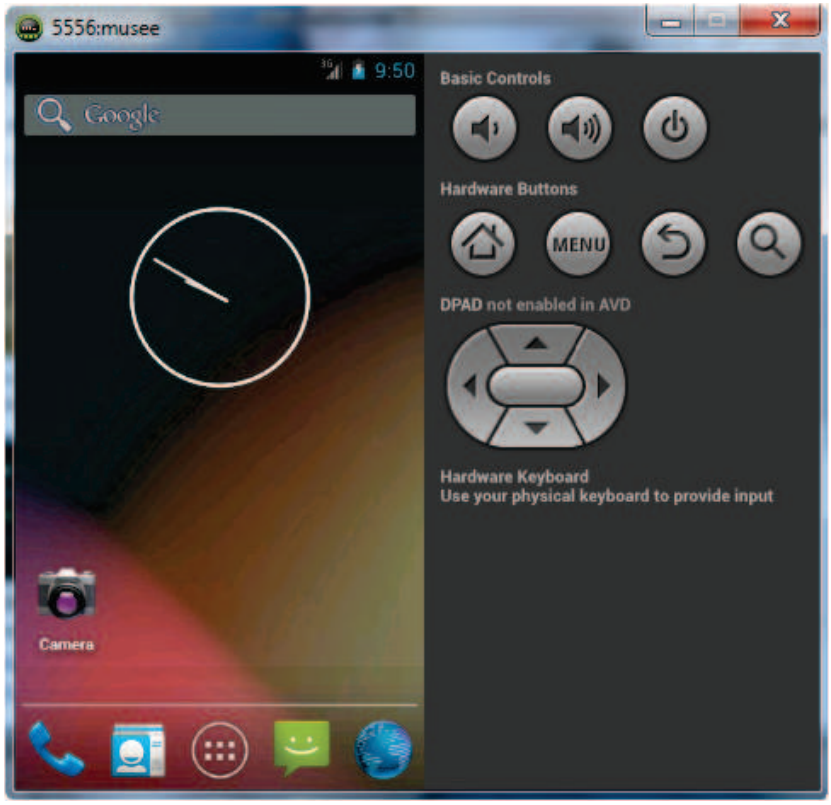


FIGURE 5.3 – L'AVD 'musee'.

5.3.3 EasyPHP

EasyPHP est un paquetage contenant à la fois Apache, PHP et MYSQL. Il permet d'installer automatiquement et facilement une plateforme permettant l'exploitation des applications web qui éventuellement auraient besoin d'un accès à une base de données [87].

Dans notre cas, il ne s'agit pas d'une application web mais d'une application mobile native qui doit être connectée à une base de données via des scripts PHP.

5.4 Création de la base de données

La base de donnée de l'application musée se nomme **"musee"** elle est constituée d'un ensemble de tables qui font références au diagramme de classe. Elle est crée en utilisant l'outil PHPMyadmin et le gestionaire de base de donnée MySQL (Figure 5.4).

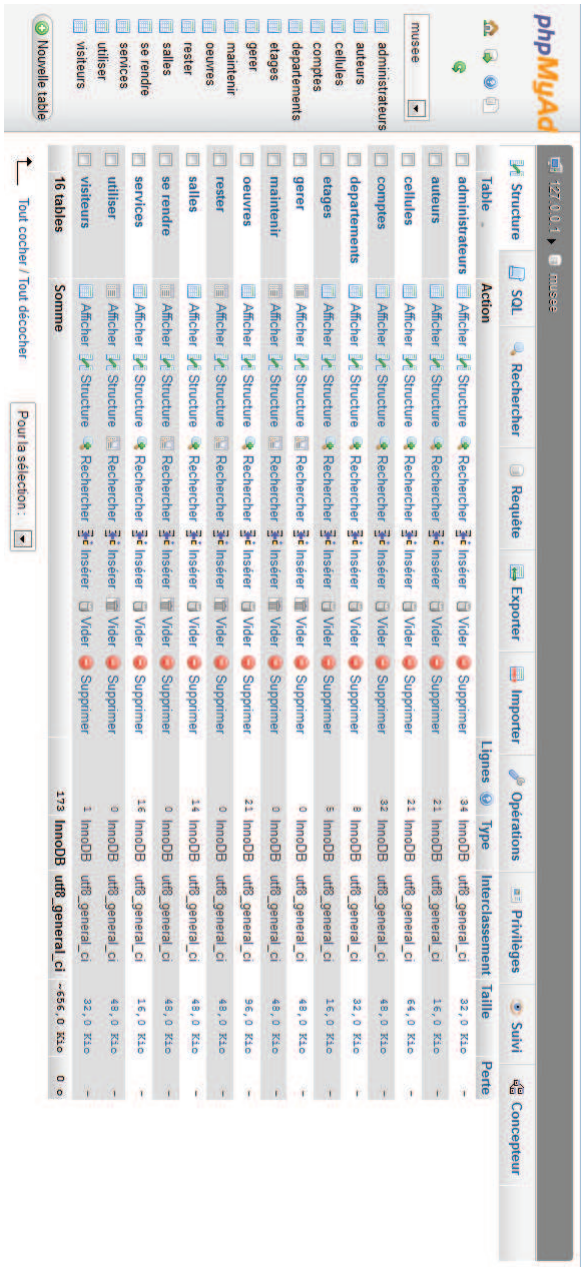


FIGURE 5.4 – Création de la base de données.

5.5 Fonctionnement de l'application

5.5.1 la partie clients-serveurs

Les serveurs représentent la partie transparente du système. C'est à ce niveau que sont implémentés les différents serveurs (serveur de contexte statique, serveur de contexte dynamique et serveur hébergeur de service) sous java et c'est là aussi que les différents tests sont effectués.

Le client représente l'application du visiteur, cette dernière est implémentée sous Android.

La communication entre les serveurs et l'application 'client' se fera en utilisant des sockets client- serveur.

Au lancement de l'application, un client se connecte à ces trois serveurs à travers des sockets. Chaque client dispose de trois sockets contenant l'adresse du serveur et le numéro du port sur lequel la connexion s'établie entre les deux. Ainsi, nous déterminons trois liaisons essentielles :

- Client-serveur de contexte statique ;
- Client-serveur de contexte dynamique ;
- Client-serveur hébergeur de services.

Au niveau de chaque serveur ou client est implémeté le code suivant permettant de se connecter :

- **Coté client :**

Socket ClientSocket ;
String addr = "adresse de la machise où se trouve le serveur" ;
int port = numéro de port ;
ClientSocket = new Socket (addr, port) ;

- **Coté serveur :**

int port = numéro de port ;
int nbr = nombre de client qui se connecte au serveur en simultané ;
Socket ServerSocket ;
ServerSocket = new Socket (port, nbr) ;

À l'établissement de la connexion, un message est envoyé aux serveurs du contexte statique et dynamique, contenant l'identifiant du visiteur et son type, en faisant appel à une méthode 'SendMessage' comme suit :

String id = "identifiant du visiteur "; SendMessage (id) ;

Le serveur du contexte statique lit le message venant de la part du client et récupère ses informations contextuelles à partir de la base de données via une requête SQL :

*String request = "SELECT * FROM 'visiteurs' WHERE 'id'='id'";*

Ensuite il effectue des tests sur les champs nbr (nombre de visite), état de santé, profession, centre d'intérêt, ect. Afin de déterminer le service approprié. Nous citons comme exemple :

- **Le service consigne**

  Si le nombre de visites est égale à 1 alors : interroger le serveur hébergeur de services par une requête contenant l'identifiant du visiteur et le service consigne à déclencher. Ainsi les consignes lui seront affichées dans son interface d'accueil.(Figure 5.12)

Le serveur du contexte dynamique lit les messages en provenance du client, afin de soustraire son identifiant et collecter ses position dans le musée en faisant appel à l'application capteur. Nous allons illustrer les deux scénarios suivants :

- **Service description :** le serveur du contexte dynamique teste les positions récupérées à celles des cellules. Si l'une d'elles coïncide avec celle d'une oeuvre, ce dernier sollicite le serveur hébergeur de services pour lancer le service de description et l'afficher sur l'interface d'accueil du visiteur.(Figure 5.12).
- **Service guidage :** le serveur du contexte dynamique vérifie si le visiteur a choisi d'être guidé, alors, le serveur hébergeur de services sera interrogé pour transmettre le service de guidage afin d'assister le visiteur dans le musée (Figure 5.12(2)) . L'algorithme est le suivant :

  Etablir la connexion client-serveur du contexte dynamique (id, type visiteur) ;
  Si type visiteur = 'guidé' alors interroger le serveur hébergeur de service (id) ;
  Lancer le service (id) ;
  Afficher les informations relatives au guidage sur le périphérique du visiteur ;

5.5.2 La partie capteur

C'est un programme implémenté sous java et qui représente un capteur de position. En effet, ce dernier nous permet de renvoyer la position du pointeur de la souris selon deux valeurs (x, y) en l'occurrence : l'axe des X et l'axe des Y (Figure 5.13). Les positions des visiteurs au sein du musée seront donc remplacées par les mouvements de la souris. Cette application est interrogée par le serveur de contexte dynamique par une connexion client-serveur.

5.5.3 La partie client

1. **Lancer l'application**

   Une fois l'émulateur lancé le visiteur aura à sa vision l'ensemble des applications dont le périphérique mobile dispose y compris celle de la visite du musée. Il lui suffit alors de cliquer sur l'icône "**Musee**" pour que l'application démarre et lui affiche alors, l'interface d'accueil.

2. **Inscription/connexion**

   L'inscription se fait par le remplissage d'un formulaire contenant les informations requises pour l'ajout d'un nouveau visiteur : le nom, le prénom, l'identifiant, un mot de passe, etc. du visiteur. Après confirmation, ce dernier peut se connecter à son compte en insérant son identifiant et son mot de passe (Figure 5.8).

3. **Récupération du mot passe**

   L'application compte le nombre d'échecs de tentative de connexion. Elle teste avant chaque connexion si ce nombre est égal à trois tentatives. Dans ce cas le système récupère auprès de la base de données la question secrète et la réponse choisis lors de l'inscription du visiteur via une requête SQL, lui affiche la question et lui demande de saisir la réponse. Si cette dernière est égale à la réponse récupérée de la base de données, son mot de passe lui sera affiché après l'avoir récupère aussi par une requête SQL. Si le visiteur n'arrive pas à répondre correctement à la question posée au bout de trois tentatives, un message lui sera affiché lui demandant de passer à l'accueil du musée afin de le récupérer (Figure 5.10).

5.5.4 La partie administration

C'est l'application réservée aux administrateurs du musée. Grâce à cette dernière, un administrateur gère les oeuvres et les services du musée. Il pourra donc effectuer les opérations d'ajout, de suppression et de modifications et d'autres opérations. Cette partie est implémentée sous java et connectée à la base de données en utilisant un fichier jar (JDBC). Le JDBC est une API qui fait partie intégrante de la plate-forme Java, et qui est constituée de classes permettant l'accès à la base de données depuis l'application Java.

5.6 Structure de l'application client et gestion du musée

5.6.1 L'application client

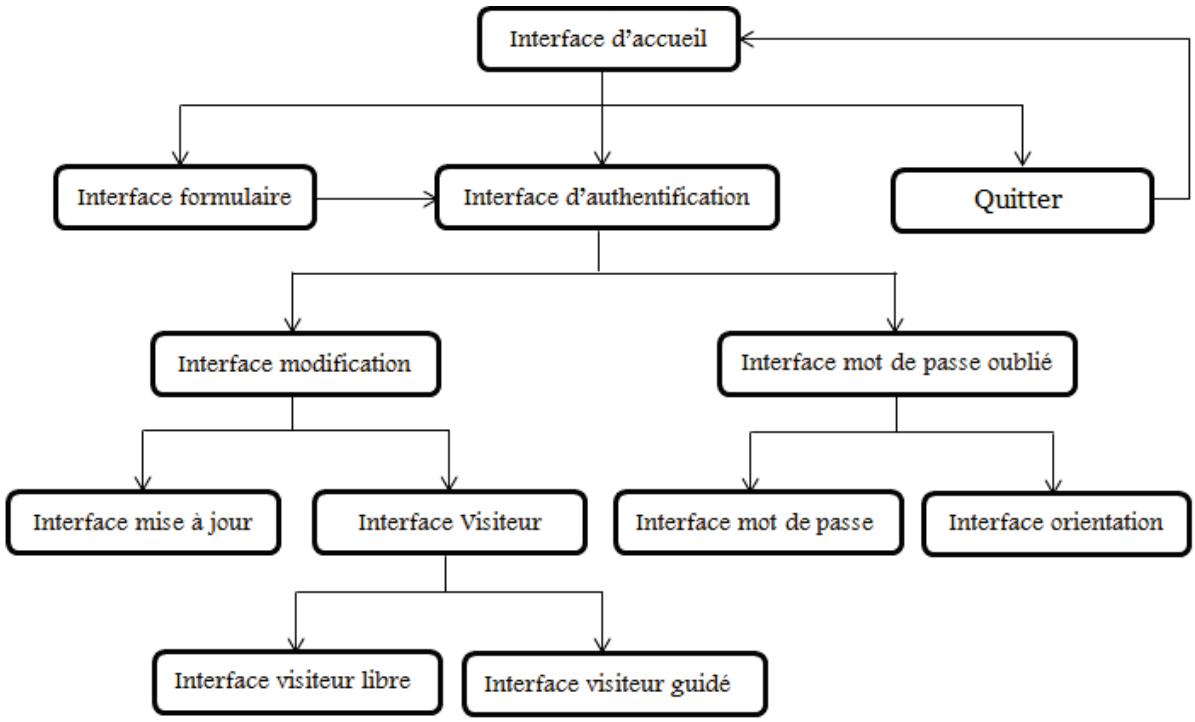


FIGURE 5.5 – Plan de l'application client.

5.6.2 L'application de gestion

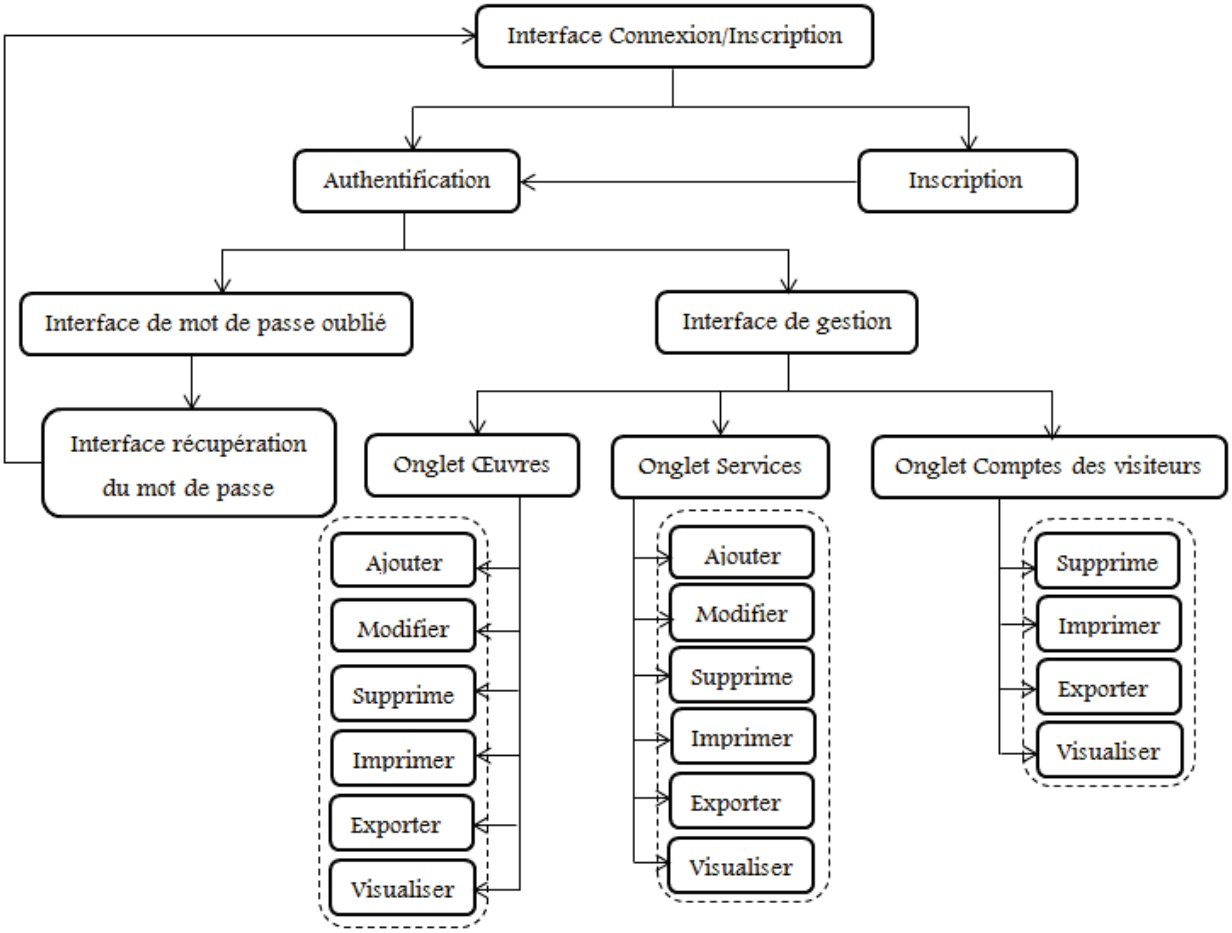


FIGURE 5.6 – Plan de l'application de gestion.

5.7 Captures d'interfaces de l'application

Dans cette partie nous allons exposer quelque interfaces de l'application avec des explications :

5.7.1 L'application client

5.7.1.1 Interface d'accueil

C'est la première interface qui s'affiche après le lancement de l'application. Elle contient principalement trois boutons : "Lancer", "S'inscrire" et "Quitter" (Figure 5.7).

Selon le choix effectué, le visiteur sera redirigé soit vers l'interface "inscription" Figure(5.8 (1)) ou l'interface "authentification" Figure(5.8 (2)).

FIGURE 5.7 – Interface d'accueil du visiteur.

FIGURE 5.8 – (1)Interface d'inscription du visiteur. (2) Interface d'authentification du visiteur.

5.7.1.2 Interface mot de passe oublié

C'est l'interface qui permet à un visiteur de récupérer son mot de passe (Figure 5.9). Selon sa réponse, il peut être orienté soit vers l'interface "mot de passe " (Figure 5.10 (1)) ou vers l'interface "orientation" (Figure 5.10 (2)).

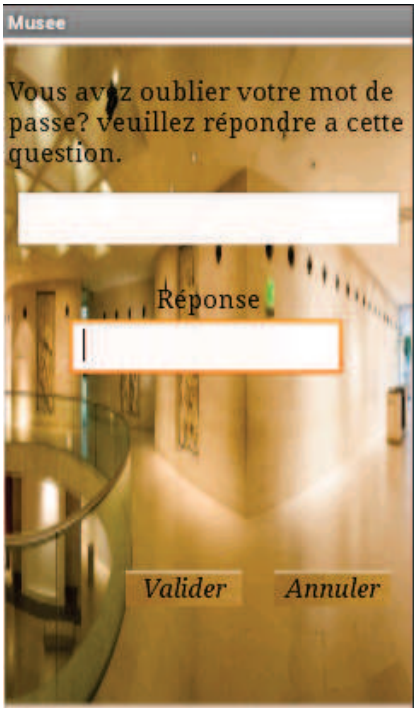


FIGURE 5.9 – Interface 'mot de passe oublié'.

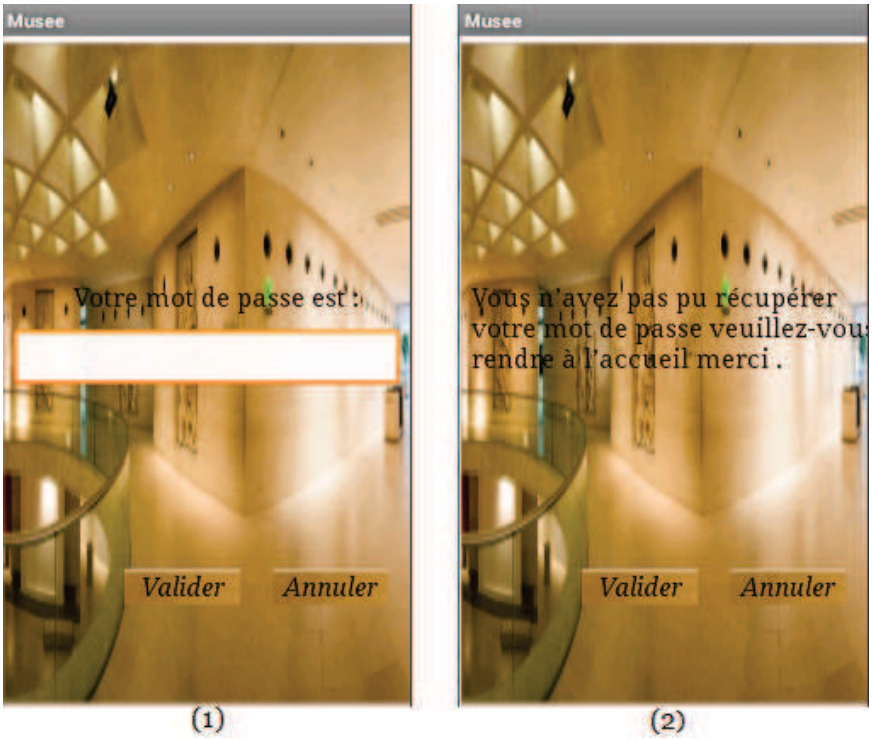


FIGURE 5.10 – (1) Interface affichage du mot de passe du visiteur, (2) interface d'orientation du visiteur en cas d'oubli de la réponse à la question secrète.

5.7.1.3 Interface type du visiteur

Dans cette interface, le visiteur a le choix de déterminer quel type de visiteur veut être (Libre ou guidé) ainsi il sera orienté vers l'interface principale de l'application (Figure 5.11).

FIGURE 5.11 – Interface type du visiteur.

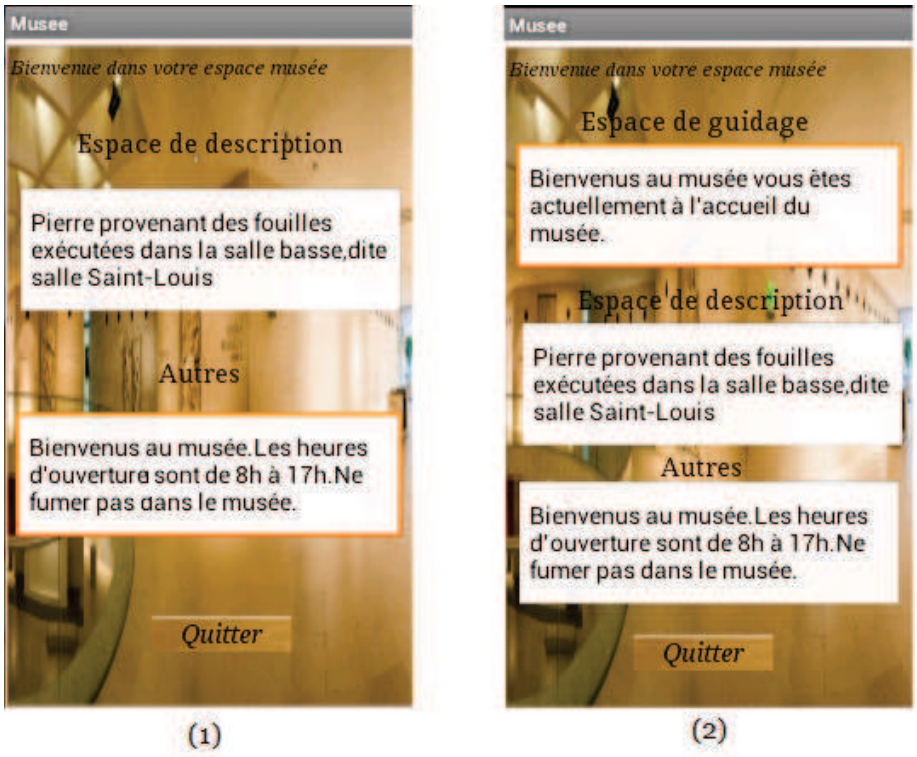


FIGURE 5.12 – (1) Interface visiteur libre, (2) interface visiteur guidé.

5.7.2 L'application capteur

L'application capteur est dotée d'une interface représentant les différents étages du musée :

- Rez-de-chaussée (Accueil) ;
- Histoire et civilisations ;
- Art ;
- Education ;
- Informatique.

Comme montré dans l'interface, l'étage histoire et civilisations et l'étage art contiennent une variété d'oeuvres. (Figure5.13)

FIGURE 5.13 – Application capteur.

5.7.3 L'application de gestion du musée

5.7.3.1 Interface Accueil

C'est la première interface affichée lors du lancement de l'application gestion de musée. Elle s'affiche le temps de lancement de la fenêtre principale (Figure 5.14).

FIGURE 5.14 – L'interface d'accueil de l'administrateur.

5.7.3.2 Interface Connexion

La fenêtre 'Connexion' est l'interface principale de l'application 'gestion du musée'. Elle se divise en deux parties : une pour l'authentification et l'autre pour la création d'un nouveau compte administrateur (Figure 5.15).

FIGURE 5.15 – L'interface d'accueil de l'administrateur.

Après trois echècs de tentatives de connexion au compte administrateur, une nouvelle fenêtre (Figure 5.16) lui sera affichée, indiqunt la possibilité de récupérer le mot de passe.

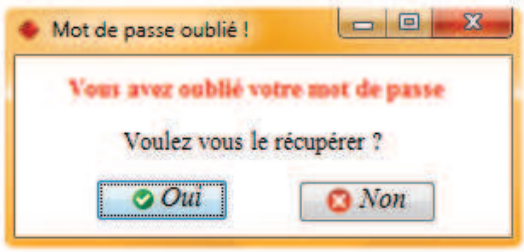


FIGURE 5.16 – Messages d'erreur (mot de passe oublié).

L'administrateur a le choix entre récupérer ou ne pas récupérer son mot de passe. Dans le cas où il clique sur oui, une fenêtre lui sera affichée contenant la question secrète qui a été choisi lors de son inscription (Figure 5.17).

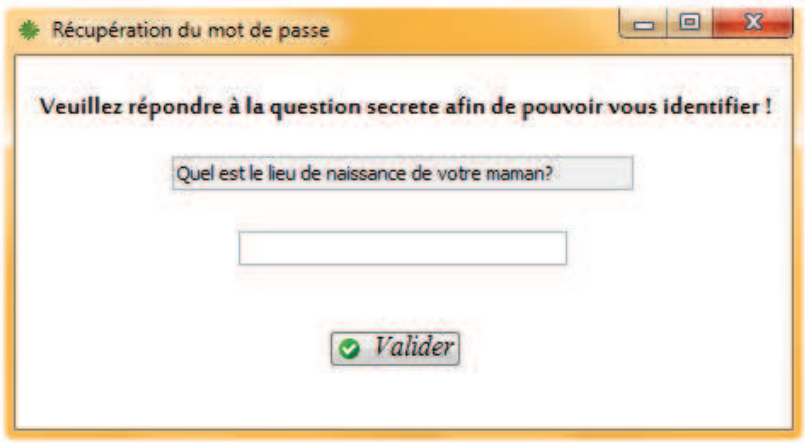


FIGURE 5.17 – Interface récupération du mot de passe.

5.7.3.3 Interface Gestion du musée

C'est l'interface qui s'affiche après une authentification de l'administrateur. Elle est composée de trois onglets : l'onglet 'Ouvres' qu'on aperçoit comme première vue de l'interface 'Gestion du musée', l'onglet 'Service', l'onglet 'Comptes des visiteurs' et l'onglet 'statistiques' (Figure5.18).

FIGURE 5.18 – Espace de gestion du musée (onglet oeuvre).

Si l'administrateur souhaite observer clairement le contenu de l'un des tableaux, il n'a qu'à cliquer sur le bouton 'Visualiser'. Par exemple si l'on est sur l'onglet 'oeuvres', cliquer sur le bouton 'Visualiser' mène vers l'interface 'Les oeuvres du musées' qui dispose de trois onglets. Chacun de ces derniers est découpé en trois parties : un espace sous forme d'un tableau contenant des informations sur des oeuvres, une biographie d'un auteur et une description d'une oeuvre. Á la sélection d'une oeuvre, les deux champs : biographie de l'auteur et description de l'oeuvre seront affichés dans leurs espaces réservés (Figure 5.19).

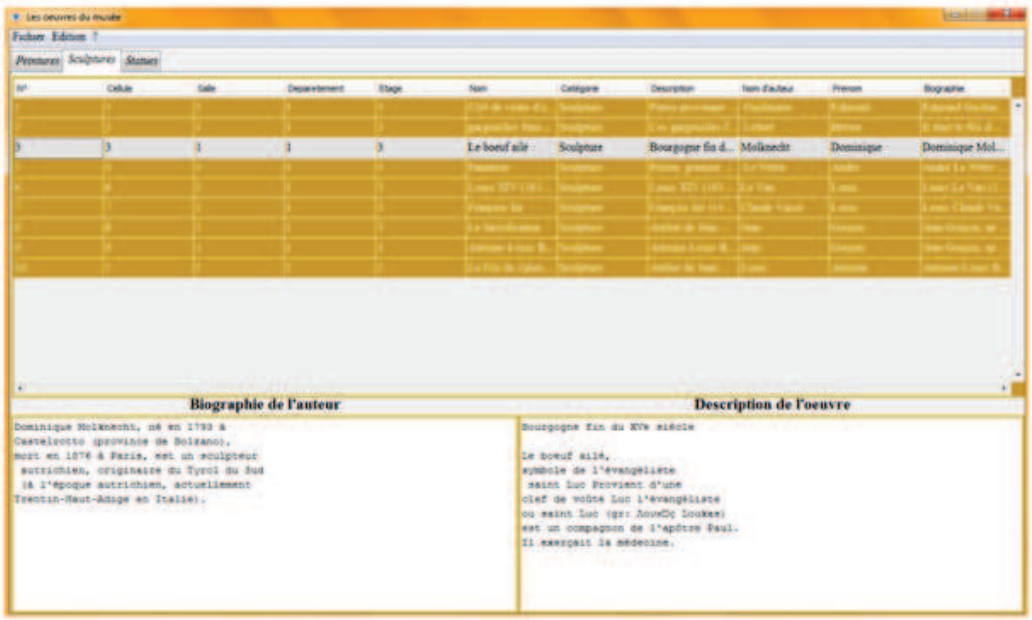


FIGURE 5.19 – Espace de gestion du musée (visualisation oeuvre).

5.8 Conclusion

La phase de réalisation est l'étape la plus importante dans le cycle de vie d'une application.

Ce chapitre constitue un essai de proposition d'implémentation d'une application mobile sensible au contexte. Les résultats qu'on a pu avoir sont la visualisation des changements des descriptions des services du musée en fonction du contexte du visiteur.

Conclusion générale

L'informatique ubiquitaire constitue la pierre angulaire de l'informatique future. Ce domaine combine à la fois les aspects de l'informatique distribuée et de l'informatique mobile et adopte une nouvelle vision des équipements et des applications.

L'un des aspects contribuant à la réalisation de cette notion d'informatique ubiquitaire est de permettre à ces systèmes informatiques de s'adapter pro-activement aux changements du contexte des utilisateurs. Dans un tel contexte les services doivent pouvoir s'adapter de manière dynamique au contexte de ces derniers avec un minimum d'intervention de leurs parts afin de satisfaire leurs besoins. C'est alors sur cette vision que notre travail s'est focalisé.

En abordant une variété de notions telles que l'informatique diffuse, la sensibilité au contexte et les applications mobiles nous avons essayé de mettre en oeuvre ces dernières et de construire un système ubiquitaire pour un musée.

L'application implémentée s'articule autour de quatre serveurs logiciels principaux : **un serveur de base données** où toutes les informations du musée sont stockées (oeuvres, tarifications, actualités, etc.) ainsi que les informations des visiteurs en particulier leurs profils qui représentent leurs contexte statique ; **un serveur de contexte statique** qui se charge de collecter les informations statiques de chaque visiteur au lancement de l'application mobile 'musée' qui doit être installée sur le périphérique du visiteur au préalable ; **un serveur de contexte dynamique** qui a pour rôle la collecte des positions des visiteurs à l'intérieur du musée ; **un serveur hébergeur de services** qui traite les informations issues des deux serveurs précédents (serveur de contexte statique et serveurs de contexte dynamique) et les comparent aux descriptions des services afin de prédire les besoins des visiteurs.

L'utilisation de ces informations contextuelles pour la découverte de services fait de l'application implémentée un système "intelligent", capable de réagir de manière dynamique et automatique sans intervention de l'utilisateur (visiteur) ce qui répond à l'un des objectifs principaux de l'informatique ubiquitaire.

Ce projet nous a été très bénéfique, car il nous a permis d'enrichir nos connaissances sur les deux plans : théorique et pratique. Il nous a aussi permis de découvrir et d'acquérir de nouvelles connaissances en matière de programmation et de développement dans le domaine des applications mobiles (Android).

Le développement d'un système ubiquitaire nécessite énormément du temps, de ressources, de recherches et aussi de grands efforts de programmation.

Donc, faute de temps et de moyen certains points n'ont pas été pris en compte dans la partie réalisation de l'application de visite du musée et d'autre n'ont pas été achevés.

En ce qui est de l'application capteur, on a essayé dans un premier lieu de créer un simulateur pour les capteurs, et cela en utilisant un projet client ayant les mêmes classes que le Framework Android. Et un projet serveur sous Visual Editor 6 qui permet de renvoyer des positions dès qu'il y ait une connexion de la part d'un client mais cela a nécessité la possession d'un téléphone portable à base d'Android, de plus même si on utilise le téléphone, le client sera obligé d'actualiser à chaque fois l'application afin de donner une nouvelle position, ce qui est contradictoire avec notre système qui devait être transparent. En effet, la solution finale consiste à s'en servir des mouvements de la sourie à l'écran pour simuler les positions du visiteur à l'intérieur du musée.

Ces contraintes de temps et de ressources nous ont permis de tracer une ligne de perspectives, nous évoquons :

- La duplication des serveurs de l'application : nous proposons donc d'utiliser des serveurs miroirs pour éviter la surcharge des serveurs, notamment le déni de service qui présente l'une des contraintes de la sécurité des réseaux informatiques ;
- Utiliser des capteurs matériaux devant chaque oeuvre qui vont permettre de limiter le nombre de tests effectués au niveau de chaque serveur ;
- Le déploiement sur le terrain pour mesurer ses performances ;
- Enrichir le profil des utilisateurs pour une meilleure gestion de contexte ;
- Etendre l'utilité de l'application à l'environnement de l'utilisateur (maison, etc.).

Glossaire

-A-

ADSL (**A**symmetrical **D**igital **S**ubscriber **L**ine numérique), technique de communication numérique.

API (**A**pplication **P**rogramming **I**nterface), interface de programmation d'applications.

Arctefacts Un artefact est la spécification d'un élément physique qui est utilisé ou produit par le processus de développement du logiciel ou par le déploiement du système. C'est donc un élément concret comme par exemple : un fichier, un exécutable ou une table d'une base de données. Un artefact peut être relié à d'autres artefacts par notamment des liens de dépendance.

AURA (**A**dvanced **U**ser **R**esource **A**nnotation), système permettant de scanner un code à barres des objets.

AVD (**A**ndroid **V**irtual **D**evice), un appareil virtuel d'Android.

-B-

Béta test Lors du lancement de la période de simulation et d'évaluation des projets informatiques, les concepteurs ont souvent recours au bêta-testeurs, qui essayent leurs applications gratuitement , en bénéficiant de l'exclusivité des fonctionnalités, plus ou moins complètes.

-C-

CDMA(**C**ode **D**ivision **M**ultiple **A**ccess), désigne un procédé de téléphone sans fils utilisant un large spectre de fréquence.

CD(**C**opact **D**isc), disque copact permet de stocker des donées.

CLS (**C**omputer **S**cience **L**aboratory), laboratoire de science informatique.

CMF(**C**ontext **M**anagement **F**ramework), cadre de travail de gestion de contexte.

CML(**C**ontext **M**odeling **L**anguage), langage de modélisation de contexte.

CoBrA(**C**ontext **B**roker **A**rchitecture), Architecture de courtier de contexte.

ConteXtML(**C**ontexte **M**arkup **L**anguage), langage de balisage de contexte.

CORTEX (**CO**-operating **R**eal-**T**ime senTient objects : architecture and **Ex**perimental evaluation), Objets sensibles de coopération en temps réel : architecture et évaluation expérimentale

CSS(**C**ascading **S**tyle **S**heet), modele de feuille de style.

-D-

DTP (**D**ata **T**ools **P**latform), modèle de gestion distribuée des transactions.

DUPS (**D**imension **U**ser **P**rofil **S**ystem), méthode permettant de recommander des services à l'utilisateur en fonction de ses préférences.

DUMBO(**D**ynamic **U**biquitous **M**obile **M**eeting **B**oard), un tableau physique blanc utilisé lors d'une réunion.

DVD(**D**isc **V**ersatile **D**egital), disque vidéo digital.

-E-

EMF (**E**clipse **M**odeling **F**ramework), c'est un cadre de modélisation et des installations de génération de code pour les outils de construction et d'autres applications basées sur un modèle de données structuré

-G-

GC (**G**arbage **C**ollector), c'est un gestionnaire automatique de la mémoire.

GEF (**G**raphics **E**diting **F**ramework), c'est un cadre d'édition graphique qui fournit la technologie pour créer des éditeurs graphiques riches.

GMF(**G**raphical **M**odeling **F**ramework), c'est un projet de modélisation graphique qui fournit un ensemble de composants génératifs et les infrastructures d'exécution pour le développement des éditeurs graphiques.

GPRS (**G**eneral **P**acket **R**adio **S**ervice), service général de transmission radioélectrique par paquet.

GPS (**G**lobal **P**ositioning **S**ystem), système de positionnement par satellite.

GSM(**G**lobal **S**ystem for **M**obile communications), Système global pour les communications mobiles.

-H-

HP(**H**ewlett **P**ackard), alouette-placard.

HTML (**H**yperText **M**arkup **L**anguage), format de document du web.

HTTP (**H**yper **T**ext **T**ransfer **P**rotocol), protocole de transmission client-serveur.

HTTPS (**H**yper **T**ext **T**ransfer **P**rotocol **S**ecure), protocole de transmission client-serveur sécurisé.

-I-

IDE (**I**ntegrated **D**evelopment **E**nvironment), environnement de développement intégré.

-M-

MIDP (**M**obile **I**nformation **D**evice **P**rofile), c'est un profil J2ME utilisé par certains téléphones mobiles.

MID (**M**obile **I**nternet **D**evice), un appareil de taille intermédiaire entre le smartphone et l'ordinateur portable, offrant un accès sans fil à Internet

MMS (**M**ultimedia **M**essaging **S**ystem), système de messagerie multimédia.

-O-

OCL (**O**bject **C**onstraint **L**anguage), est un langage informatique d'expression des contraintes utilisé par UML .

OHA(**O**pen **H**andset **A**lliance), est un consortium de plusieurs entreprises dont le but est de développer des normes ouvertes pour les appareils de téléphonie mobile.

OWL (**O**ntology **W**eb **L**anguage), langage web d'ontologie.

-P-

PC (Personal computer), ordinateur personnel.

PDA (Personal Digital Assistant), assistant numérique personnel.

PDF (Portable Document Format), format de document portable.

PHP (Hypertext Preprocessor), est un langage de programmation utilisé pour produire des pages Web dynamiques via un serveur HTTP

-R-

RFID (Radio Frequency Identification), Identification par Radio Fréquence.

RIM (Research In Motion), société canadienne spécialisée dans la conception, la fabrication et la commercialisation de solutions sans fil pour le marché de la communication mobile.

RUP (Rational Unified Process), processus unifé de Rational.

-S-

SAMU Service Ambulatoire d'urgences.

SDK (**S**oftware **D**evelopment **K**it),kit de développement logiciels.

SFR Société Française de Radiologie.

SGBD(en anglais **DBMS** pour **D**ata**B**ase **M**anagement **S**ystem), système de gestion de base de données.

SGML (**S**tandard **G**eneric **M**arkup **L**anguage) , Langage de balisage générique standard.

SID **S**ystème **I**nformatique **D**iffus.

SIM (**S**ubscriber **I**dentification **M**odule), est une puce contenant un microcontrôleur et de la mémoire. Elle est utilisée en téléphonie mobile pour stocker les informations spécifiques à l'abonné d'un réseau mobile.

SMS (**S**hort **M**essage **S**ervice), service de message court.

SOCAM(**S**ervice-**O**riented **C**ontext-**A**ware **M**iddleware), intergiciel sensible au contexte orienté service.

SOUPA (**S**tandard **O**ntology for **U**biquitous and **P**ervasive **A**pplications), ontologie standard pour les applications ubiquitaires et diffuses.

SQL (**S**tructured **Q**uery **L**anguage), langage d'interrogation de bases de données.

SSL(**S**ecure **S**ocket **L**ayer), couche de sockets sécurisées.

-T-

TDMA (Temporal Division Multiple Access), accès multiple à répartition dans le temps.

TPTP(Eclipse Test and Performance Tools Platform),c'est le projet de base dans le projet de niveau supérieur Eclipse.

TV : TéléVision.

-U-

ULE(**U**biquitous **L**earning **E**nvironnement), Environnement d'apprentissage ubiquitaire.

UML(Unified Modeling Language), langage de modélisation unifié.

UP(Unified Process), processus unifié.

URL(**U**niforme **R**essource **L**ocator), méthode d'accès aux documents distants.

-V-

VPN(**V**irtual **P**rivate **N**etwork), réseaux virtuels privés.

-W-

WiFi Norme IEEE 802.11 pour la communication sans fil.

WLAN(**W**ireless **L**ocal **A**rea **N**etwork), réseaux wifi locaux.

WMAN(**W**ireless **M**etropolitan **A**rea **N**etwork), réseaux wifi métropolitains.

WPAN(**W**ireless **P**ersonal **A**rea **N**etwork), réseaux wifi personnels.

WTP (Eclipse **W**eb **T**ools **P**latform), Il s'agit d'un ensemble d'outils Eclipse pour le developpement d'applications.

WWAN(**W**ireless **W**ide **A**rea **N**etwork), réseaux wifi distants.

-X-

Xerox PARC(**X**erox **P**alo **A**lto **R**esearch **C**enter), est un centre de recherches en informatique situé à Palo Alto en Californie.

XML(e**X**tended **M**arkup **L**anguage), langage de balisage étendu.

XP (e**X**treme **P**rogramming), est une méthode agile plus particulièrement orientée sur l'aspect réalisation d'une application.

Annexe

Cette partie sert d'un complémentaire des deux chapitres 'conception' et 'réalisation', elle comprend quelques notions non abordées au préalable dans ces chapitres.

Elle permet dans un premier lieu d'illustrer les étapes d'installation et de préparation des environnements de travail pour (Eclipse et Android). Pour donner en suite quelques définitions en relation avec les bases de données et un exemple pour la création de la base de données 'musée' et des tables (visiteurs et cellules).

Au final, elle comporte des définitions concernant quelques notations utilisées dans les diagrammes d'UML lors de la conception de notre application.

1. **Eclipse**

   (a) **Installation et préparation de l'environnement Eclipse IDE**

   La procédure étant simple et se résume en trois étapes suivantes :

   – **Etape 1 : téléchargement et installation d'Eclipse 3.3**

   Nous avons utilisé la version 3.3 d'eclipse que nous avions téléchargé depuis le site d'IBM : **http ://www.ibm.com/developerworks/eclipse /downloads/europa**. Il existe notament plusieurs distributions d'Eclipse avec plus au moins de plugins et d'outils installés. Nous avons opté pour le bundle le plus complet : Europa enterprise project bundle. Ce package comprend Core Eclipse 3.3, Eclipse Web Tools Platform (WTP) 2.0, Eclipse Modeling Framework (EMF) 2.3, Graphics Editing Framework (GEF) 3.3, Data Tools Platform (DTP) 1.5, Eclipse Test and Performance Tools Platform (TPTP) 4.4, Eclipse Web Tools Platform (WTP) 2.0, Graphical Modeling Framework (GMF) 2.0, EMF (OCL, Validation, Query, Transaction) 1.1, c'est à dire, entre autre, tous les prérequis pour pouvoir installer Visual Editor, à savoir Eclipse Modeling Framework (EMF) et le Graphical Mo-

deling Framework (GMF). ? la fin du téléchargement, il faut décompresser l'archive .zip dans le répertoire d'installation souhaité.

- **Etape 2 : téléchargement et installation de la dernière version du Web Tool Plateform**

  Le Web Tool Plateform est fourni de base avec le bundle d'Eclipse que nous avons récupéré. Il est néanmoins préférable de télécharger et d'installer la dernière version de ce composant depuis le site officiel :
  http ://www.eclipse.org/webtools. Après l'avoir téléchargé, nous l'avons décompressé dans le même répertoire que celui d'Eclipse.

- **Etape 3 : Téléchargement et installation de visual editor**

  Cette dernière étape consiste à télécharger et installer une bonne version de Visual Editor. Sur le site : **http ://www.ehecht.com/eclipse_ve/ve.html** se trouve une version du plugin testée et approuvée avec le bundle Eclipse 3.3. Le fichier s'appelle **ve_eclipse_33_v200709242300_win.zip.** Á la fin aussi, le fichier téléchargé doit être décompressé dans le répertoire d'installation d'Eclipse.

  Pour pouvoir vérifier que tout s'est bien passé, il faut démarrer Eclipse. Se rendre ensuite dans "Help puis Software ensuite Updates au final sur Manage Configuration". Une nouvelle fenêtre s'ouvre alors et doit afficher tous les plugins et composants installés dans l'Eclipse. Si Visual Editor s'est bien installé, une ligne "Visual Editor 1.3.0.v200709242300" apparaisse.

2. **Android**

   (a) **présentation**

   Android dispose d'une architecture principale qui lui associer afin de mieux pouvoir protéger le fonctionnement des téléphone, il comprend les :

   - **Activity (Activité)**

     La brique de base de l'interface utilisateur s'appelle activity (activité). Vous pouvez la considérer comme l'équivalent Android de la fenêtre ou de la boîte de dialogue d'une application classique. Bien que des activités puissent ne pas avoir d'interface utilisateur, un code "invisible" sera délivré le plus

souvent sous la forme de fournisseurs de contenus (content provider) ou de services.

– **Content providers (fournisseurs de contenus)**

Les fournisseurs de contenus offrent un niveau d'abstraction pour toutes les données stockées sur le terminal est accessibles aux différentes applications. Le modèle de développement Android encourage la mise à disposition de ses propres données aux autres programmes - construire un content provider permet d'obtenir ce résultat tout en gardant un contrôle total sur la façon dont on accédera aux données.

– **Intents (intentions)**

Les intentions sont des messages système. Elles sont émises par le terminal pour prévenir les applications de la survenue de différents événements, que ce soit une modification matérielle (comme l'insertion d'une carte SD) ou l'arrivée de données (telle la réception d'un SMS), en passant par les événements des applications elles-mêmes (votre activité a été lancée à partir du menu principal du terminal, par exemple). Vous pouvez non seulement répondre aux intentions, mais également créer les vôtres afin de lancer d'autres activités ou pour vous prévenir qu'une situation particulière a lieu (vous pouvez, par exemple, émettre l'intention X lorsque l'utilisateur est à moins de 100 mètres d'un emplacement Y).

– **Services**

Les activités, les fournisseurs de contenus et les récepteurs d'intentions ont une durée de vie limitée et peuvent être éteints à tout moment. Les services sont en revanche conçus pour durer et, si nécessaire, indépendamment de toute activité. Vous pouvez, par exemple, utiliser un service pour vérifier les mises à jour d'un flux RSS ou pour jouer de la musique, même si l'activité de contrôle n'est plus en cours d'exécution.

– **Fonctionnalités**

Android fournit un certain nombre de fonctionnalités pour vous aider à développer des applications.

– **Stockage**

Vous pouvez empaqueter (packager) des fichiers de données dans une application, pour y stocker ce qui ne changera jamais - les icônes ou les fichiers d'aide, par exemple. Vous pouvez également réserver un petit emplacement sur le terminal lui-même, pour y stocker une base de données ou des fichiers contenant des informations nécessaires à votre application et saisies par l'utilisateur ou récupérées à partir d'une autre source. Si l'utilisateur fournit un espace de stockage comme une carte SD, celui-ci peut également être lu et écrit en fonction des besoins.

- **Réseau**

  Les terminaux Android sont généralement conçus pour être utilisés avec Internet, via un support de communication quelconque. Vous pouvez tirer parti de cet accès à Internet à n'importe quel niveau, des sockets bruts de Java à un widget de navigateur web intégré que vous pouvez intégrer dans votre application.

- **Multimédia**

  Les terminaux Android permettent d'enregistrer et de jouer de la musique et de la vidéo. Bien que les caractéristiques spécifiques varient en fonction des modèles, vous pouvez connaître celles qui sont disponibles et tirer parti des fonctionnalités multimédias offertes, que ce soit pour écouter de la musique, prendre des photos ou enregistrer des mémos vocaux.

- **GPS**

  Les fournisseurs de positionnement, comme GPS, permettent d'indiquer aux applications où se trouve le terminal. Il vous est alors possible d'afficher des cartes ou d'utiliser ces données géographiques pour retrouver la trace du terminal s'il a été volé, par exemple.

- **Services téléphoniques**

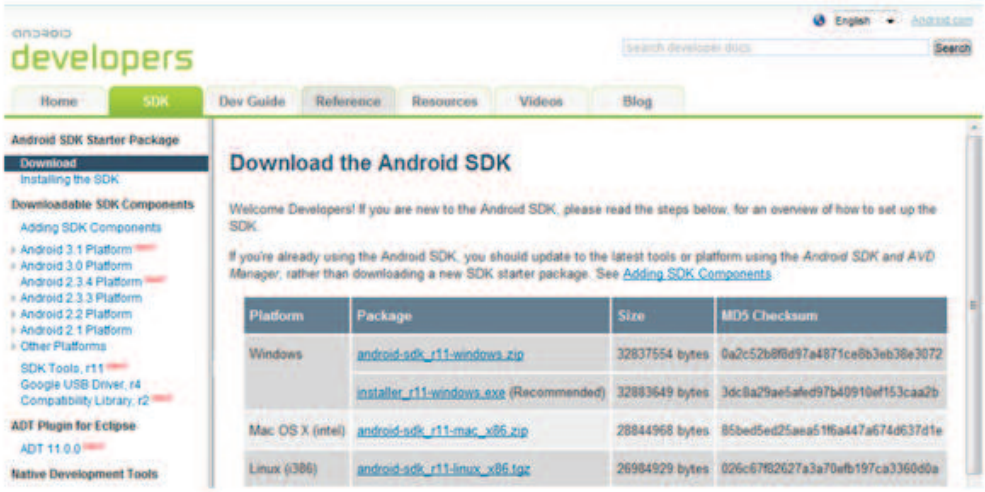
  Evidemment, les terminaux Android sont généralement des téléphones, ce qui permet à vos programmes de passer des appels, d'envoyer et de recevoir des SMS et de réaliser tout ce que vous êtes en droit d'attendre d'une technologie téléphonique moderne.

(b) **Installation d' Android**

– **Téléchargement d'eclipse et du SDK Android**

Pour développer une application Android nous aurons besoin de l'éditeur du texte Eclipse qu'on a téléchargé dans la section précédente, comme on peut télécharger la toute nouvelle version sur le site :

http ://www.eclipse.org Nous aurons aussi besoin de du SDK Andoid qu'on a récupéré sur : **http ://www.developer.android.com**.



Sur l'onglet SDK, on télécharge la dernière version du SDK Anroid pour Windows. Le SDK se présente sous la forme d'un fichier ZIP contenant uniquement la dernière version des outils de développement Android.

– **Installation d'eclipse et présentation de l'IDEInstallation d'eclipse et présentation de l'IDE**

Á la fin du téléchargement d'eslipse et du SDK Android, il ne reste plus qu'à les décompresser dans un même dossier.

Une fois le compactage des deux fichiers zip est effectué, il suffit d'aller sur le dossier eclipse et de lancer eclipse.exe pour d'intégrer le SDK Android et le pluging Android.

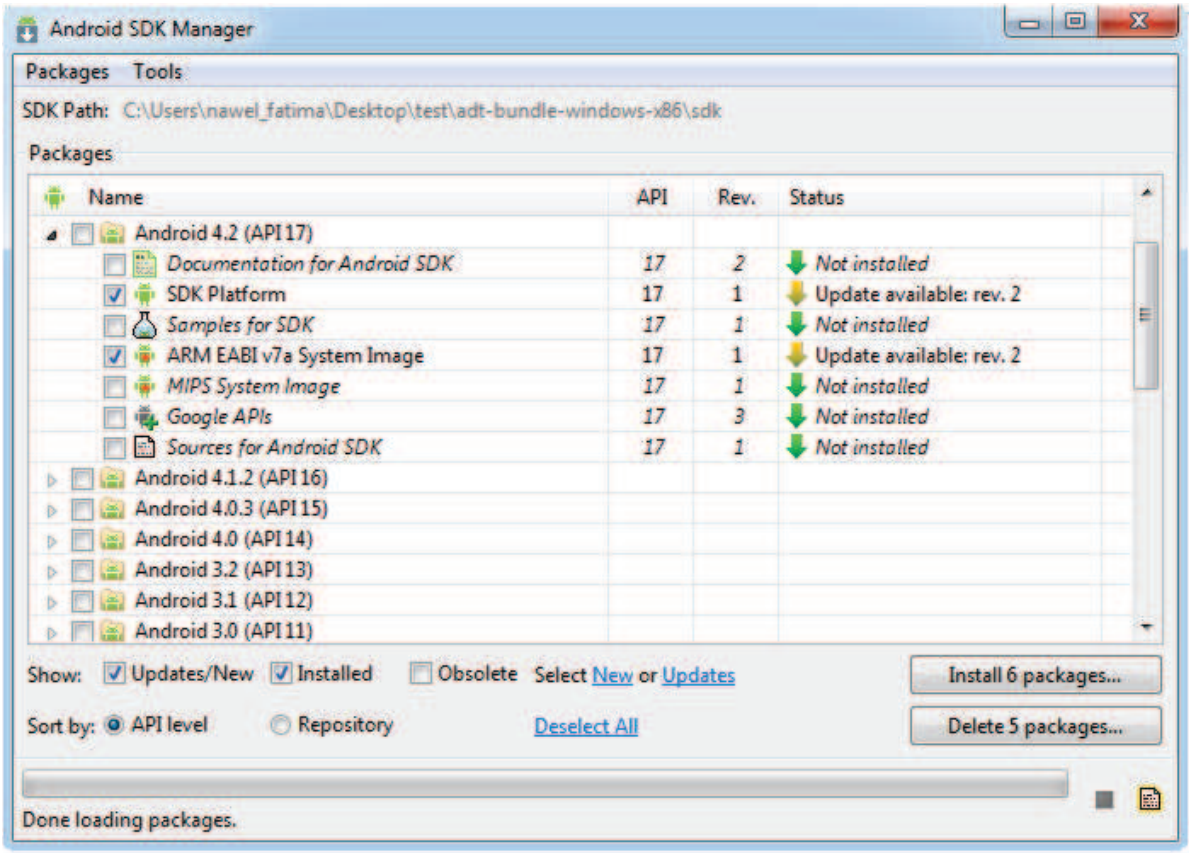
Pour intégrer le pluging Android,on se rend Dans le menu 'Help' puis 'Instal New Software' et on tape l'adresse suivante :

http ://dlssl.gougle.com/android/eclipse puis on clique sur 'ajouter', on lui donne un nom (Plug in Android) et on clique sur ok. La eclipse va chercher les informations et après la confirmation le télécharge du pluging

Android est lancé.

En fin de téléchargement, le redémarrage d'eclipse est requis. Pour installer le SDk, il suffit d'aller sur le menu 'Window' "preferences"'Android' et on insère le chemin où se trouve le SDK Android dans 'le SDK location', on clique en suite sur appliquer et ok.

Afin de gérer au mieux la variété des versions des systèmes d'exploitation, de SDK et de terminaux, un outil, Android SDK Manager, a été mis en place. Ce petit utilitaire permet de récupérer les différentes entités nécessaires au développement sur Android.



Effectivement, avant de pouvoir commencer à développer, nous devons ajouter au moins une plateforme SDK grâce à ce petit utilitaire. Maintenant l'environnent est prêt pour être utiliser pour le développement Android.

(c) **Structure d'un projet**

Le système de construction d'un programme Android est organisé sous la forme d'une arborescence de répertoires spécifique à un projet, exactement comme n'importe quel projet Java. Les détails, cependant, sont spécifiques à Android et à sa préparation de l'application qui s'exécutera sur le terminal ou l'émulateur.

i. **Contenu de la racine**

La création d'un projet Android (avec la commande android create project ou via un environnement de programmation adapté à Android) place plusieurs éléments dans le répertoire racine du projet :

- **AndroidManifest.xml** est un fichier XML qui décrit l'application à construire et les composants - activités, services, etc. - fournis par celle-ci.
- **build.xml** est un script Ant1 permettant de compiler l'application et de l'installer sur le terminal (ce fichier n'est pas présent avec un environnement de programmation adapté, tel Eclipse).
- **default.properties** et **local.properties** sont des fichiers de propriétés utilisés par le script précédent.
- **bin/** contient l'application compilée.
- **gen/** contient le code source produit par les outils de compilation d'Android.
- **libs/** contient les fichiers JAR extérieurs nécessaires à l'application.
- **src/** contient le code source Java de l'application.
- **res/** contient les ressources - icônes, descriptions des éléments de l'interface graphique (layouts), etc.

  - empaquetées avec le code Java compilé.
- tests/ contient un projet Android entièrement distinct, utilisé pour tester celui que vous avez créé.
- assets/ contient les autres fichiers statiques fournis avec l'application pour son déploiement sur le terminal.

(d) **Création d'un projet Android**

Lors de la création d'un projet Android (avec android create project, par exemple), vous devez fournir le nom de classe ainsi que le chemin complet (paquetage) de l'activité "principale" de l'application (com.exemple.musee, par exemple). Vous constaterez alors que l'arborescence src/ de ce projet contient la hiérarchie des répertoires définis par le paquetage ainsi qu'un squelette d'une sousclasse d'Activity représentant l'activité principale (src/com/Musee/-Main.java). Vous pouvez bien sûr modifier ce fichier et en ajouter d'autres à l'arborescence src/ selon les besoins de votre application. La première fois que vous compilerez le projet (avec ant, par exemple), la chaîne de production d'Android créera le fichier R.java dans le paquetage de l'activité "principale". Ce fichier contient un certain nombre de définitions de constantes liées aux différentes ressources de l'arborescence res/. Il est déconseillé de le modifier manuellement : laissez les outils d'Android s'en occuper. Vous rencontrerez de nombreuses références à R.java dans les exemples de ce livre (par exemple, on désignera l'identifiant d'un layout par R.layout.main).

i. **Exemple d'une interface sous Android**

ii. Code source de l'interface

– Activity

```
package com.example.musee;
import com.example.musee.R;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.View;
import android.view.View.OnClickListener;
import android.view.animation.TranslateAnimation;
import android.widget.Button;

public class Main extends Activity {
        protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

         Button b=((Button)findViewById(R.id.button1));
          TranslateAnimation trans1 = new TranslateAnimation(0,0,480, 0);
          trans1.setStartOffset(480);
          trans1.setFillAfter(true);
          trans1.setDuration(2000);
          this.findViewById(R.id.button1).startAnimation(trans1);
           b.setOnClickListener(new OnClickListener() {

               public void onClick(View v) {
               Intent intent = new Intent(Main.this,Formulaire.class);  startActivity(intent);
               }
               });

               Button L=((Button)findViewById(R.id.button2));

               TranslateAnimation trans2 = new TranslateAnimation(0,0,480, 0);
          trans2.setStartOffset(480);
          trans2.setFillAfter(true);
          trans2.setDuration(3000);
          this.findViewById(R.id.button2).startAnimation(trans2);
          L.setOnClickListener(new OnClickListener() {
               public void onClick(View v) {
               Intent intent = new Intent(Main.this,Authentification.class);
startActivity(intent);
               }
               });
```

– **Suite du code**

```
        Button Q=((Button)findViewById(R.id.button3));
        TranslateAnimation trans3 = new TranslateAnimation(0, 0, 480, 0);
       trans3.setStartOffset(480);
       trans3.setFillAfter(true);
       trans3.setDuration(4000);
      this.findViewById(R.id.button3).startAnimation(trans3);
        Q.setOnClickListener(new OnClickListener() {
        public void onClick(View v) {
                        quit (false, null);
        }
        });}
        private void quit (boolean success, Intent i)
        {
                // On envoie un résultat qui va permettre de quitter l'appli
                setResult ((success)? Activity.RESULT_OK : Activity.RESULT_CANCELED, i);
                finish();
        }
                public boolean onCreateOptionsMenu(Menu menu) {
                // Inflate the menu; this adds items to the action bar if it is present.
                getMenuInflater().inflate(R.menu.main, menu);
                return true;
        }
}
```

– **le fichier XML (res/layou)**

```
<AbsoluteLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/Mima_Faty"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@drawable/image"
    tools:context=".Main" >

    <TextView
        android:id="@+id/textView1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_x="44dp"
        android:layout_y="14dp"
        android:text="Bienvenus au musée"
        android:textAppearance="?android:attr/textAppearanceMedium"
        android:textColor="#000000"
        android:textSize="25sp"
        android:textStyle="italic"
        android:typeface="serif" />

    <Button
        android:id="@+id/button2"
        android:layout_width="89dp"
        android:layout_height="wrap_content"
        android:layout_x="217dp"
        android:layout_y="401dp"
        android:background="@drawable/test"
        android:text="Lancer"
        android:textColor="#000000"
        android:textSize="20sp"
        android:textStyle="italic"
        android:typeface="serif" />

    <Button
        android:id="@+id/button3"
        android:layout_width="78dp"
        android:layout_height="wrap_content"
        android:layout_x="173dp"
        android:layout_y="444dp"
        android:background="@drawable/test"
        android:text="Quitter"
        android:textColor="#000000"
        android:textSize="20sp"
        android:textStyle="italic"
        android:typeface="serif" />

    <Button
        android:id="@+id/button1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_x="116dp"
        android:layout_y="401dp"
        android:background="@drawable/test"
        android:text="S&apos;inscrire"
        android:textColor="#000000"
        android:textSize="20sp"
        android:textStyle="italic"
        android:typeface="serif" />

</AbsoluteLayout>
```

– **Manifeste**

```
<activity android:name= Main></ activity >
```

3. **MySQL**

   (a) **bases de données**

   – **Description d'une base de données**

   Une base de données (son abréviation est BD, en anglais DB, database) est une entité dans laquelle il est possible de stocker des données de façon structurée et avec le moins de redondance possible. Ces données doivent pouvoir être utilisées par des programmes, par des utilisateurs différents. Ainsi, la notion de base de données est généralement couplée à celle de réseau, afin de pouvoir mettre en commun ces informations, d'où le nom de base.

   – **Le rôle d'une base de données**

   Une base de données permet de mettre des données à la disposition d'utilisateurs pour une consultation, une saisie ou bien une mise à jour, tout en s'assurant des droits accordés à ces derniers. Cela est d'autant plus utile que les données informatiques sont de plus en plus nombreuses. Une base de données peut être locale, c'est-à-dire utilisable sur une machine par un utilisateur, ou bien répartie, c'est-à-dire que les informations sont stockées sur des machines distantes et accessibles par réseau. L'avantage majeur de l'utilisation de bases de données est la possibilité de pouvoir être accédées par plusieurs utilisateurs simultanément.

   – **La gestion des bases de données**

   Afin de pouvoir contrôler les données ainsi que les utilisateurs, le besoin d'un système de gestion s'est vite fait ressentir. La gestion de la base de données se fait grâce à un système appelé SGBD (système de gestion de bases de données) ou en anglais DBMS (Database management system). Le SGBD est un ensemble de services (applications logicielles) permettant de gérer les bases de données, c'est-à-dire :

   – Permettre l'accès aux données de façon simple ;
   – Autoriser un accès aux informations à de multiples utilisateurs ;
   – Manipuler les données présentes dans la base de données (insertion, suppression, modification).

(b) **MySQL**

MySQL est un système de gestion de base de données Relationnelles (SGBDR) et basé sur un modèle client - serveur. Il fait partie des logiciels de gestion de base de données les plus utilisés au monde, Il utilise pour cela le langage SQL et autant par le grand public (applications web principalement) que par des professionnels, en concurrence avec Oracle et Microsoft SQL Server.

– **Création de la base de données**

Elle s'effectue en tapant la requête SQL suivante dans l'outil php-MyAdmin :

CREATE DATABASE ‘musee‘ DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci ; USE ‘musee‘ ;

Apres la création de la base de données on passe à la création des différentes tables qui la composent, la procédure étant la même pour l'ensemble des tables nous allons donner comme exemple la création de la tables **"visiteurs"** et **"cellules"** :

– **Structure de la table ‘cellules‘**

```
CREATE TABLE IF NOT EXISTS ‘cellules‘ (
‘numero_cellule‘ int(11) NOT NULL AUTO_INCREMENT,
‘x_min‘ double NOT NULL, ‘x_max‘ double NOT NULL,
‘y_min‘ double NOT NULL, ‘y_max‘ double NOT NULL,
‘numero_oeuvre‘ int(11) NOT NULL,
‘numero_salle‘ int(11) NOT NULL,
‘numero_dep‘ int(11) NOT NULL,
‘numero_etage‘ int(11) NOT NULL,
PRIMARY KEY (‘numero_cellule‘,‘numero_salle‘,‘numero_dep‘, ‘numero_etage‘),
KEY ‘numero_salle‘ (‘numero_salle‘),
KEY ‘numero_oeuvre‘ (‘numero_oeuvre‘),
KEY ‘numero_dep‘ (‘numero_dep‘,‘numero_etage‘))
ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=14 ;
```

– **Structure de la table ‘visiteurs‘**

```
CREATE TABLE IF NOT EXISTS ‘visiteurs‘ (
```

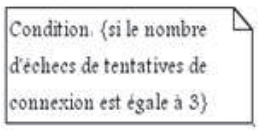
```
ùnumero_vs‘ int(11) NOT NULL AUTO_INCREMENT,
‘nom_vs‘ char(40) NOT NULL,
‘prenom_vs‘ char(40) NOT NULL,
‘date_nais‘ date NOT NULL,
‘ville_resid‘ char(20) NOT NULL,
‘profession‘ char(40) NOT NULL,
‘centre_interet‘ char(40) NOT NULL,
‘etat_sante‘ char(20) NOT NULL,
‘nbr_visites‘ int(11) NOT NULL,
‘numero_compte‘ int(11) NOT NULL,
PRIMARY KEY (‘numero_vs‘),
KEY ‘numero_compte‘ (‘numero_compte‘))
ENGINE=InnoDB DEFAULT CHARSET=utf8 AUTO_INCREMENT=4 ;
```

4. **Notations et terminologies des diagrammes d'UML utilisés**

Cette annexe fournit une liste non exhaustive des figures employées durant la conception pour la construction de diagrammes UML. Nous avons reçu 3 types de diagrammes :

- Diagramme de cas d'utilisation ;
- Diagramme de séquence.
- Diagramme d'activité ;
- Diagramme de classes ;
- Diagramme de déploiement.

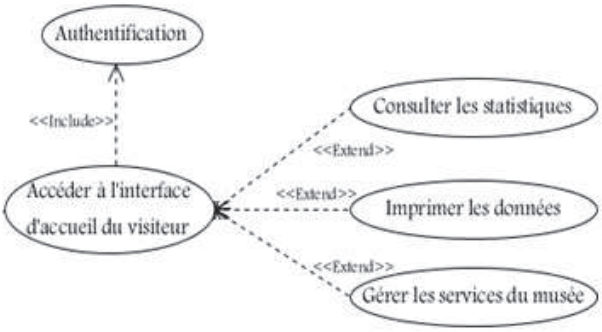
Note : élément contenant des remarques ajoutées à un diagramme donné.



Inclusion et Extension : flèche en tirets reliant deux cas d'utilisation. Elle est accompagnée d'un stéréotype qui définit la nature de la relation :

- **«include» :** Cas 2 est inclut dans Cas 2 (Cas 1 n'est réalisable que si Cas 2 est d'abord réalisé).

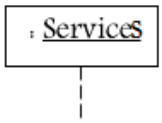
– **«extend» :** Cas 1étend Cas3 (Cas 1 ajoute des séquences de traitement à Cas 3, qui sont facultatives selon l'acteur relié à Cas 3 ou le scénario suivi).



Acteur : rôle joué par une entité externe (utilisateur humain, dispositif matériel ou autre système) qui interagit directement avec le système étudié.

Cas d'utilisation : représente un ensemble de séquences d'actions qui sont réalisées par le système et qui produisent un résultat observable intéressant pour un acteur particulier.

Ligne de vie : ligne verticale en pointillés sur laquelle sont représentées les actions d ?un participant dans un ordre chronologique. Tout participant d'un diagramme possède sa ligne de vie.



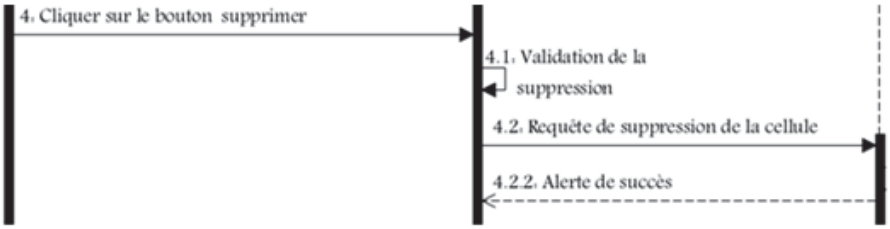
Barre d'activation : (facultative) barre positionnée sur une ligne, et qui signifie que le participant est occupé par une action.

Message : flèche horizontale qui sert à modéliser une communication entre deux participants à un instant donné. Chacune de ses deux extrémités est positionnée sur une ligne de vie. Un message possède un émetteur et un récepteur, un participant peut être à la fois émetteur et récepteur.

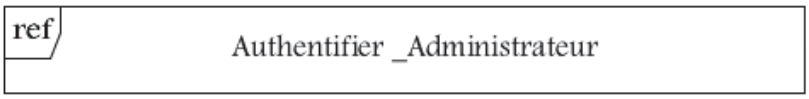
Signature : (facultative) texte avec le message spécifiant son nom et ses paramètres. message 4 (Cliquer sur le bouton supprimer) est une signature.

Message synchrone/asynchrone : les messages synchrones sont de la forme de message 4.2 tandis que les messages asynchrones sont de la forme de message 4

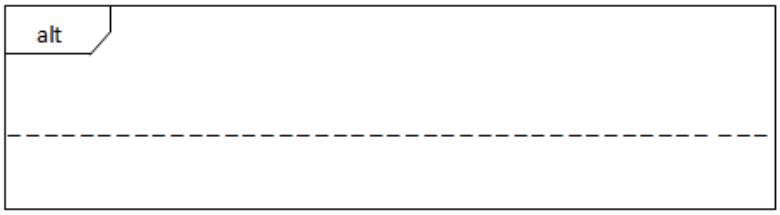
Message de retour : (facultatif) lorsqu'un participant appelle un autre en envoyant un message, l'appelé émet un message de retour à la fin de son action. Selon le schéma, le message de retour est modélisé par une flèche en pointillés.



Fragment de séquence : fragment qui fait référence à un autre diagramme de séquence. Il est modélisé par un bloc avec une étiquette "ref " et un nom identique au nom du diagramme de référence. Les participants dans un fragment de séquence sont les mêmes dans son diagramme de référence.



Opérateur " alt " : l'opérateur " alt " correspond à une instruction de test avec une ou plusieurs alternatives possibles. Il permet aussi d'utiliser les clauses de type sinon et se représente dans un fragment possédant au moins deux parties séparées par des pointillés.



Opérateur " loop " : l'opérateur " loop " correspond à une instruction de boucle qui permet d'exécuter une séquence d'interactions tant qu'une condition est satisfaite.

Bibliographie

[1] M.WEISER. *The computer for the 21st century*. Scientific American, September 1991.

[2] J.MERCADAL. *Approche langage au développement logiciel : application au domaine des systèmes d'informatique ubiquitaire*, these présentée à l'Université de Bordeaux, 10 octobre 2011.

[3] C.TACONET. *Intergiciels pour la sensibilité au contexte en environnement ubiquitaire*, Mémoire d'Habilitation à Diriger des Recherches en Informatique, Université d'evry-Val-d 'Essonne, jeudi 10 février 2011.

[4] M.SLOMAN, E.LUPU. *Engineering Policy-Based Ubiquitous Systems*, the computer journal, 2009.

[5] N.THANH BINH. *Gestion de profil apprenant distribue dans un environnement ubiquitaire*, mémoire de stage de fin d'etude master en informatique,Institut de la Francophonie pour l'informatique, Août, 2009.

[6] A.KUICEM. *Sensibilité au contexte*, Cour master 2 SCITIC, Université de Bejaia, 2012.

[7] A.KUICEM. *Composition dynamique de services en environnements ubiquitaires*, Mémoire de Magister En Informatique, Université Abderrahmane Mira de Bejaïa, 2008.

[8] S.GHANEM et Z.BOUANANI. *Gestion de Contexte et Découverte de Services Sensibles au Contexte dans les Environnement Ubiquitaires*. Mémoire de fin de Cycle pour obtention du master. Université A.Mira, Béjaïa.juin 2012.

[9] A.BEGDOURI, M.BERRAHO, R.AJHOUN. *L'informatique mobile au service des applications médicales*, Séminaire SIM'07, FMP de Fès, le 02 juin 2007.

[10] C.JANG, J.KIM, H.CHANG, E.CHOI, B.KIM, G.SOOLEE. *Method of Profile Storage for Improving Recommendation Accuracy on Ubiquitous Computing1*, Second International Conference on Future Generation Communication and Networking, Hainan Island,China, 13-15 December 2008.

[11] J.ZHU, L.GAO, X.ZHANG. *Preliminary Research on Wearable Healthcare in Ubiquitous Computing Age*, International Conference on Computer Science and Software Engineering,vol 3, pp519-523, China,12-14 December 2008.

[12] G.REY. *Systèmes interactifs sensibles au contexte*,dea d'informatique :systemes et communication, Université Joseph Fourier U.F.R Informatique et Mathématiques Appliquées, Institut National Polytechnique de Grenoble Ensimag, 19 juin 2001.

[13] G.PUJOLLE. *Les réseaux*,Eyrolles, 2006.

[14] L.LI, Y.ZHENG. *Research on Ubiquitous Learning Environment Design*, International Conference on Computer Science,vol5 , pp 1196-1196,China,12-14 December 2008.

[15] J.KRUMM. *Ubiquitous computing fundamentals*, Microsoft Corporation Redmond, Washington, U.S.A.CHAPMAN and HALL BOOK, 2010.

[16] J-A.BROTHERTON, G-D.ABOWD, K-N.TRUONG. *Supporting capture and access interfaces for informal and opportunistic meetings*, GVU Center et College of Computing Georgia Institute of Technology, Atlanta,USA.

[17] A.DEY, G-D.ABOWD et D.SALBER. *A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications*, Human-Computer Interaction, vol. 16, no 2-4, p. 97-166. 2001

[18] A.CHIBANI. *Intergiciel multi agents orienté web sémantique pour le développement d'applications ubiquitaires sensibles au contexte*, THESE DE DOCTORAT de l'Université Paris XII-Val de Marne, décembre 2006.

[19] MIRAOUI, Moeiz. *Architecture logicielle pour l'informatique diffuse : modélisation du contexte et adaptation dynamique des services* ,Thése de doctorat,école technologie supérieure université du Québec,JUILLET 2009.

[20] Grand Dictionnaire : Office québécois de la langue française En ligne, «http ://www.granddictionnaire.com/», Consulté le 20Mai, 2008.

[21] F.SEBBAK. *Architecture intergicielle semantique basé sur le standard OSGI pour les services robotiques*, Mémoire de Magistère en Informatique, Université Abderrahmane Mira de Béjaïa, 2007.

[22] B-N.SHILIT et M.THEIMER. *Disseminating Active Map Information to Mobile Hosts*. Network, IEEE, Vol.8, N°5, pp. 22-32, September/October 1994.

[23] B.SCHILIT, N.ADAMS et R.WANT. *Contextaware computing applications* . Dans In Proceedings of the Workshop on Mobile Computing Systems and Applications, pages 8590. IEEE Computer Society, 1994.

[24] J.BROWN, D.BOVEY et X.CHEN.*Context-aware applications : from the laboratory to the marketplace* , IEEE Personal Communications, vol. 4, no 5, p. 58-64,(October 1997)

[25] N.RYAN, J.PASCOE et D.MORSE *Enhanced Reality Fieldwork : The Context-awareArchaeological Assistant* , In Computer Applications in Archaeology, Oxford, UK.1997

[26] G.CHEN et D.KORT.*A Survey of Context-Aware Mobile Computing Research*,TR2000-381, Dartmouth : Dartmouth College Computer Science,2000.

[27] A-K.DEY. *Understanding and using context* , Personal and ubiquitous computing,vol. 5, p. 4-7.2001.

[28] T.CHAARI. *Adaptation d'applications pervasives dans des environnements multi-contextes*, thèse de doctorat. L'institut national des sciences appliquées de Lyon. 28/09/2007.

[29] P.BREZILLON, M-R.BORGES, J-A.PINO et J-Ch.POMEROL. *Context-Awareness in Group Work : Three Case Studies* . IFIP Int. Conf. on Decision Support Systems(DSS 2004). Prato, Italie, Juillet 2004.

[30] N.BELHANAFI BEHLOULI. *Ajout de mécanismes de réactivité au contexte dans les intergiciels pour composants dans le cadre d'utilisateurs nomades*. Thèse présentée pour l'obtention du grade de Docteur de l'Institut National des Télécommunications. Université d'evry Val d'Essonne. 27 Novembre 2006.

[31] J-P MULLER.*Ontologie et modélisation conceptuelle*.IMAS,fevrier2009.

[32] B.SCHILIT, N.ADAMS, N., et R.WANT. *Context-Aware Computing Applications*, In-Proc. of the Workshop on Mobile Computing Systems and Applications (Santa Cruz, CA, Dec.1994), pp. 85-90.

[33] P-J BROWS, . 1995. *The stick-e document : a framework for creating context-aware applications* . Electronic Publishing Origination, Dissemination and Design, vol. 8, no 2 (JUNE AND SEPTEMBER 1995), p. 259-272.

[34] A-K.DEY. *Understanding and Using Context*. Personal and Ubiquitous Computing,5(1) :4-7, February 2001.

[35] http ://cobra.umbc.edu.

[36] G-K.MOSTEFAOUI, J.PASQUIER-ROCHA ET P.BRIZILLON. *Context-Aware Computing : A Guide for the Pervasive Computing Community.* In ICPS : IEEE International Conference on Pervasive Services, pages 39-48, 2004.

[37] A.ACHILLOS,K.YANG, N.GEORGALAS. *Context modelling and a context-aware framework for pervasive service creation : A model-driven approach*, Pervasive Computing et mobile, B.V. 2010.

[38] B.SHILIT, N.ADAMS et R.WANT. 1994. *Context-Aware Computing Applications* . In Proceedings of the Workshop on Mobile Computing Systems and Applications, Santa Cruz, CA, (Decenber 1994). p. 85-90. Santa Cruz, CA, : IEEE Computer Society.

[39] A.HELD, S.BUCHHOLZ et A.SCHILL. 2002. *Modeling of context information for pervasive computing applications* . In the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI '02) (July 2002). Orlando, Fla, USA.

[40] H.KAMMANAHALLI, S.GOPALAN, V.SRIDHAR et K.RAMAMRITHAM. 2004. *Context-aware retrieval in web-based collaborations* . In PerCom Workshops, sous la dir. De Society, IEEE Computer. p. 8-12. Orlando, FL, USA.

[41] T.KIM CUONG. *Mobilité, Contexte et Adaptation dans un système informatique éducatif.* Mémoire de fin d'etude master d'informatique,Institut de la Francophonie pour l'informatique,août 2007.

[42] K.HENRICKSEN,J.INDULSKA,A.RAKOTONIRAINY, *Modeling context information in pervasive computing systems*, School of Information computing Technology and Electrical Engineering, The yniversity of Queensland St Lucia QLD 4072 Australia, 2002.

[43] K.DEY, D.SALBER, M.FUTAKAWA et D.Abowd. *An Architecture To Support Context-Aware Applications*, GVU Technical Report GIT-GVU-99-23. Submitted to the 12th Annual ACM Symposium on User Interface Software and Technology (UIST 99), June 1999.

[44] H.CHEN, T.FININ et A.JOSHI. *A Context Broker for Building Smart Meeting Rooms* . Proceedings of the Knowledge Repre- sentation and Ontology for Autonomous Systems Symposium, 2004 AAAI Spring Symposium, pages 5360. (URL http ://ebiquity.umbc.edu/paper/ html/id/143/).

[45] S.ZAIDENBERG. *Apprentissage par renforcement de modeles de contexte pour l'informatique ambiante*, THESE de doctorat,octobre 2009.

[46] P.KORPIPAA, J.MANTYJARVI, J.KELA, H.KERANEN et E-J.MALM. 2003. *Managing context information in mobile devices*. IEEE Pervasive Computing, vol. 2, no 3 , p. 42-51,(July- September, 2003).

[47] J-M.PIEPLU. *GPS et Galileo Systèmes de navigation par satellites*.

[48] https ://en.wikipedia.org/wiki/Mobile_device.

[49] http ://www.epfl.ch/mobile.

[50] http ://www.gsmarena.com/nokia_9500-678.php.

[51] http ://www.gsmarena.com/ samsung_galaxy_grand_i9082-5163.php.

[52] http ://www.gsmarena.com/ hp_touchpad-3839.php.

[53] http ://www.techopedia.com/definition/2953/mobile-applicationmobileapp.

[54] N.TIEN THINH. *Système d'exploitation pour les mobiles*, Rapport final (Travail personnel Encadré), Hanoi, Juillet 2009.

[55] http ://fr.wikipedia.org/wiki/IOS(Apple), http ://www.apple.com/ca/fr/ios/.

[56] http ://fr.wikipedia.org/w/index.phptitle=BlackBerryOSetoldid=88870706.

[57] http ://www.gsmarena.com/ imate PDA 2.php.

[58] http ://www.physorg.com.

[59] fr.wikipedia.org/wiki/plateforme_(informatique).

[60] Fr.wikipédia.org/wiki/ système_d'exploitation_mobile.

[61] http ://www.iphonedeveloppeur.fr/2011/04/05/fonctionnementduneapplicationiphone/

[62] Application mobile web ou natif_ekito people.htm.

[63] http ://www.boreal-business.net/application/iphone/depannemoi.html.

[64] https ://fr.wikipedia.org/wiki/Firefox_Mobile.

[65] http ://www.definitionswebmarketing.com.

[66] https ://fr.wikipedia.org/wiki/Google_Maps.

[67] M.WINTER. *Méthode de développement "RUP"*, Cour master 2 MIAGE, Université Nice, 2012.

[68] P.ROQUES. *UML 2 Modéliser une application Web*, Eyrolles, 2007.

[69] P.ROQUES. *UML 2 modéliser une application web (les Cahiers du Programmeur)*, Eyrolles, 2008.

[70] J.GABAYet D.GABAY. *UML 2 analyse et conception*, DUNOD, 2008.

[71] P.ROQUES et F.VALLEE. *UML 2 en action- De l'analyse des besoins à la conception* , Eyrolles, 2007.

[72] F.Barbier, UML 2 et MDE - Ingénierie des modèles avec études de cas, DUNOD, 2005.

[73] I.JACOBSON, G.BOOCH et J.RUMBAUGH. *Le Processus unifié de développement logiciel*, Eyrolles, 2000.

[74] P.GERARD. *Processus de Développement Logiciel Cours M1*, 2007/2008.

[75] P.B.KRUCHTEN.*The 4+1 View Model* , IEEE Software Novembre 1995.

[76] P-A.MULLE et N.GAERTNER. *Modélisation objet avec UML*, Eyrolles, 2003.

[77] O.CAPUOZZ. *Cas d'utilisation,une introduction.* Mars 2004.

[78] http ://fr.wikipedia.org/wiki/Mod

[79] K.HAMILTON et R.MILES. *Learning UML 2.0, O'Reilly*, 2006.

[80] L. AUDIBERT. *UML 2 - de l'apprentissage à la pratique*, Ellipses 2009.

[81] P.ROQUES.*UML 2par la pratique - étude de cas et exercices corrigés*, Eyrolles, 2006.

[82] F-C.EDGAR. *A Relational Model of Data for Large Shared Data Banks.* ACM,Vol.13, N°6, pp377-386,June 1970.

[83] E.PUYBARET. *Les cahiers du programmeur.* java 1.4 et 5.0 (3e édition), EYROLLES, 2004.

[84] J-M.DOUDOUD. *Développer en java (version 1.10).* Aout 2009.

[85] P.RIGAUX. *Pratique de MySQL et PHP.*Paris,DUNOD,2009

[86] C.SOUTOU. *Apprendre SQL avec MySQL Avec 40 exercice corrigés.* Paris, groupe Eyrolles, 2006.

[87] I.KIBER. *Formation au langage PHPMySQL.* Tunisie,juillet 2005.

[88] T.CONVERSE, J.PARK et C.MORGAN. *PHP5 and MySQL Bible.*Indianapolis, 2004.

[89] CYRILLE HERBY. *Apprenez à programmer en java :la programmation professionnelle à la portée de tous.*(www.siteduzero.com).

[90] http ://siguillaume.developpez.com/tutoriels/apache/installationconfiguration-serveurweb-apache/ ?page=introduction.

[91] http ://edu.ca.edu/IMG/pdf/adminlinux.pdf.

[92] http ://www.openhandsetaliance.com/android_overview.html

[93] http ://www.phpmyadmin.net/home_page/index.php

[94] http ://www.Courses.Coresevlets.com

[95] http ://www.Courses.Coresevlets.com

Résumé

L'informatique ubiquitaire, dit aussi pervasive, adresse une vision d'un monde numérique assistant des individus dans leurs activités de tous les jours sans être intrusifs. Cela fait appel à un concept clé dans les systèmes ubiquitaire : le contexte, qui exige une bonne compréhension et utilisation. Le défi est alors de pouvoir proposer des applications qui s'adaptent tant aux souhaits des utilisateurs qu'à l'environnement physique.

Dans ce travail, nous avons développé une application mobile sensible au contexte d'un visiteur au sein d'un musée. Cette dernière permet une adaptation des services du musée aux souhaits de l'utilisateur en se basant sur la gestion de son contexte (profil et position).

Le mémoire contient une partie théorique sur les systèmes ubiquitaires, l'introduction au contexte et les applications mobiles puis une partie pratique (conception et réalisation).

Nous avons opté pour l'approche UML pour décrire les principales fonctionnalités de notre système, et nous nous sommes appuyés sur la méthode RUP qui offre l'avantage de construire un système de manière itérative. PHPMyAdmin été l'outil qu'on a utilisé pour la création de la base de données.

Nous avons utilisé la plateforme Android pour la réalisation de l'application mobile. Les autres parties du système sont développées en utilisant différents langages et environnements de développements à l'exemple : java (Eclipse), PHPMyAdmin ,etc.

Mots-clés : Informatique ubiquitaire(diffuse), contexte, sensibilité au contexte, adaptation de services, Android, java, PHPMyAdmin.

Abstract

Ubiquitous computing, also known as pervasive, addresses a vision of digital world which assists individuals in their daily activities without being intrusive. It uses a key concept in ubiquitous systems : context, which requires a good understanding and use. So nowadays the challenge is to be able to offer applications that suit both the users' needs and the physical environment.

In our work, we have developed a context-aware mobile application for a visitor in a museum. It allows an adaptation of museum services to the wishes of the user based on the management of her/his context (profile and position).

This research contains a theory part about ubiquitous computing, introduction to the context and mobiles applications, then practical part (conception and implementation).

For the design of our application we have chosen UML, following an incremental and iterative Process (RUP). PHPMyAdmin tool has been used for creating the database.

We have used the Android platform for the realization of the mobile application. For the others part of the system, we have used different languages and environment, for instance : java (eclipse), phpMyAdmin, and so on.

Keywords :ubiquitous computing (pervasif),context, context-awarness and services adapation, Android, JAVA, PHPMyAdmi.