REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE

MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE MOULOUD MAMMERI TIZI-OUZOU

FACULTE DE GENIE ELECTRIQUE ET D'INFORMATIQUE

DEPARTEMENT D'INFORMATIQUE

MEMOIRE

En vue de l'obtention du diplôme de master en INFORMATIQUE

Thème :

Conception et réalisation d'une application Android pour la gestion d'un service de restauration

Spécialité : Système informatique

Proposé par :

Mr Taleb Hakim

Dirigé par :

Mr SADOU Samir

Réalisé par :

Mr ICHALLALENE Toufik

Promotion : 2014/2015

REMERCIEMENTS

Tout d'abord je tiens à remercier mon promoteur monsieur SADOU Samir d'avoir dirigé mon mini projet avec une constante disponibilité tout au long de l'année universitaire comme je tiens à remercier mes amis

Mes remerciements vont également à tous ceux qui ont participés de prés ou de loin à la concrétisation de mon modeste projet.

Enfin, je remercie les membres de jury qui m'ont fait l'honneur de juger mon travail.

Dédicaces

A la mémoire de mon cher promoteur Taleb Hakim

A mes très chers parents

A mes frères

A tous mes amis (es)

Ka Rim, Bylka ,Brahim,Mekhelouf

Et Dabi Ould

Sommaire

Chapitre II : La technologie Android Introduction :

Chapitre III : Analyse et conception

Chapitre IV : Réalisation

Liste des figures

Introduction générale

Depuis qu'il a fait son apparition dans les années 50, le téléphone mobile n'a pas cessé d'évoluer, passant d'une utilisation très limitée à une utilisation aisée et accessible à tous, devenant ainsi un véritable compagnon de l'homme moderne. Longtemps les téléphones mobiles ont été limités à des affichages simplistes et une connectivité limitée. De plus, L'évolution technologique fait, qu'aujourd'hui des téléphones plus puissants, avec plus de mémoire et un meilleur affichage apparaissent et ce, pour des coûts raisonnables. On voit actuellement des téléphones qui intègrent des applications. De plus, leur connectivité réseau a été améliorée, en intégrant des supports de technologie. La plupart des téléphones actuels intègrent en standard un environnement d'exécution Android, leur conférant ainsi la possibilité d'exécuter des applications écrites pour la plate-forme Android. Cette plate-forme a été proposée par la société Android à destination des équipements mobiles et embarqués, caractérisés par de fortes contraintes matérielles (taille mémoire, vitesse du processeur, etc.). Toutes ces caractéristiques des téléphones modernes nous permettent d'étendre leurs fonctionnalités. C'est dans cette optique que s'inscrit notre travail, qui consiste à développer une application Android pour la gestion d'un service de restauration. Pour l'organisation de notre travail, nous avons adopté la structure suivante :

- Le premier chapitre intitulé **‹‹ Architecture Client/serveur, Généralités sur les réseaux et Web ››**, donnera un aperçu général sur les réseaux sans fil.
- Le second chapitre intitulé **‹‹ Technologie Android ››**, présentera en détail cette technologie.
- Le troisième chapitre intitulé **‹‹ Analyse et conception ››**, sera consacrée à l'analyse et la conception de notre application, en utilisant le langage UML.
- Le quatrième chapitre intitulé **‹‹ Réalisation ››**, présentera les différents outils utilisés pour le développement de notre application, suivis de quelques captures d'écran illustrant son fonctionnement.
- Enfin, une conclusion générale dans laquelle sont récapitulés les points clés de notre travail, les difficultés rencontrées et les améliorations.

Chapitre 1

Architecture Client/serveur, Généralités sur les réseaux et Web.

Introduction

Le monde des télécommunications est en pleine évolution depuis des années. Jusqu'à une époque récente, la transmission des informations se faisait par la voix ou l'écrit essentiellement à l'aide du téléphone ou du télex. Aujourd'hui, l'utilisation des réseaux spécialisés permet d'acheminer tous les types d'information : sons, images, etc.

Dans le début des années 90, les terminaux passifs se bousculent sur le même bureau en même temps que les bases de données se multiplient dans l'entreprise. Le remède à cette explosion incontrôlée des applications verticales s'appelle alors le « Client/serveur».L'architecture Client-serveur reste vécue comme une nécessité pour partager les ressources de la station serveur.

I. L'architecture CLIENT/SERVEUR :

I.1 Définition :

C'est un modèle informatique basé sur le traitement distribué selon le quel un utilisateur lance un logiciel client à partir d'un ordinateur relié à un réseau, déclenchant simultanément le lancement d'un logiciel serveur situé dans un autre ordinateur possédant les ressources souhaitées par le client.

I.2 Présentation de l'architecture d'un système client/serveur :

De nombreuses applications fonctionnent selon un environnement client/serveur, cela signifie que les machines liées (des machines faisant parties du réseau) contactent un serveur, une machine généralement très puissante en termes de capacité d'entrée-sortie, qui leur fournit des services.

Les services sont exploités par des programmes, appelés programme clients, s'exécutant sur des machines clientes. On parle ainsi du client FTP, client de messageries,…, lorsqu'on désigne un programme, tournant sur une machine cliente, capable de traiter des informations qu'il récupère auprès du serveur. (Dans le cas du client FTP il s'agit de fichiers tandis que pour le client messagerie il s'agit du courrier électronique).Dans un environnement purement client/serveur, les ordinateurs du réseau (les clients) ne peuvent contacter que le serveur, c'est l'un des principaux atouts de ce modèle (chaque machine est soit client, soit serveur).

I.3 Notions de bases :

- **Client :** c'est le processus demandant l'exécution d'une opération à un processus par envoi d'un message contenant le descriptif de l'opération à exécuter et attendant la réponse à cette opération par un message en retour
- **Serveur :** c'est un processus accomplissant une opération sur demande d'un client.
- **Requête :** c'est un message transmis par un client à un serveur décrivant l'opération pour le compte d'un client.

- **Réponse :** c'est un message transmis par un serveur à un client suite à l'exécution d'une opération contenant des paramètres de retour de l'opération.
- **Middleware :** c'est le logiciel, qui au milieu assure les dialogues entre les clients et les serveurs souvent hétérogènes.

I.4 Fonctionnement d'un système client/serveur :

Un système client/serveur fonctionne comme suit :

1. Le client émet une requête vers un serveur grâce à son adresse et son port, qui désigne un service particulier du serveur.

2 Le serveur reçoit la demande et répond à l'aide de l'adresse de la machine client et son port.

I.5 Les architectures client/serveur :

➢ **Architecture à deux niveaux :**

C'est l'architecture la plus couramment utilisée pour assurer la communication entre un serveur et un client. Dans cette architecture, le client communique directement avec le serveur. On distingue deux approches :

La première approche : on ne sépare pas l'interface utilisateur et les traitements, ces derniers sont assurés par le client, les données sont gérées par le serveur comme par exemple un serveur de base de données MySQL. Les traitements de gestions des données sont liés à l'interface utilisateur.

La deuxième approche : on sépare les traitements et les données de l'interface utilisateur, les données et traitements étant assurés totalement par le serveur, le client ne s'occupe que de l'interface utilisateur.

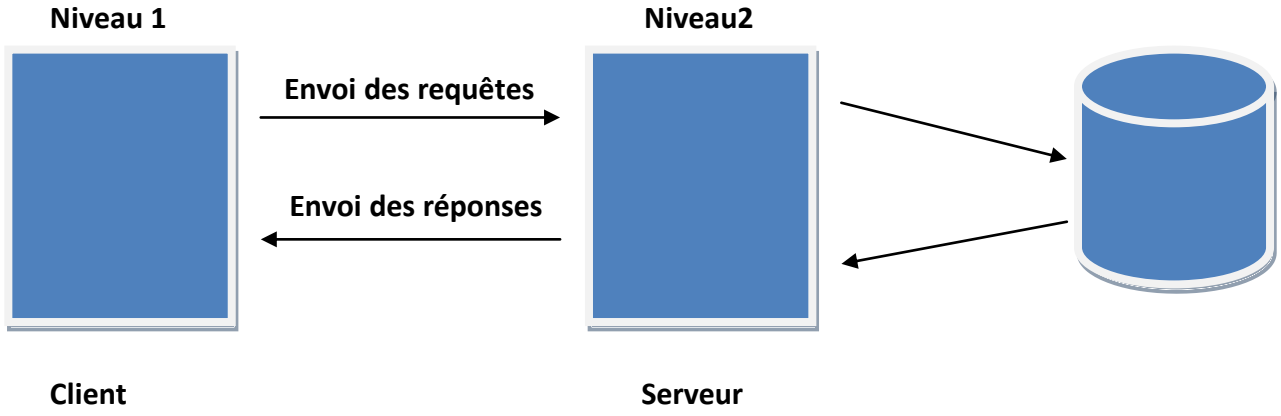


Figure I.1: Architecture Client/serveur à deux niveaux

➢ **Avantages et inconvénients de l'architecture à deux niveaux :**

L'architecture à deux niveaux est facile à mettre en œuvre et évite d'installer le gestionnaire de base de données sur chaque machine cliente mais présente certaines limites :

- Les applications clientes a distance sont lourdes (première approche).
- L'exploitation non optimale des machines clientes (deuxième approche).
- Cette architecture est limitée aux petites applications.

➢ **Architecture à trois niveaux :**

L'architecture à trois niveaux (appelées aussi architecture à 3- tiers) est la plus raffiné, elle est définie par trois niveaux.

- Le premier niveau est constitué du poste client doté d'un navigateur pouvant télécharger et exécuter du code mais seuls les aspects logiques de présentation sont effectivement fournis par le client ; Autrement dit le premier niveau s'occupe de l'interface utilisateur.
- Le second niveau fournit un service intermédiaire, un serveur de transaction ou d'application, il fait appel à un autre serveur.
- Le troisième est un serveur de bases de données corporatives, en général, un système de gestion de bases de données (SGBD).

La partie original de cette architecture se situe au niveau intermédiaire, le serveur de traitement, parfois appelé serveur de transaction ou Middleware.

On peut schématiser cette architecture comme suit :

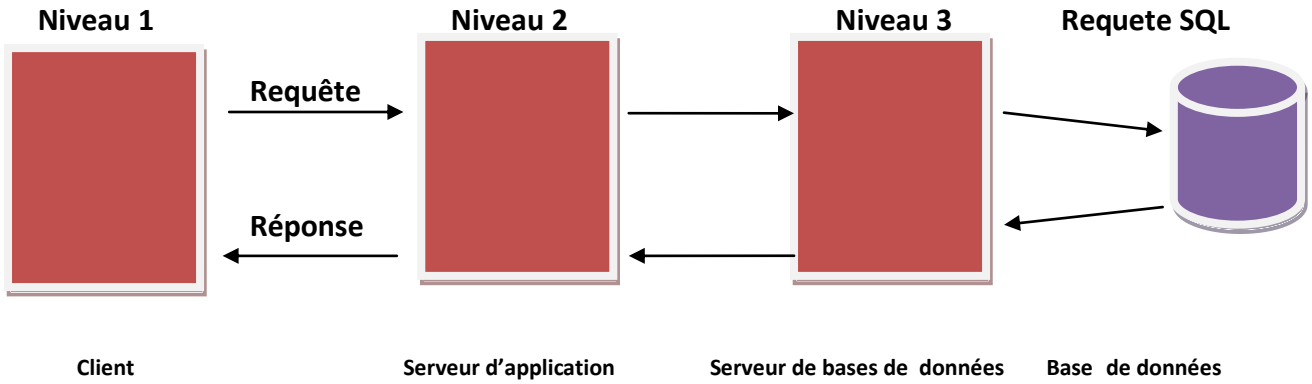


Figure I.2: Architecture Client/serveur à trois niveaux

- **Avantages de l'architecture à trois niveaux :**

- une plus grande flexibilité /souplesse.
- Une plus grande sécurité (la sécurité peut être définie pour chaque service).
- Des meilleures performances (les taches sont partagées).

- **L'architecture multi-niveaux :**

C'est une architecture ou l'on dispose de plusieurs serveurs (minimum deux), chacun de ces derniers effectue une tache spécialisée et peut utiliser les services d'un ou plusieurs autres serveurs afin de fournir son propre service.

I.6 Avantages de l'architecture client/serveur :

Le modèle client/serveur est particulièrement recommandé pour les réseaux nécessitant un grand niveau de fiabilité. Ses principaux atouts sont :

- **Des ressources centralisées :** étant donne que le serveur au centre de réseau, il peut gérer des ressources communes a tous les utilisateurs, comme une base de données centralisée, afin d'éviter les problèmes de redondance et de contradiction.
- **Une meilleure sécurité** : car le nombre de points d'entrée permettant l'accès aux données est moins important.

- **Une administration au niveau serveur** : les clients ayant peu d'importance dans ce modèle, ils ont moins besoin d'être administrés.
- **Un réseau évolutif :** grâce à cette architecture on peut supprimer ou rajouter des clients sans perturber le fonctionnement du réseau et sans modification majeures.

I.7 Inconvénients du modèle client/serveur :

L'architecture client/serveur a tout de même quelques lacunes parmi lesquelles, on cite :

- Un cout élevé du a la technicité du serveur.
- Le serveur est considéré comme étant le maillon faible du réseau client/serveur, étant donne que tout le réseau est architecturé autour de lui.

II. Généralités sur les réseaux informatiques

II.1 Définition d'un réseau

Un **réseau** est un ensemble d'ordinateurs reliés ensemble par des canaux électroniques de communication, qui leur permettent de partager des ressources: données ou périphériques (imprimante, connexion Internet, scanner, ...).

II.2 Le modèle OSI

Est un modèle définie par l'ISO (International Standardization Organisation) par opposition aux systèmes propriétaires, il permet l'interconnexion des systèmes hétérogènes et issus de constructeurs différents. Ce modèle est hiérarchisé en sept couches destiné à normaliser les échanges entre machines, chaque couche dispose de fonctionnalité qui lui son propre et fournit des services aux couches immédiatement adjacentes.

II.3. Les couche du modèle OSI

a. La couche physique

La couche physique s'occupe de la transmission des bits de façon brute sur un canal de communication. Concrètement, cette couche doit normaliser les caractéristiques électriques (un bit 1 doit être représenté par une tension de 5 V, par exemple), les caractéristiques mécaniques (forme des connecteurs, de la topologie...), les caractéristiques fonctionnelles des circuits de données et les procédures d'établissement, de maintien et de libération du circuit de données. À ce niveau, on s'intéresse à l'amplitude du signal, à la durée d'un bit, à la possibilité de transmettre simultanément dans les deux sens, à l'établissement et la libération du canal de connexion.

L'unité de donnée est le bit.

b. La couche liaison

La couche liaison de données est en charge d'encoder les données pour qu'elles soient transportables par la couche physique et doit être capable de renvoyer une trame lorsqu'il y a eu un problème sur la ligne de transmission. De manière générale, un rôle important de cette couche est la détection et la correction d'erreurs intervenues sur la couche physique.

Cette couche intègre également une fonction de contrôle de flux pour éviter l'engorgement du récepteur. Grâce à des mécanismes de régulation pour éviter la saturation du canal de communication par un émetteur unique.

L'unité de donnée est les trames.

c. La couche réseaux

En charge du transport des paquets en utilisant des informations d'adressage.

L'une des principales fonctions exercées dans la couche réseau est le routage des paquets, c'est une méthode d'acheminement des paquets à la bonne destination à travers le réseau.

L'unité d'information de la couche réseau est le paquet.

d. La couche transport

Cette couche est responsable du bon acheminement des messages complets au destinataire. Le rôle principal de la couche transport est de prendre les messages de la couche session, de les découper s'il le faut en unités plus petites et de les passer à la couche réseau, tout en

s'assurant que les morceaux arrivent correctement de l'autre côté. Cette couche effectue donc aussi le réassemblage du message à la réception des morceaux.
Un des rôles qu'il faut évoquer aussi est le contrôle de flux.
L'unité d'information de la couche transport est le message.

e. La couche session :

Elle permet l'ouverture et la fermeture d'une session de travail entre les utilisateurs, elle offre deux principaux services : Le premier est la gestion du dialogue (du jeton) : en fait elle commande le dialogue entre deux programmes d'applications (autorisation d'émission…).Le deuxième est la synchronisation : cette technique consiste à insérer des éléments tests dans le flot de données de manière à ne pas devoir reprendre la totalité d'une opération en cas d'échec.

f. La couche Présentation

Lors de la couche de présentation du modèle OSI, les données étant transmises sont traduites. Elle se traduit par les formats de chaque ordinateur à un format commun de transfert qui peuvent être interprétées par chaque ordinateur.

g. La couche application

Cette couche est le point de contact (interface) entre l'utilisateur et le réseau. On y trouve toutes les applications connues : transfert de fichiers, courrier électronique, Web, multimédia, etc…

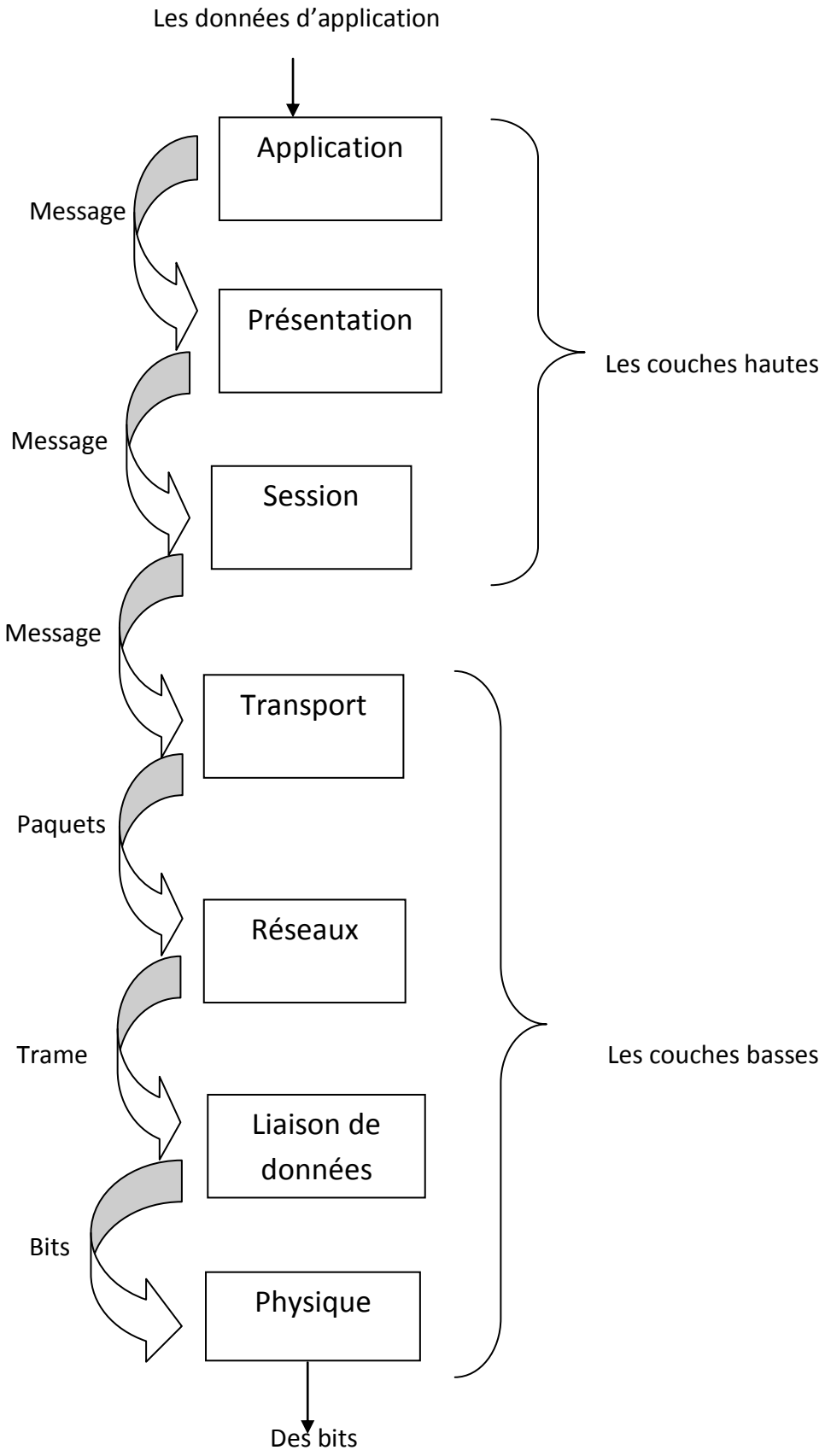


Figure I.3 : Représentation graphique du modèle OSI.

II.4 La transmission des données par le modèle OSI :

Le processus émetteur doit émettre des données vers le processus récepteur, il remet ces données à la couche application qui leur ajoute un en-tête qui contient : l'adresse source, l'adresse de destination, des informations nécessaire pour la synchronisation de la transmission des données, des bits de contrôle,....Puis transmet le bloc résultant à la couche présentation, cette dernière va ajouter un autre en-tête et expédie le bloc à la couche session. Ce processus est répété jusqu'à ce que les données atteignent la couche physique où elles sont transmises à la machine émettrice.

Sur cette machine, les différents en-têtes sont éliminés un à un lorsque le message remonte les couches jusqu'à arriver à la couche application du processus récepteur.

II.5. Le modèle TCP/IP:

TCP/IP désigne une architecture réseau, mais cet acronyme désigne en fait 2 protocoles étroitement liés : un protocole de transport TCP (Transmission Control Protocol) qu'on utilise pardessus un protocole réseau IP (Internet Protocol). Ce qu'on entend par "modèle TCP/IP" c'est en fait une architecture réseau en 4 couches dans laquelle les protocoles TCP et IP jouent un rôle prédominant.

Le modèle TCP/IP est apparu au début des années 70 (10 ans avant OSI), issue d'un organisme d'état américain le DOD (Department Of Defense) il s'agit des protocoles de communication et d'application les plus populaires pour connecter des systèmes hétérogènes indépendamment de la couche physique.

II.6. Les couche du modèle TCP/IP :

a. La couche "accès réseau" :

Elle regroupe les couches physiques et liaison de données du modèle OSI. En fait, cette couche permet à un hôte d'envoyer des paquets IP sur le réseau. L'implémentation de la couche accès réseau est typique de la technologie utilisée sur le réseau local.

Elle assure l'acheminement des données sur le réseau, la conversion des signaux, le control des erreurs…

L'unité de donnée de cette couche est : la trame.

b. La couche internet :

La couche Internet assure la fragmentation des segments TCP en datagrammes (paquets de données) puis leurs acheminements vers des machines distantes en empruntant des chemins différents (la commutation de paquets), ainsi que de la gestion de leur assemblage à la réception.

La couche internet utilise plusieurs protocoles, les plus importants sont : IP (Internet Protocol), ARP (), ICMP ().

L'unité de donnée de cette couche est : Le datagramme.

c. La couche transport:

Elle permet de segmenter (ou de réassembler) plusieurs applications de couche supérieure puis assure leurs transport.

La couche Transport fournit 2 protocoles : TCP (Transfert Control Protocol) et UDP (User (Datagram Protocol).

L'unité de donnée de la couche transport est : Le segment.

d. La couche application :

Elle englobe les couches session et présentation du modèle OSI et contient tous les protocoles de haut niveau, comme TFTP (Trivial File Transfer Protocol), SMTP (Simple Mail Transfer Protocol),

HTTP (HyperText Transfer Protocol). Le point important pour cette couche est le choix du protocole de transport à utiliser. Par exemple, TFTP (surtout utilisé sur réseaux locaux) utilisera UDP. A l'inverse, SMTP utilise TCP.

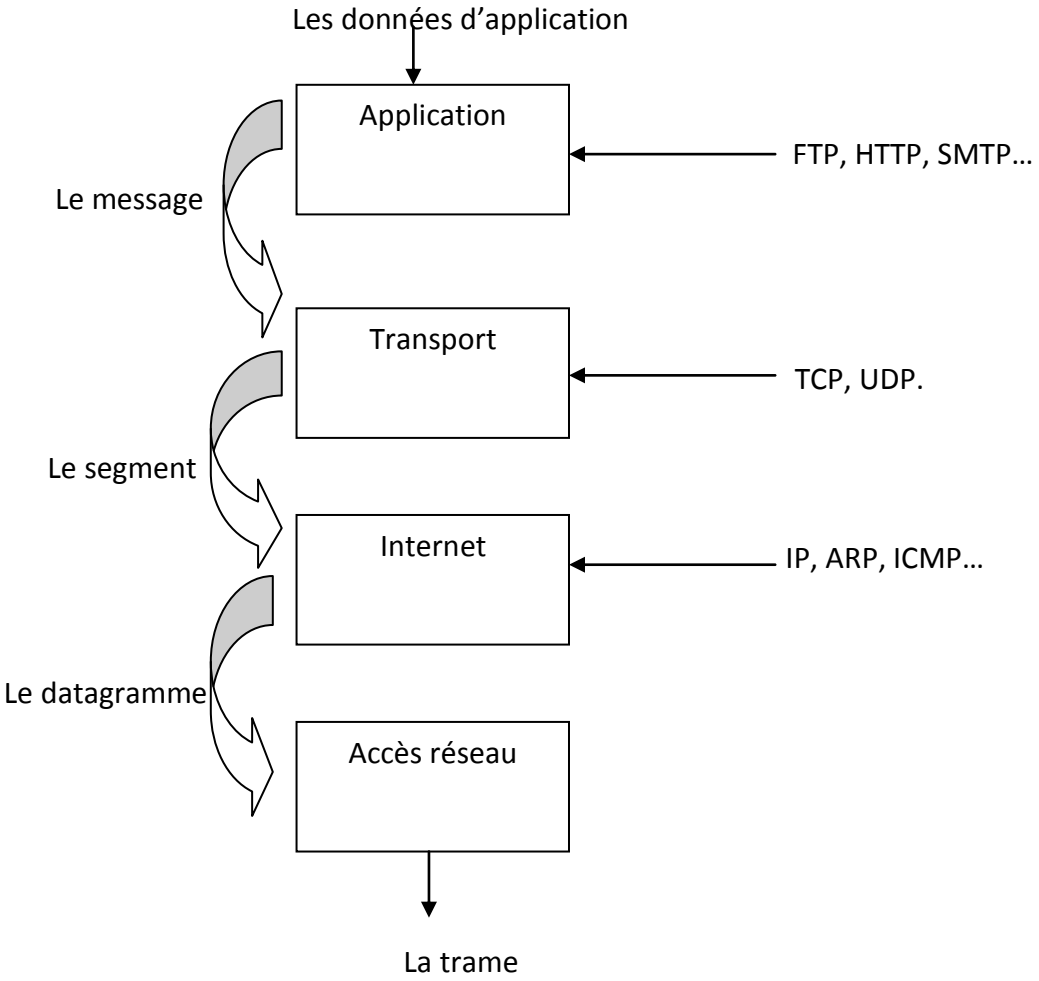


Figure I.4 : Représentation graphique du modèle TCP/IP.

II.7 L'adressage IP:

Chaque ordinateur du réseau dispose d'une adresse IP unique codée sur 32 bits. Une adresse IP est toujours représentée dans une notation décimale pointée constituée de 4 membres (1 par octet) compris chacun entre 0 et 255 et séparés par un point.

II.8 Les protocoles de la famille TCP/IP :

Les protocoles les plus importants de la famille TCP/IP sont :

• **TCP (Transmission Control Protocol) :** littéralement « protocole de contrôle de transmissions » est un protocole orienté connexion (il vérifie la bonne transmission de données par des signaux d'accusés de réception (acquittements) du destinataire) et fiable, il assure ainsi le contrôle des erreurs. Ce protocole présente l'avantage de garantir la transmission des segments.

TCP permet d'effectuer une tâche importante: le multiplexage/démultiplexage, c'est-à-dire faire transiter sur une même ligne des données provenant d'applications diverses ou en d'autres mots mettre en série des informations arrivant en parallèle.

• **UDP (User Datagram Protocol)** est non orienté connexion et non fiable, il n'assure aucun contrôle de transmission des données. Bien que chargé de la transmission des messages, il n'exécute aucune vérification sur l'acheminement des segments au niveau de cette couche.

L'avantage de ce protocole est sa vitesse. Comme il ne fournit pas d'accusés de réception, le trafic sur le réseau est plus faible, ce qui accélère les transferts.

• **IP (Internet Protocol) :** littéralement "le protocole d'Internet", permet le routage et l'adressage des paquets entre hôtes, ne gère pas les erreurs (paquets perdus, dupliquée, retard de transmission..). En effet chaque paquet envoyé est enveloppé par IP qui ajoute différentes informations comme l'adresse de l'expéditeur, celle du destinataire et des données supplémentaire pour bien contrôler l'acheminement des paquets.

• **ARP (Adresse Resolution Protocol)** : Ce protocole gère les adresses des cartes réseaux et la correspondance avec l'adresse IP. Chaque carte a sa propre adresse MAC d'identification codée sur 48 bits.

• **DNS (Domain Name System):** Chaque ordinateur est repéré à l'aide d'une adresse IP, mais on ne peut pas retenir toutes les adresses d'où l'utilité du protocole DNS qui associe à chaque adresse IP un nom symbolique qui sera utilisé à sa place.

• **ICMP** (Internet Control Message Protocol) : Gère les informations relatives aux erreurs de transmission. ICMP ne les corrige pas, il signale uniquement que le message contient des erreurs.

II.9 Comparaison entre les modèles OSI et TCP/IP :

Ressemblances :

Les deux modèles OSI et TCP/IP comportent :

• Des couches.

• Une couche application, bien que chacune fournisse des services différents.

• Des couches réseau et transport comparables.

• Tous deux s'appuient sur un réseau à commutation de paquets, et non sur un réseau à commutation de circuits.

Différences :

• Le modèle TCP/IP est relativement flou vis avis la couche application en effet c'est aux constructeurs de spécifier cette couche.

• Le modèle OSI est trop détaillé.

• La dernière grande différence est liée au mode de connexion. Certes, les modes orienté connexion et sans connexion sont disponibles dans les deux modèles mais pas à la même couche : pour le modèle OSI, ils ne sont disponibles qu'au niveau de la couche réseau (au niveau de la couche transport, seul le mode orienté connexion n'est disponible), alors qu'ils ne sont disponibles qu'au niveau de la couche transport pour le modèle TCP/IP (la couche internet n'offre que le mode sans connexion). Le modèle TCP/IP a donc cet avantage par rapport au modèle OSI : les applications (qui utilisent directement la couche transport) ont véritablement le choix entre les deux modes de connexion.

III. Le web

III.1. Définition du web :

Le web est un système de documents hypermédia distribué. Il a été développé au CERN (Centre Européen de recherche en physique Nucléaire) par Tim Bernes-Lee en 1989.Il est constitué de documents multimédias (page de texte enrichi de sons, graphiques, images fixes et animées, vidéos, etc.)

Qui sont reliés entre eux par des liens hypertextes, et qui permet de consulter ces informations disposé en pages web à l'aide d'un navigateur.

Chaque document (ou page) est identifié par une adresse précise de la forme

http://www.site.dom/répertoire/nompage.html

Exemple : http://www.wikipédia.org/wiki/web.html

III.2. Les principaux termes du web :

- **A.Document Hypertexte** : c'est un fichier contenant les liens soit vers l'autres parties du document lui même, soit vers d'autres document. Ces liens permettant de relier plusieurs documents web entre eux et de les rendre interactifs.
- **b. Hypermédia** : Ce terme désigne un hypertexte dont on insiste sur la nature multimédia de son contenu informationnel.
- **c. HTML (Hypertexte Mark up Langage)** : est un langage de programmation utilisé pour créer des pages web. Ce langage permet entre autres de créer des liens hypertextes.

- **d. Page web** : est une ressource du web conçue pour être consultée par des visiteurs à l'aide d'un navigateur web. Elle a une adresse web et est constituée d'un document écrits en Hypertexte Mark up Langage (HTML) ou (XHTML), ainsi elle peut contenir des images, des son ou de la vidéo.

Une page web est dite statique quand son contenu est fixé une fois pour toute au moment de sa création

Une page web est dite dynamique quand elle est produite au moment où le navigateur demande de la visualiser.

- **e. Site web** : est un ensemble de pages web hyper liées entre elles et mises en ligne à une adresse web. On dit aussi site internet par abus de langage.

- **f. HTTP (Hypertexte Transfer Protocol)** : est le Protocol de communication utilisée pour la transmission des pages web depuis un serveur vers un navigateur client.
- **g. Navigateur web** : est un logiciel conçu pour consulter le web techniquement, c'est au minimum un client http par exemple : Internet Explorer (Microsoft), FireFox (Mozilla), Chrome (Google).
- **h. Serveur web** : est un programme spécifique qui répond à des demandes de consultation des documents faites par des clients (navigateurs) par exemple : Apache, IIS (Internet Information Serveur).
- **i.Page d'accueil** : c'est la page d'introduction d'un site web, qu'on peut aussi appeler "page de couverture".elle est très importante car elle détermine la première impression du visiteur du site web (internaute).
- **j. URL (Uniform Ressource Locator)** : littéralement «localisateur uniforme de ressource » est une adresse d'accueil qui contient à la fois le nom d'une machine et aussi le nom du service demandé. L'URL permet de réparer des documents, des fichiers, des sources, et des données fournies par différents serveurs.

III.3. Type des sites web :

III.3.1. Site web statique :

Un site web statique conçu en langage HTML. Chaque page web est fixe, elle est écrite pour durer un certain temps.

Ce format convient au site vitrines qui n'ont pas besoin d'évoluer. En cas de mises à jour, celles-ci sont confiées à l'administrateur qui envoie des modifications via un serveur FTP.

Figure I .5 : Schéma illustratif d'un site web statique.

III.3.2 Site web dynamique :

Est appelé aussi application web, constitué essentiellement de pages dynamiques, l'administrateur du site et le visiteur utilise lé même outil : le navigateur web.

Le contenu des pages fournies est issu d'une base de données ou de ressources distantes, les informations visibles sur les pages ne sont pas contenues dans les pages mais dans une base de données par exemple

La particularité des sites web dynamiques est que les pages lues par les visiteurs n'existent pas physiquement, elles sont fabriquées en temps réel par un ensemble de logiciels scripts résidents dans le serveur sur lequel il est hébergé.

Figure I .6 : Schéma illustratif d'un site web dynamique.

III.4. Les langages de développement d'un site web dynamique :

III.4.1. Les langages exécutés côté serveur :

◦ **PHP (Personnel Home Page)** : c'est un langage de script fonctionnant essentiellement sur les serveurs, c'est le plus populaire (plus de 70% de serveurs l'utilisent).

Ce langage est incorporé au document HTML, mais exécuté par le serveur web et non par le client. Conçu pour réaliser des pages dynamiques, le résultat de script est un document HTML standard.

◦ **JSP (Java Server Page)** : il est le plus récent parmi toutes les technologies utilisées, et permet d'utiliser toute la puissance de java pour créer des pages web dynamiques.

◦**ASP (Active Server Page)** : permet de développer des applications web interactifs.

C'est-à-dire dont le contenu dynamique.

◦ **Servlets** : permet d'effectuer les traitements selon la requête de l'utilisateur (accès a des bases de données, communication avec d'autres servlets, interrogation d'annuaire).

◦ **CGI (Common Gateway Interface)** : permet d'envoyer au navigateur de l'internet un code HTML créé automatiquement par le serveur, l'objectif de cette interface était la publication d'une manière dynamique par la création des documents HTML à la demande de l'utilisateur (au lieu d'obtenir une page statique on obtient une page dynamique).

III.4.2. Les langage exécuté côté client :

◦ **HTML (Hypertexte Mark up Langage)** :c'est un langage de description des documents, il des marqueurs explicites (balises) qui précisent la structure et la mise en forme du contenu du document, ces marqueurs seront reconnus par les navigateurs. Ce langage permet de définir la présentation de document ainsi que les liens Hypertextes vers d'autres documents à l'aide des balises de formatage.

◦**XML (eXtensible Markup Langage)** :C'est en quelque sorte un langage html amélioré qui permet de structurer, poser le vocabulaire, la syntaxe des données d'un document et de séparer le contenu de la présentation.

En réalité, les balises XML décrivent le contenu plutôt que la représentation (contrairement à HTML).

◦**JavaScript** :c'est un langage de programmation qui permet d'apporter des améliorations au langage HTML en permettant d'exécuter des commandes de côté client (au niveau de navigateur).En principe, JavaScript permet d'accroitre l'interactivité des pages web et offre la possibilité de créer une interface utilisateur active.

◦**Applets** : ce sont des petits programmes exécutables sur le navigateur du client, intègres aux documents HTML, même a l'intérieur d'une page web permettant de rendre la page beaucoup plus interactive (mais aussi beaucoup plus lente).

III.5. Les différentes étapes de création d'un site web :

Pour disposer d'un site internet, nous devons passer par plusieurs composants indispensables :

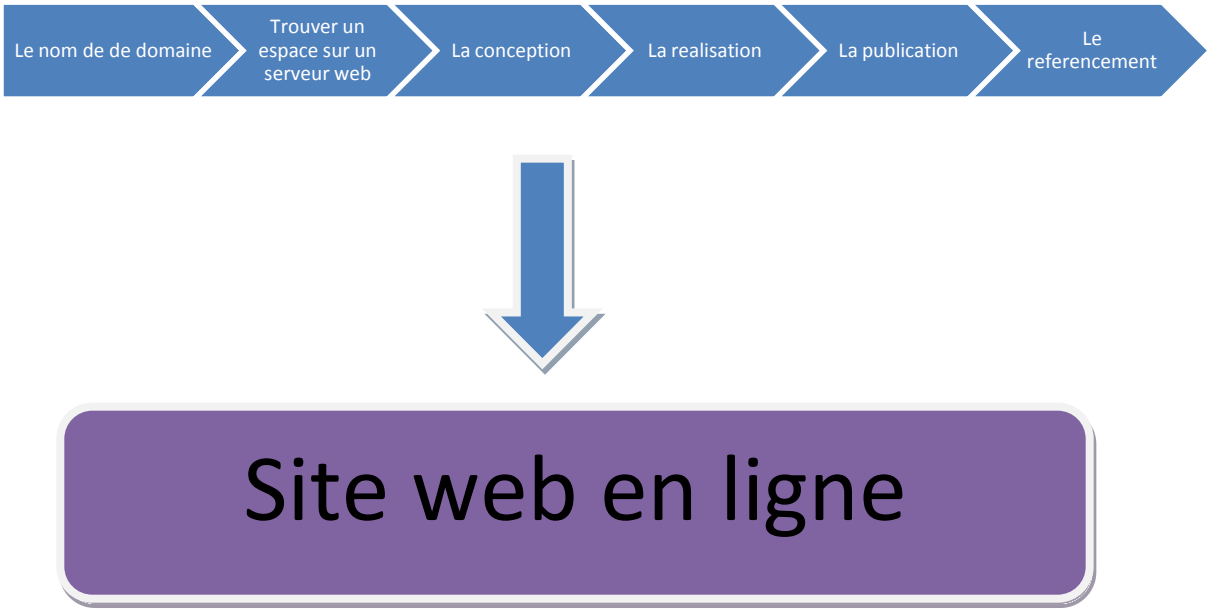


Figure I .7 : Les différentes étapes de création d'un site web.

- **Obtenir un nom de domaine** : chaque page web est associée a une adresse unique : son URL, or cette URL est elle-même dépendante d'un nom de domaine.

Un nom de domaine est un nom que vous achetez et qui vous permet ensuite de rendre votre page web accessible par des adresses (URL) liée à ce nom.

Un nom de domaine est constitué par la chaine de caractère que vous saisissez dans la barre d'adresse du navigateur pour accéder a un site web. Il est constitué d'un nom et d'un suffixe (extension), ces deux éléments étant séparer par un point et d'un chemin d'accès.

- **Trouver un espace sur un serveur web** : pour que votre site soit visible par le monde entier 7 jour sur 7 et 24 heures sur 24, les fichiers ou pages web doivent être stockées sur sites web et de les rendre accessibles : c'est un serveur web.son rôle est d'héberger des sites web et de les rendre accessibles sur internet. On dispose de deux solutions pour héberger un site web : en interne ou en externe.

Dans le cas ou vous choisissez d'externaliser la prestation d'hébergement, vous avez encore le choix entre divers types d'hébergement qui sont soit des hébergements gratuits ou bien des hébergements payants.

Définition d'un hébergeur :

Un hébergeur web est une entreprise de services informatiques hébergeant (mettant en ligne) sur ses serveurs web les ressources constituant les web de ses clients.

.les hébergeurs gratuits : les fournisseurs d'accès à l'internet proposent généralement dans leur offre une formule d'hébergement de site web avec un espace de stockage assez limité, en effet le serveur d'hébergement est souvent rendu en contrepartie de publicité sur le site.

. **Les hébergeurs professionnels :** Ce type d'hébergement garanti un service de qualité et de sécurité.

- **Conception :** c'est l'étape qui conditionne la réussite de notre projet, dans cette phase de conception, nous réfléchissons aux éléments suivants :
  - **L'objectif de notre site web** : celui doit répondre au besoin des internautes.
  - **La cible de notre site web** : c'est à-dire à qui souhaiter vous adresser ?

    - ✓ **Les concurrents de notre site web** sot les visiteurs qui sont en concurrence adoptant un œil critique.
    - ✓ **L'arborescence de notre site** web : représente la hiérarchie des rubriques et des sous rebiques avec, le sommet, la page d'accueil de notre site web.
    - ✓ **La charte graphique de notre site web**: l'écran est un support très visuel, la première impression est esthétique.
- **La réalisation** : dans cette étape de réalisation, on utilisera les outils logiciels spécialisés dans la conception des pages web.

Le principal outil logiciel qu'on utilisera pour la construction de nos pages web est un éditeur HTML par exemple Dreamwever de macromedia

- **La Publication** : Une fois le site réalisé, pour qu'il soit visible et consultable par le monde entier, il nous faut le publier c'est-à-dire transférer nos pages vers le serveur que nous avons réservé chez l'hébergeur. Ce serveur constitue pour notre site la porte ouverte sur l'internet.

Pour effectuer le transfère de nos fichiers, nous utiliserons un logiciel spécifique appelé client FTP.

- **Référencement** : consiste a inscrire le site web sur les outils de recherche.la démarche de référencement comporte plusieurs étapes :
  - Insertion des mots clés dans les pages.
  - Demande d'indexation sur les outils de recherche.

Les mots clés sont les mots ou expressions que les internautes saisissent lorsqu'ils cherchent un site web en utilisant les outils de recherche.

IV. Services web :

IV.1 Définition des services web

La technologie des services Web est un moyen rapide de distribution de l'information entre clients, fournisseurs, partenaires commerciaux et leurs différentes plates-formes. Les services Web sont basés sur le modèle SOA

D'autres technologies telles que RMI, DCOM et CORBA ont précédemment adopté ce style architectural mais ont généralement échoué en raison de la diversité des plates-formes utilisées dans les organisations et aussi parce que leur usage n'était pas adapté à Internet (problème de passage à travers des FireWalls, etc.) d'où la lenteur, voire l'absence de réponses sur ce réseau. Les applications réparties fondées sur ces technologies offrent des solutions caractérisées par un couplage fort entre les objets. Les solutions proposées par les services Web, permettent néanmoins un couplage moins fort. De plus, l'utilisation des technologies standards du Web telles HTTP et XML par les services Web facilite le développement d'applications réparties sur Internet, et permet d'avoir des applications très faiblement couplées. L'intégration est sans doute le facteur essentiel qui favorise l'utilisation des services Web.

IV.2 Les principaux avantages d'un service Web :

- Une interface décrite d'une manière interprétable par les machines, qui permet aux applications clientes d'accéder aux services de manière automatique ;
- L'utilisation des langages et des protocoles indépendants des plates-formes d'implantation, qui renforcent l'interopérabilité entre services ;
- L' utilisation des normes actuelles du Web, qui permettent la réalisation des interactions faiblement couplées et favorisent aussi l'interopérabilité.

IV.3 L'intérêt d'un Service Web

Les services Web fournissent un lien entre applications. Ainsi, des applications utilisant des technologies différentes peuvent envoyer et recevoir des données au travers de protocoles compréhensibles par tout le monde.

Les services Web sont normalisés car ils utilisent les standards XML et HTTP pour transférer des données et ils sont compatibles avec de nombreux autres environnements de développement. Ils sont donc indépendants des plates-formes. C'est dans ce contexte qu'un

intérêt très particulier a été attribué à la conception des services Web puisqu'ils permettent aux entreprises d'offrir des applications accessibles à distance par d'autres entreprises. Cela s'explique par le fait que les services Web n'imposent pas de modèles de programmation spécifiques. En d'autres termes, les services Web ne sont pas concernés par la façon dont les messages sont produits ou consommés par des programmes. Cela permet aux vendeurs d'outils de développement d'offrir différentes méthodes et interfaces de programmation au-dessus de n'importe quel langage de programmation, sans être contraints par des standards comme c'est le cas de la plate-forme *CORBA* qui définit des ponts spécifiques entre le langage de définition IDl et différents langages de programmation. Ainsi, les fournisseurs d'outils de développement peuvent facilement différencier leurs produits avec ceux de leurs concurrents en offrant différents niveaux de sophistication.
Les services Web représentent donc la façon la plus efficace de partager des méthodes et des fonctionnalités. De plus, ils réduisent le temps de réalisation en permettant de tirer directement parti de services existants.

IV.4 Les caractéristiques d'un service Web

La technologie des services Web repose essentiellement sur une représentation standard des données (interfaces, messageries) au moyen du langage XML. Cette technologie est devenue la base de l'informatique distribuée sur Internet et offre beaucoup d'opportunités au développeur Web.

Un service Web possède les caractéristiques suivantes :

- il est accessible via le réseau.
- il dispose d'une interface publique (ensemble d'opérations) décrite en XML.
- ses descriptions (fonctionnalités, comment l'invoquer et où le trouver ?) sont stockées dans un annuaire.
- il communique en utilisant des messages XML, ces messages sont transportés par des protocoles Internet (généralement HTTP, mais rien n'empêche d'utiliser d'autres protocoles de transfert tels : SMTP, FTP, BEEP...).
- l'intégration d'application en implémentant des services Web produit des systèmes faiblement couplés, le demandeur du service ne connaît pas forcément le fournisseur.
  Ce dernier peut disparaître sans perturber l'application cliente qui trouvera un autre fournisseur en cherchant dans l'annuaire.

IV.5 Architecture d'un service Web

Les services Web reprennent la plupart des idées et des principes du Web (HTTP, XML), et les appliquent à des interactions entre machines. Comme pour le **World Wide Web**, les services Web communiquent via un ensemble de technologies fondamentales qui partagent une architecture commune. Ils ont été conçus pour être réalisés sur de nombreux systèmes développés et déployés de façon indépendante. Les technologies utilisées par les services Web sont HTTP, WSDL, REST, XML-RPC, SOAP et UDDI.

REST :

REST (*Representational State Transfer*) est une architecture de services Web. Élaborée en l'an 2000 par *Roy Fiedling*, l'un des créateurs du protocole HTTP, du serveur Apache HTTPd et d'autres travaux fondamentaux, REST est une manière de construire une application pour les systèmes distribués comme le World Wide Web.

XML-RPC :

XML-RPC est un protocole simple utilisant XML pour effectuer des messages RPC. Les requêtes sont écrites en XML et envoyées via HTTP POST. Les requêtes sont intégrées dans le corps de la réponse HTTP. XML-RPC est indépendant de la plate-forme, ce qui lui permet de communiquer avec diverses applications. Par exemple, un client Java peut parler de XML-RPC à un PerlServer !

SOAP :

SOAP (*Simple object Access Protocol*) est un protocole standard de communication. C'est l'épine dorsale du système d'interopérabilité. SOAP est un protocole décrit en XML et standardisé par le W3C. Il se présente comme une enveloppe pouvant être signée et pouvant contenir des données ou des pièces jointes.
Il circule sur le protocole HTTP et permet d'effectuer des appels de méthodes à distance.

WSDL :

WSDL (*Web Services Description Language*) est un langage de description standard. C'est l'interface présentée aux utilisateurs. Il indique comment utiliser le service Web et comment interagir avec lui. WSDL est basé sur XML et permet de décrire de façon précise les détails concernant le service Web tels que les protocoles, les ports utilisés, les opérations pouvant être effectuées, les formats des messages d'entrée et de sortie et les exceptions pouvant être envoyées.

UDDI :

UDDI (*Universal Description, Discovery and Integration*) est un annuaire de services. Il fournit l'infrastructure de base pour la publication et la découverte des services Web. UDDI permet aux fournisseurs de présenter leurs services Web aux clients.

Les informations qu'il contient peuvent être séparées en trois types :

- les pages blanches qui incluent l'adresse, le contact et les identifiants relatifs au service Web ;
- les pages jaunes qui identifient les secteurs d'affaires relatifs au service Web ;
- les pages vertes qui donnent les informations techniques.

IV.6 Fonctionnement des services Web

Le fonctionnement des services Web s'articule autour de trois acteurs principaux illustrés par le schéma suivant :

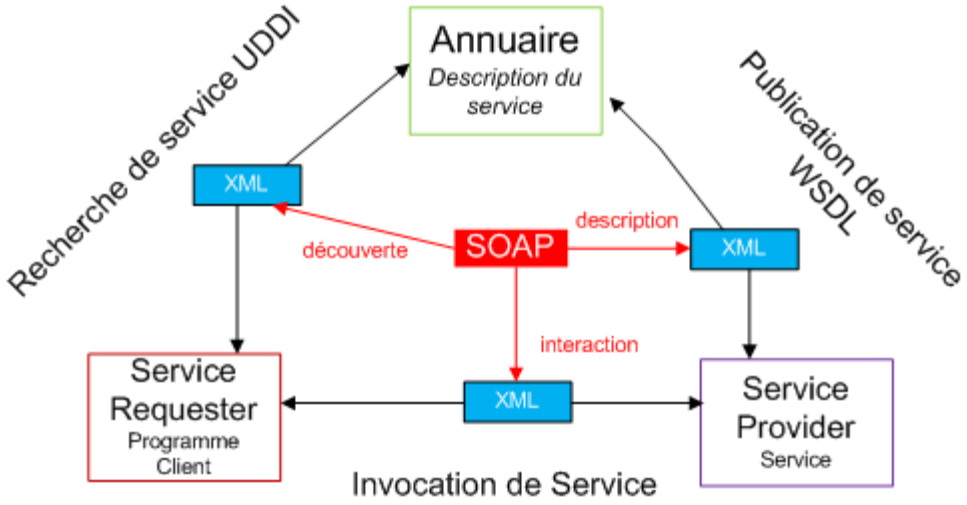


Figure I .8 : schéma illustrant les trois principaux acteurs des services Web.

Service provider service

Le fournisseur de service met en application le service Web et le rend disponible sur Internet.

Service requester programme client

C'est n'importe quel consommateur du service Web. Le demandeur utilise un service Web existant en ouvrant une connexion réseau et en envoyant une demande en XML (REST, XML-RPC, SOAP).

Annuaire service registry

Le registre de service est un annuaire de services. Le registre fournit un endroit central où les programmeurs peuvent publier de nouveaux services ou en trouver. Les interactions entre ces trois acteurs suivent plusieurs étapes :

- **La publication du service** : le fournisseur diffuse les descriptions de ses services Web dans l'annuaire.
- **La recherche du service** : le client cherche un service particulier, il s'adresse à un annuaire qui va lui fournir les descriptions et les URL des services demandés afin de lui permettre de les invoquer.
- **L'invocation du service** : une fois que le client récupère l'URL et la description du service, il les utilise pour l'invoquer auprès du fournisseur de services.

Description en couche des services Web

Les services Web emploient un ensemble de technologies qui ont été conçues afin de respecter une structure en couches sans être dépendante de façon excessive de la pile des protocoles. Cette structure est formée de quatre couches majeures :

Découverte de services	UDDI
Description de services	WSDL
Communication	SOAP
Transport	HTTP

Figure I .9 Couches technologiques des services Web.

- Le transport de messages XML-RPC ou SOAP est assuré par le standard HTTP.
- SOAP ou XML-RPC prévoit la couche de communication basée sur XML pour accéder à des services Web.
- La description d'un service Web se fait en utilisant le langage WSDL. WSDL expose l'interface du service.
- La publication et la découverte des services Web sont assurées par le biais du référentiel UDDI. Un référentiel UDDI est un catalogue de services Web.

Couche transport :

Cette couche est responsable du transport des messages XML échangés entre les applications. Actuellement, cette couche inclut HTTP, SMTP, FTP, et de nouveaux protocoles tels que BEEP.

Couche communication :

Cette couche est responsable du formatage des données échangées de sorte que les messages peuvent être compris à chaque extrémité. Actuellement, deux styles architecturaux totalement différents sont utilisés pour ces échanges de données. Nous avons d'un côté l'architecture orientée opérations distribuées (protocoles RPC) basée sur XML et qui comprend XML-RPC et SOAP et de l'autre côté une architecture orientée ressources Web, REST (Representational State Transfer) qui se base uniquement sur le bon usage des principes du Web (en particulier, le protocole HTTP).

Conclusion :

Au cours de ce chapitre nous avons abordé certains concepts des nouvelles technologies de l'information et de la communication à savoir : l'internet et le web.

Ces différents concepts traités nous aiderons a mieux comprendre le mode d'opération et le fonctionnement dans notre application.

Chapitre 2

ANDROID

Introduction :

Dans ce chapitre nous allons présenter la technologie Android, son historique, son architecture, et les outils nécessaires au développement Android.

I. Présentation générale :

Android est un système d'exploitation édité par Google pour des appareils embarqués et/ou mobiles, comme les Smartphones ou les tablettes. On le retrouve aussi dans certains GPS, ordinateurs de bord de voitures, dans des télévisions, autoradios, et même des montres. De plus, de nombreux prototypes d'appareils électroménagers, comme des réfrigérateurs ou des machines à laver, fonctionnant sous Android ont été présentés ces derniers temps, permettant ainsi de lancer une machine à l'aide de son téléphone, ou encore d'être prévenu par son frigo lorsqu'il manque certaines choses.

Le système Android est basé sur un noyau de Linux. Ce dernier a été modifié pour être plus adapté aux terminaux mobiles ayant peu de puissance de calcul, de mémoire et de batterie. En effet, certaines librairies standards ne sont pas supportées par le système, et des améliorations ont été apportées sur la gestion de l'énergie.

Les applications sont écrites en Java, et fonctionnent au sein d'une machine virtuelle Dalvik. Cette machine virtuelle a été aussi modifiée pour être le plus adapté possible aux appareils de faible puissance. Ainsi, beaucoup d'efforts ont été fournis sur la consommation de mémoire, qui a été largement diminuée par rapport à la machine virtuelle java classique.

D'autre part, ce système est open source, ce qui permet à toute personne de lancer sa propre version d'Android. De nombreuses personnes utilisent ainsi des "ROM custom", c'est à dire des versions modifiées par rapport au code de base fourni par l'éditeur.

II.1 Historique :

Android commence en octobre 2003, où la société Android Inc. est créée. Officiellement, elle développe des logiciels pour mobiles. Mais en réalité, elle se préparait à sortir un tout nouveau système d'exploitation pour Smartphones. En 2005, Google rachète cette entreprise, et sort une première béta en novembre 2007, avant de lancer la version 1.0

en septembre 2008 avec le HTC Dream. À partir de ce moment-là, le rythme des nouvelles sorties est très élevé : pas moins de 32 versions différentes sont apparues à notre jour.

La version actuelle est la 4.4, nommée " Kit Kat", et est disponible depuis 31 octobre 2013. Toutefois, peu de Smartphones ont été mis à jour, cette tâche étant déléguée aux constructeurs.

Plus de 1 300 000 applications sont disponibles sur le Market en juillet 2014 et huit téléphones sur dix acheté au niveau mondial fonctionnent sous Android. Ce système d'exploitation est, en effet, gratuit pour les constructeurs, ce qui les incite à sortir de nouveaux Smartphones : de fait, plus de 30 marques proposent des Smartphones ou des tablettes : HTC, Samsung, Sony Ericsson, etc.

Android, même s'il est majoritaire, n'est pas le seul sur le marché des Smartphones. Il doit, en effet, faire face à la concurrence. De plus, Google doit aussi compter sur la concurrence de Microsoft avec Windows Mobile 8, et RIM (Blackberry). Quant à Nokia et son système Symbian.

II.2 Bugdroid :

Figure II.1 : Bugdroid

Le personnage nommé Bugdroid est la mascotte verte utilisée par Google et toutes les communautés Android pour représenter le système d'exploitation (OS).

Ce personnage est sous licence « creative commons by (3.0)» et peut donc être utilisé librement. C'est ainsi que l'on peut voir une multitude de petits robots verts. Certains sont juste Fun, d'autres sont le symbole d'une communauté d'internautes passionnés.

Selon certains sites comme Engadget, Bugdroid serait en fait un personnage d'un jeu des années 1990 sur Atari : Gauntlet: The ThirdEncounter.

III - Caractéristiques d'Android :

Le système d'exploitation Android est développé sur un noyau Linux ce qui fait de ce système un open-source. Son interface est développée en java ce qui lui procure une grande souplesse d'interactivité. C'est un interpréteur de type JIT (Just-In-Time) qui exécute les programmes en faisant une compilation à la volée ce qui procure un gain de place énorme pour le système, toutefois il est possible de passer outre cette interface.

Certaines applications ont été développées directement en C ce qui donne beaucoup de souffle au système, libère de la mémoire et accélère les programmes, de plus le développement en C est beaucoup plus efficace et beaucoup plus rapide.

La grande hétérogénéité des coprocesseurs procurera certaines différences d'un appareil à l'autre malgré une architecture ARM (Advanced Risc Machine). Seul le système HAL (Hardware Abstraction Layer), couche d'abstraction matérielle, peut laisser une impression de léger retard pour le système sur ses concurrents car il est déjà considéré comme obsolète !
Par contre, on trouvera dans le système Android tous les utilitaires indispensables à tout téléphone portable.

Android a été conçu pour intégrer au mieux des applications existantes de Google comme le service de courrier Gmail, celui de cartographie, Google Maps, ou encore Google Agenda, Google Talk, YouTube…etc.

III.1 Architecture de la plateforme Android :

L'architecture de la plateforme Android se décline, selon une démarche bottom up, en quatre principaux niveaux que sont :

- Le noyau linux ;
- Les librairies et l'environnement d'exécution ;
- Le module de développement d'application ;
- Les différentes applications

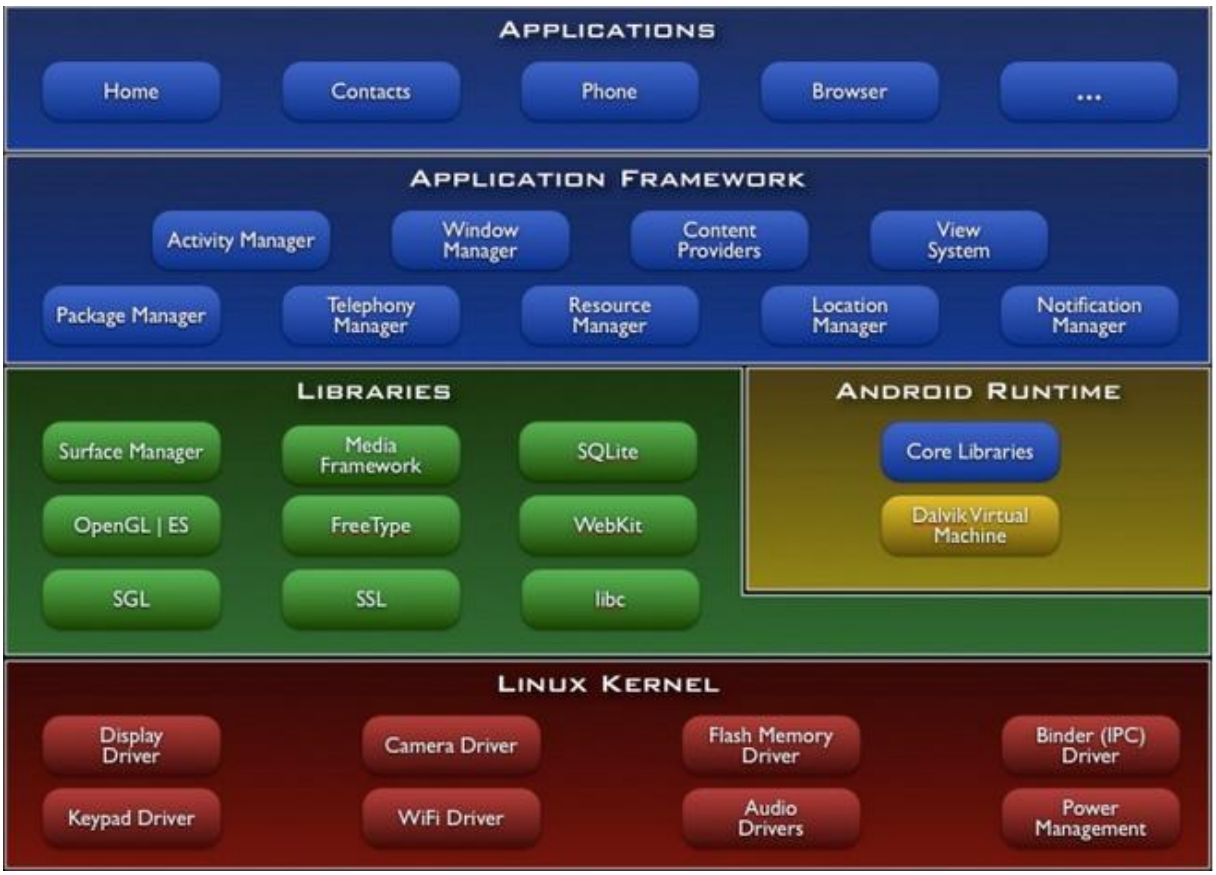


Figure II.2 : Architecture de la plateforme Android.

III.1.1 Premier niveau : noyau linux :

Android s'appuie sur le noyau Linux 2.6. Le noyau est l'élément du système d'exploitation qui permet de faire le pont entre la partie matérielle et la partie logicielle. La version du noyau utilisée avec Android est une version conçue spécialement pour l'environnement mobile, avec une gestion avancée de la batterie et une gestion particulière de la mémoire.

III.1.2 Deuxième niveau : librairies et environnement d'exécution :

a) Les librairies :

Ces bibliothèques proviennent de beaucoup de projets open-sources, écrits en c/c++ pour la plupart, comme SQLite pour les bases de données, WebKit pour la navigation web ou encore OpenGL, afin de produire des graphismes en 2D ou 3D.

Malgré que le développement d'applications se fasse en langage JAVA, Android comprend très bien le C et le C++.

b) L'environnement d'exécution :

Le runtime Android est basé sur le concept de machine virtuelle utilisée en java. Etant donné les limitations des dispositifs (peu de mémoire et vitesse du processeur), il n'a pas été possible d'utiliser une machine virtuelle java standard. Google a pris la décision de créer une nouvelle machine virtuelle Dalvik, afin de mieux répondre à ces limitations. Dans cette partie qui est l'environnement d'exécution, on retrouve essentiellement :

b.1) Core Libraries :

Les Corelibraries fournissent le langage java, disponible pour les applications. Le langage java fournit avec Android reprend en grande partie l'API (Application programming Interface) JSE 1.5. Il y a des choses qui ont été mises de côté, car cela n'avait pas de sens pour Android (comme les imprimantes, swing, etc…) et d'autres APIs spécifiques requises pour Android ont été rajoutées.

b.2) Dalvik :

Les applications java, développées pour Android, doivent être compilées au format Dalvik exécutable (.dex) avec l'outil (dx). Cet outil compile les (.java) en (.class) et ensuite il convertit ces (.class) en (.dex).

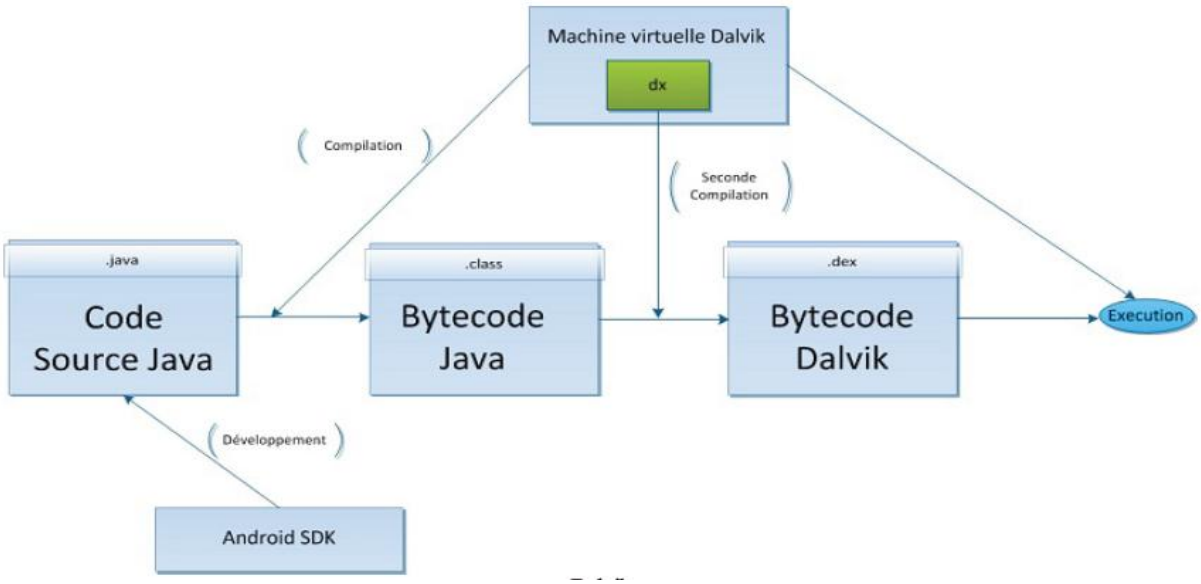


Figure II.3 : la machine virtuelle Dalvik.

III.1.3 Troisième niveau : module de développement d'applications :

Le Framework est situé au-dessus de l'Android Runtime et des librairies. Il fournit des API utilisées par les applications de base permettant aux développeurs de créer des applications extrêmement riches et innovantes. L'architecture d'applications est conçue pour simplifier la réutilisation des composants ; n'importe quelle application peut publier ses capacités et n'importe quelle autre application peut alors faire usage de ces capacités. Ce même mécanisme permet aux composants de base d'être remplacés par l'utilisateur.

Toutes les applications sous-jacentes forment un ensemble de services (un service est une application, qui n'a aucune interaction avec l'utilisateur et qui tourne en arrière-plan) et de systèmes, y compris :

- Activity Manager : gère le cycle de vie des applications et maintient une pile de navigation permettant d'aller d'une application à une autre ;
- Content Provider : gère le partage de données entre applications ;
- View System : fournit tous les composants graphiques tels que : listes, grilles, boutons, etc ;
- Wifi Service : permet d'accéder à l'interface Wifi

- Telephony Service : permet d'accéder aux interfaces téléphoniques (GSM « Global Système for Mobile Communications », 3G, etc.).

III.1.4 Quatrième niveau : applications :

Il s'agit tout simplement d'un ensemble d'applications que l'on peut trouver sur Android, par exemple, les fonctionnalités de base incluent un client, pour recevoir/envoyer des emails, une application pour envoyer/recevoir des SMS, un calendrier, un répertoire, etc.

III.2 SDK Android :

Approchant le fonctionnement de Flex (ActionScript/MXML) ou de .NET, Android fournit un kit de développement SDK (Software Development Kit) qui permet de développer des applications spécifiques de la téléphonie mobile à mettre en œuvre sur la plate-forme.

III.2.1 Développement :

III.2.1.1 Version :

Les différentes versions d'Android ont toutes des noms de desserts « en anglais » depuis la sortie de la version 1.5 et suivent une logique alphabétique (de A vers Z) : Apple Pie (Tarte aux pommes), version connue uniquement ou presque des développeurs car c'est la Sdk distribuée avant la sortie du premier téléphone Android.

- **1.0** : Apple Pie (Tarte aux pommes), sortie en 23 septembre 2008 ;
- **1.1 :** Banana bread (Pain de la banane), sortie en 9 février 2009 ;
- **1.5** : Cupcake (Petit Gâteau), sortie en avril 2009 ;
- **1.6 :** Donut (Beignet), sortie en 15 septembre 2009 ;
- **2.0 (2.0.1) :** sortie le 26 octobre 2009, version appelée Éclair au départ mais, à cause de nombreux bugs, vite remplacée par la 2.0.1 puis par la 2.1. Cette version 2.0 est très peu connue ;
- **2.1 :** Éclair, sortie en janvier 2010 ;
- **2.3 (2.3.7) :** Gingerbread (Pain d'épice), sortie le 6 décembre 2010, support la VOIP et SIP ;

- **3.0 (3.2.6) :** Honeycomb (Rayon de miel), sortie le 2 février 2011, version pour grandes tablettes et télévisions connectées ;
- **4.0.1 (4.0.4) :** IceCream Sandwich (Sandwich à la crème glacée), version unifiée pour Smartphone, tablette et Google TV, fortement inspirée d'Honeycomb, sortie le 19 octobre 2011 ;
- **4.1 (4.3) :** Jelly Bean (dragée), sortie le 27 juin 2012, il ajoute un système de notification améliorée, la reconnaissance vocale sans connexion internet, et le « Project Butter » qui augmente la fluidité d'Android ;
- **4.4 :** Kitkat, sortie le 31 octobre 2013. Consommation en ressource moins élevée nécessitant moins de RAM, nouvelles icones plus soignées, la barre de bas et celle de statut deviennent transparentes sur certains menus et changent de couleur en fonction du contenu affiché.

Selon plusieurs documents officiels, les fichiers SDK de la version developé pre view de **Android L** et le code source Android (Android Open Source Project), la prochaine version majeure de l'OS (Opérating system) de Google pour Smartphones et tablettes connue jusqu'ici sous le nom ou le diminutif Android L, pourrait bien s'appeler **Lemon Meringue Pie**« *LMP* » (Tarte au citron meringuée). La référence **LMP** aurait en effet été découverte dans de nombreux fichiers du **SDK d'Android L** et également dans un document de la WiFi Alliance au sujet de la certification WiFi d'une certaine HTC Volantis (la Nexus 8/9). Pour rappel, la version finale d'Android L est prévue pour la fin d'année 2014.

III.2.1.2 Répartition actuelle des versions d'Android :

Répartition des différentes versions, au 12 août 2014

Version	Nom de version	API	Distribution
2.2	Froyo	8	0,7%
2.3.3-2.3.7	Gingerbread	10	13,6%
4.0.3-4.0.4	Icecream sandwich	15	1O,6%
4.1. x	Jelly Bean	16	26,5%

4.2. x		17	19,8%
4.3		18	7 ,9%
4.4	Kitkat	19	20,9

Figure : II.4 : Table illustrant la part de chaque version d'Android au 12 août 2014.

III.2.2 Quelques outils que fournit SDK pour les développeurs :

Le SDK Android compte plusieurs outils et utilitaires pour nous aider à créer, tester et déboguer nos projets. On présentera brièvement quelques un de ces outils

❖ III.2.2.1 Android Virtual Device Manager :

Cet outil est utilisé pour créer et gérer les terminaux virtuels (AVD) qui hébergeront les instances de l'émulateur. Ces AVD servent à simuler les versions logicielles et les configurations matérielles disponibles sur différents appareils physiques. Les applications peuvent être ainsi testées sur plusieurs plateformes matérielles sans avoir besoin d'acheter les téléphones correspondants.

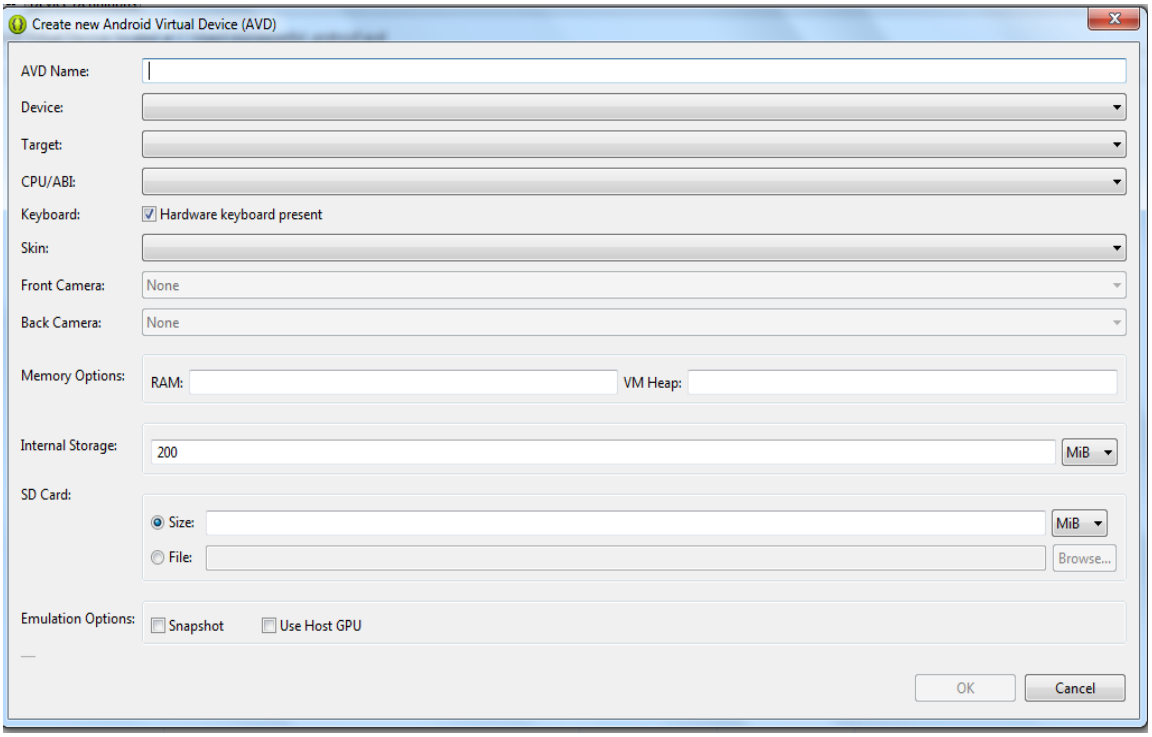


Figure II.5 : Android Virtual Device Manager.

❖ III.2.2.2 SDK Manager :

Le SDK Manager permet de connaitre la version du SDK installée et d'en installer de nouvelles lorsqu'elles sont disponibles.

Outre la version de la plateforme, il affiche les outils et un certain nombre de packages supplémentaires. Chaque version de plateforme comprend le SDK, la documentation, des outils et des exemples correspondant à la version concernée.

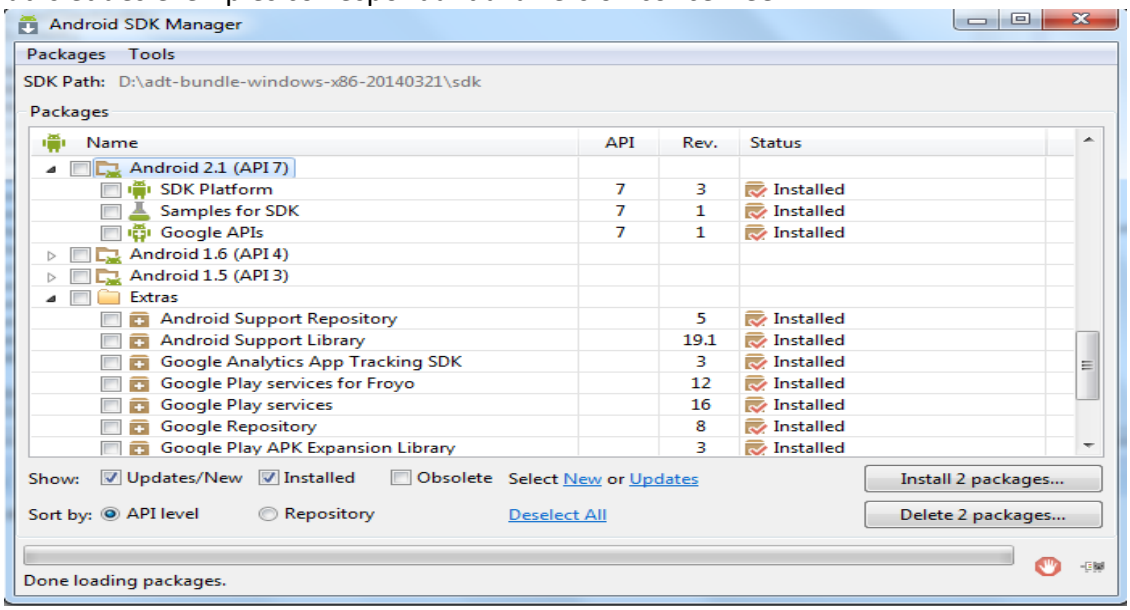


Figure II.6 : SDK Manager.

❖ III.2.2.3 L'émulateur Android :

L'émulateur est le parfait outil de test et de débogage de nos applications. C'est une implémentation de la machine virtuelle Dalvik, faisant de celle-ci une plateforme exécutant des applications Android comme le ferait n'importe quel téléphone Android. Etant découplé de tout matériel, c'est une excellente base de test de nos applications.

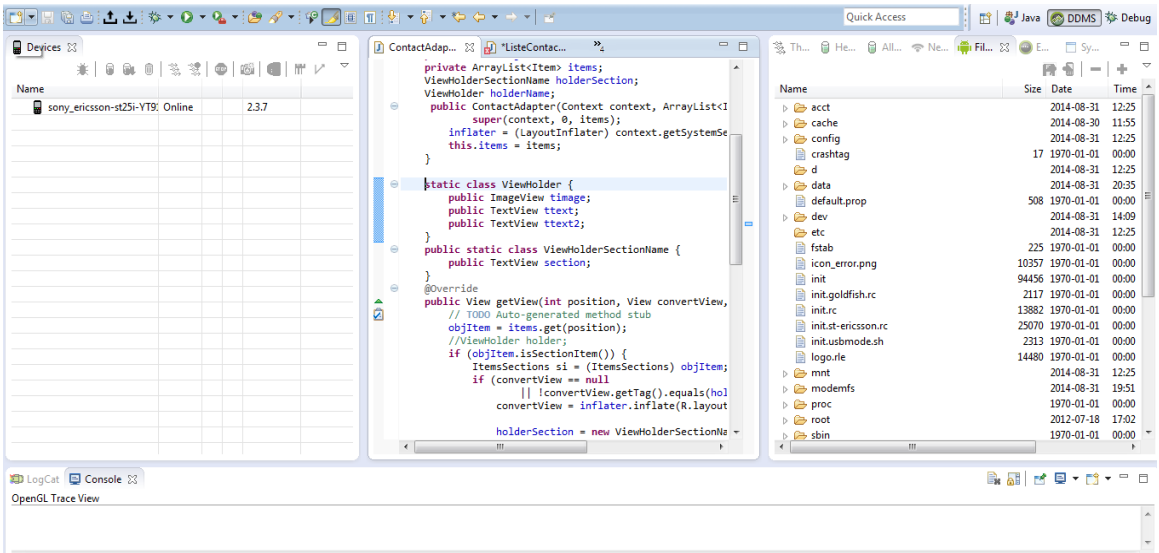
Il fournit une connectivité réseau complète ainsi que la possibilité d'ajouter finement vitesse de connexion et latence au cours de débogage des applications. Nous pouvons également simuler l'envoi et la réception d'appels et de SMS.

Figure II.7 : L'émulateur Android.

❖ **III.2.2.4 DDMS (Dalvik Debug Monitor Service) :**

L'émulateur permet de voir à quoi ressemblera notre application et la façon dont elle se comportera et interagira, mais c'est le DDMS qui nous permettra de voir ce qui se passe en profondeur. C'est un puissant outil de débogage avec lequel nous pourrons interroger les processus actifs, examinons la pile et le tas, surveillons et mettrons en pause les threads actifs et explorer le système de fichiers de n'importe quel matériel Android connecté.

La perspective DDMS sous Eclipse fournit également un accès simplifié aux captures



d'écran de l'émulateur et aux journaux générés par LogCat.

Figure II.8: Dalvik Debug Monitor Service.

❖ **III.2.2.5 ADB (Android Debug Bridge) :**

L'ADB est à la fois un client et un service qui nous connecte à un émulateur Android ou un appareil. Il est composé de trois parties :

- Une démo exécutée par l'émulateur ;
- Un service exécuté par notre machine de développement ;
- Des applications clientes (comme le DDMS) qui communiquent avec le démo via le service

L'ADB joue le rôle d'un conduit de communications entre la machine de développement et l'émulateur ou le dispositif Android, ADB permet d'installer des applications, de copier ou lire des fichiers et d'exécuter des commandes shell sur le dispositif cible. Grace au shell du dispositif, nous pouvons modifier les réglages des journaux et interroger ou modifier les bases de données SQL qu'il contient.

IV - Concepts et innovations :

IV.1 Bureaux virtuels (le home):

Le home est l'application principale du système. Elle peut s'apparenter au bureau d'un système d'exploitation pour ordinateur. L'équipe de Google ayant développé le système a aussi développé un home sous licence open source. Le home s'étend en général sur plusieurs parties coulissantes de gauche à droite sur lesquels il est possible d'y placer des raccourcis vers les applications installées et des widgets.

L'image de fond s'étend aussi sur toutes les parties et bouge légèrement quand on change de partie, ce qui donne l'impression que le contenu fait partie du décor. Cependant, certains constructeurs comme HTC modifient ce bureau : l'interface HTC Sense est dotée de 7 bureaux, avec un menu personnalisé. Sony, LG et Samsung profitent aussi du caractère ouvert d'Android pour modifier l'interface Android originelle. Plusieurs surcouches de fabricants ont été développées comme HTC Sense, Samsung TouchWiz, LG Optimus UI ou Sony Ericsson UI.

Figure II.9 : le home.

IV.2 Les Widgets :

Comme pour les bureaux des systèmes d'exploitation récents, il est possible de choisir les éléments qui se trouvent sur l'écran d'accueil. On trouve principalement des icônes, mais les utilisateurs d'Android sont aussi friands de ce qu'on appelle les Widgets,

Une Widget est une application miniature destinée à être utilisée dans une autre application, cette Widget permet d'améliorer une application à peu de frais en lui ajoutant un compagnon permanent. De plus, mettre un Widget sur son écran d'accueil permet à l'utilisateur de se rappeler l'existence de notre produit et par conséquent d'y accéder plus régulièrement. Par ailleurs, jeter un œil rapide sur une application installée et lui accordé un accès direct à certaines fonctionnalités de l'application sans avoir à l'ouvrir, ou même ouvrir l'application ou des portions de l'application.

Parmi les plus utilisées, nous avons donc : la galerie photo, le lecteur musical, l'horloge, la météo, l'agenda, la gestion de batterie etc. Et ce qui est génial, c'est de pouvoir les choisir, les modifier, les placer sur la page de notre choix.

IV.3 Centralisation des notifications :

Figure II.10 : Barre de notification

La barre du haut reçoit les statuts (réseaux utilisés, niveau de batterie, modes vibreur/sonnerie/silencieux, alarme, etc.) et les notifications. Celles-ci proviennent des applications, et peuvent avertir de l'arrivée d'emails, de SMS, d'appels en absence, ou bien afficher des informations relatives à Google Talk, à l'avancement des téléchargements en cours, etc.

Lorsque l'utilisateur glisse son doigt sur cette barre afin de la faire coulisser vers le bas, elle affiche les détails des notifications et permet, lors d'une pression sur ces notifications, l'ouverture de l'application concernée.

IV.4 Capteurs divers :

Android prend en charge des capteurs de type gyroscope, gravité, accélération linéaire ou encore baromètre. Ces capteurs peuvent ouvrir une autre dimension au système Android, pouvant le spécialiser dans des domaines de recherche, ou de sport.

Depuis la version 2.3, Android prend en charge la technologie NFC (Near Field Communication), qui ressemble au RFID (Radio Frequency Identification), pour tout ce qui peut être paiement, ticket de métro, et d'autres applications commerciales. Parmi les appareils équipés d'une puce NFC, on peut citer le Nexus S ou le GalaxyNexus.

IV.5 Fils de discussion :

Les fils de discussion sont de plus en plus utilisés sur les différents modes de communication. Sur Android, ils sont utilisés pour les courriels (comme sur Gmail) et les SMS (comme sur les Sony Ericsson).

IV.6 Android Market/Google Play Store:

Android Market a été remplacée par Google Play Store le 6 mars 2012. Cette nouvelle plate-forme rassemble des services Google déjà existants.

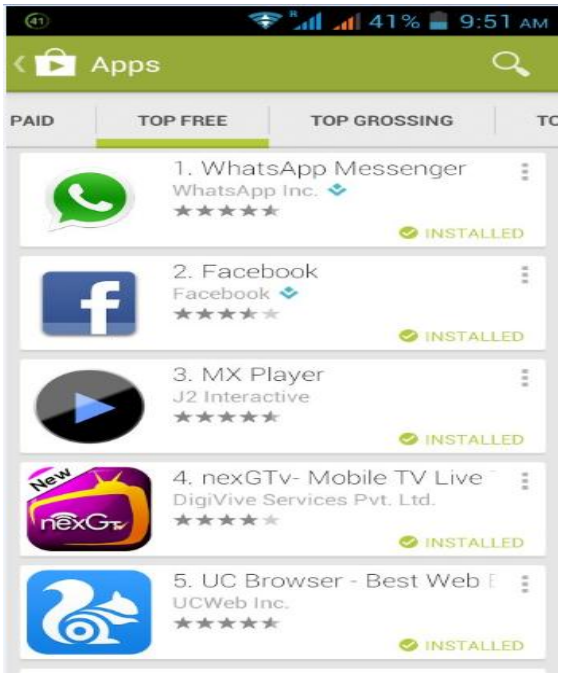


Figure II.11: Google Play Store.

Celui-ci permet de télécharger des logiciels, des livres, des films ou de la musique. Il est aussi possible de les noter et de les commenter. En juillet 2014, il y avait plus de 1.300 .000 applications sur Google Play Store, dont plus d'un million sont gratuites.

L'Android market est aussi le système de publication d'applications officielles d'Android. Néanmoins, contrairement à l'App Store d'Apple, la distribution d'applications au public n'est pas un monopole de Google et techniquement, rien n'empêche d'installer des applications depuis une autre source. D'ailleurs, des alternatives à Android Market, comme par exemple SlideMe, émergent peu à peu.

Les Smartphones disposant d'une puce Tegra, comme Nexus 7 avec une puce Tegra 3, ont accès à la Tegra Zone, un magasin d'applications permettant de télécharger des applications dédiées aux téléphones sous Tegra. Il contient notamment des applications THD (Très Haute Définition).

Pour proposer son application sur Android Market (Play Store), il faudra détenir un compte Google et s'acquitter des frais d'enregistrement qui s'élèvent à 25 dollars.

Bien sûr, l'application devra être parfaitement packagée et signée à l'aide d'un certificat P12, Le système ne peut pas installer une application qui n'est pas signée. Par ailleurs, Android Market requiert que la date de validité du certificat soit postérieure au 22 octobre 2033, qu'une icône et un intitulé soient précisés dans le fichier « AndroidManifest.xml »de l'application (dans l'attribut android : icon et android : label) et que les champs de version (android : versionCode et android : versionName) soient renseignés.

V - Conclusion :

Dans ce chapitre, nous avons fait une étude sur Android tout en présentant un bref historique, l'architecture de la plateforme Android et les éléments du SDK d'Android, afin de voir les fonctionnalités que nous pouvons trouver sur ce système d'exploitation.

Aujourd'hui, un développeur souhaitant se lancer dans la création d'applications pour mobiles a le choix entre trois plateformes : Android, iOS (iPhone), et Windows Mobile. Développer sous Android concède beaucoup d'avantages par rapport à ses concurrents. Comme nous avons pu le voir, il est gratuit, modulaire, open source, et ne nécessite pas d'apprendre un nouveau langage si l'on connaît déjà le Java (contrairement à l'iPhone et son ObjectiveC).

Chapitre 3

ANALYSE ET CONCEPTION.

Introduction:

Avant l'implémentation et la réalisation de toute application informatique, il convient de suivre une démarche méthodologique pour mettre en place cette dernière. Pour réaliser notre application, nous commençons par une analyse profonde et bien réfléchie, suivie d'une conception en se basant sur la modélisation UML.

I. Objectif de l'Application:

Notre travail consiste à développer une application Android pour un restaurant, on a pour objectif de faciliter le travail pour les serveurs, ainsi l'application aura contribué au gain de leurs temps et à limiter leurs déplacements.

II. Processus de développement

. Analyse et conception :

C'est la première étape pour créer notre application, dans laquelle on va définir les acteurs, les tâches qu'ils assurent et les scénarios. Pour cela on a appliqué la méthode d'analyse et de conception UML (Unified Modeling Language).

A. Première étape : Analyse

L'analyse débute par l'examen du modèle des cas d'utilisation et de leurs scénarios, ainsi que les besoins fonctionnels du système, et aboutit à un modèle d'analyse constitué par les classes et les collaborations de classes qui traduisent les comportements dynamiques, détaillés dans les cas d'utilisation et les besoins.

A.1 Présentation de l'UML :

Définition

UML, c'est l'acronyme anglais pour « Unified Modeling Language ». On le traduit par «Langage de modélisation unifié ». La notation UML est un langage visuel constitué d'un ensemble de schémas, appelés des diagrammes, qui donnent chacun une vision différente du projet à traiter. UML nous fournit donc des diagrammes pour représenter le logiciel à développer : son fonctionnement, sa mise en route, les actions susceptibles d'être effectuées par le logiciel, etc… Réaliser ces diagrammes revient donc à modéliser les besoins du logiciel à développer.

Cela dit, on peut tout à fait s'en servir pour décrire de futures applications, sans pour autant déjà être fixé sur le type de développement. C'est exactement ce à quoi UML sert dans des projets de réalisation de logiciels.

A.2 Quelques définitions de base :

Acteur : Un acteur est une entité externe au système, en interaction avec ce dernier.

L'entité est un rôle joué par un utilisateur, par exemple un comptable, par un autre système, un capteur par exemple.

Cas d'utilisation : Un cas d'utilisation spécifie une séquence d'action, avec variantes éventuelles, réalisée par le système en interaction avec des acteurs du système.

Scénario : Un scénario est une succession particulière d'enchainement, s'exécute du début a la fin du cas d'utilisation .Un enchainement étant l'unité de description de séquences d'actions. Un ensemble de scénarios pour un cas d'utilisation identifie tous ce qu'il peut arriver lorsque ce cas d'utilisation est mis en œuvre.

A.3 Identification des acteurs :

Les acteurs de notre application sont :

Le restaurateur (administrateur) : il prend la commande d'un client

Le cuisinier : il prépare la commande

Le caissier : il calcule l addition pour chaque table

A.4 Spécification des tâches et des scénarios:

Le tableau suivant résume les tâches assurées par tous les acteurs de notre application :

Acteur	tache
restaurateur	T1 : Ajouter une commande. T2 : Modifier une commande. T3 : Supprimer une commande. T4 : Valider une commande.
cuisinier	T6 : Gérer le menu. T7 : Préparer la commande.
caissier	T8 : Régler les additions.

Tableau III: Spécification des tâches.

A.5 Les cas d'utilisation :

Définition d'un cas d'utilisation :
Un cas d'utilisation est la description d'un ensemble d'actions, y compris des variantes, qu'un Système exécute et qui produit un résultat observable par un acteur.
Dans notre application, les cas d'utilisation les plus importants

A.6 Relations entre cas d'utilisation :
Pour affiner le diagramme de cas d'utilisation, UML définit trois types de relations standardisées entre cas d'utilisation :
Une relation d'inclusion, formalisée par le mot-clé <<include>> : le cas d'utilisation de base en incorpore explicitement un autre, de façon obligatoire. Une relation d'extension, formalisée par le mot-clé <<extend>> : le cas d'utilisation de base en incorpore implicitement un autre, de façon optionnelle.

Une relation de généralisation/spécialisation : les cas d'utilisation descendants héritent de la description de leur parent commun. Chacun d'entre eux peut néanmoins comprendre des interactions spécifiques supplémentaires.

A.7 Spécification des cas d'utilisation :

Cas d'utilisation : Ajouter une commande.

Use case : Ajouter une commande.
Acteur : Restaurateur.
Description :
1. Le restaurateur accède au bouton commande.
2. Le système lui affiche le contenu du Menu.
3. Le restaurateur sélectionne le choix du client (plat, boisson, dessert) ainsi la table.
4. Le restaurateur valide la commande qui sera envoyée au cuisinier.

Cas d'utilisation : Validation d'une commande.

Use case : Validation d'une commande
Acteur : Cuisinier.
Description :
1. Le Cuisinier accède la liste des commandes.
2. Le Cuisinier prépare la commande.
3. Le Cuisinier valide la commande.

Cas d'utilisation : Modifier une commande.

Use case: Modification d'une commande
Acteur : Restaurateur.
Description :
1. Le Restaurateur accède la liste des commandes.
2. Le Restaurateur sélectionne la commande à modifier.
3. Restaurateur modifie et valide la commande.

Cas d'utilisation : Supprimer une commande.

Use case : Suppression d'une commande
Acteur : Restaurateur.
Description :
1. Le Restaurateur accède la liste des commandes.
2. Le Restaurateur sélectionne a commande a supprimer.
3. Restaurateur supprime et valide.

Cas d'utilisation : Gérer menu.

Use case : Ajouter un plat.
Acteur : Cuisinier.
Description :
1. Le cuisinier accède la liste des menu ensuite gérer plat.
2. Le cuisinier sélectionne ajouter plat.
3. Le cuisinier remplie le formulaire.
4. le cuisinier valide l'ajout.

Cas d'utilisation : Préparer commande.

Use case : Préparer commande.
Acteur : Cuisinier.
Description :
1. Le cuisinier accède la liste des commandes.
2. Le cuisinier sélectionne la commande à préparer.
3. Le cuisinier prépare la commande.

Cas d'utilisation : Régler les additions.

Use case : Régler les additions.
Acteur : Caissier.
Description :
1. Le caissier accède la liste des commandes.
2. Le caissier sélectionne la commande à encaisser.
3. Le caissier encaisse et supprime la commande.

A.8 Diagramme de cas d'utilisation :

La relation « include » :

Une relation d'inclusion d'un cas d'utilisation A par rapport à un cas d'utilisation B signifie qu'une instance de A contient le comportement décrit en B, le cas d'utilisation A ne peut être utilisé seul.

La relation « extend » :

Une relation d'extension d'un cas d'utilisation A par rapport à un cas d'utilisation B signifie qu'une instance de A peut être étendue par le comportement décrit en B.

La relation « Use » :

Lorsqu'une ou plusieurs taches sont utilisées régulièrement, on peut les factoriser dans un même use case et faire de telle sorte que d'autre uses cases l'utilise en pointant par une flèche.

Les diagrammes des cas d'utilisation de notre système pour chaque acteur sont

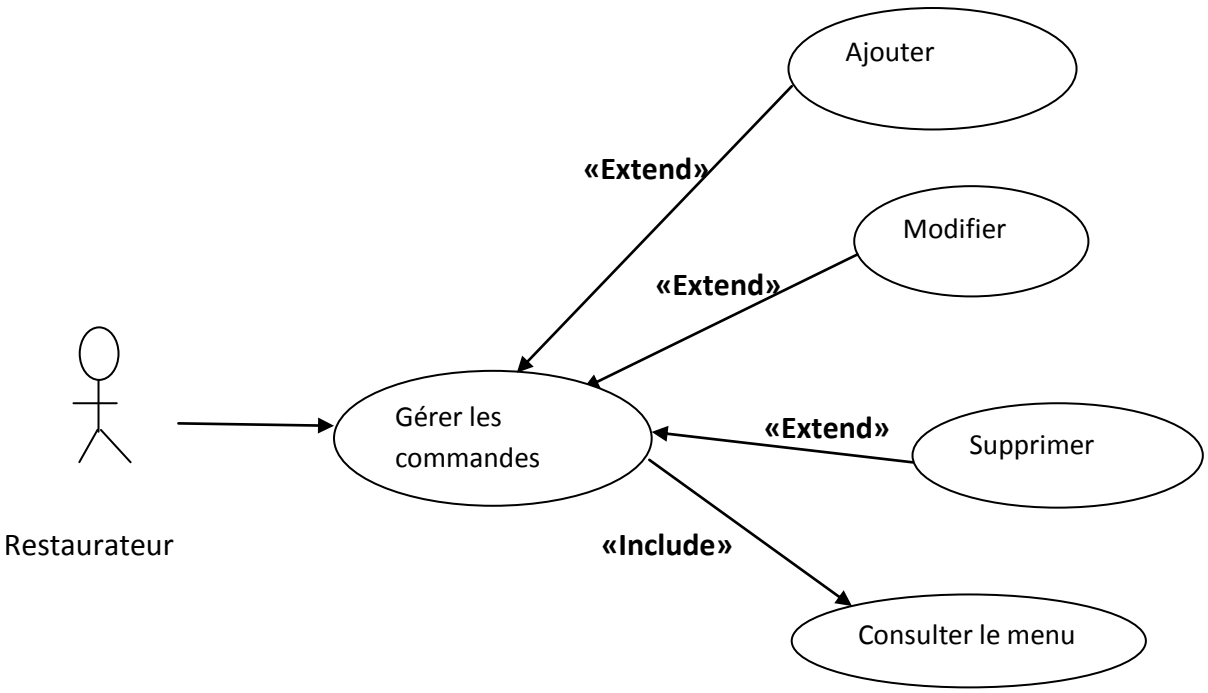


Figure III.1: Diagramme des cas d'utilisations pour le restaurateur.

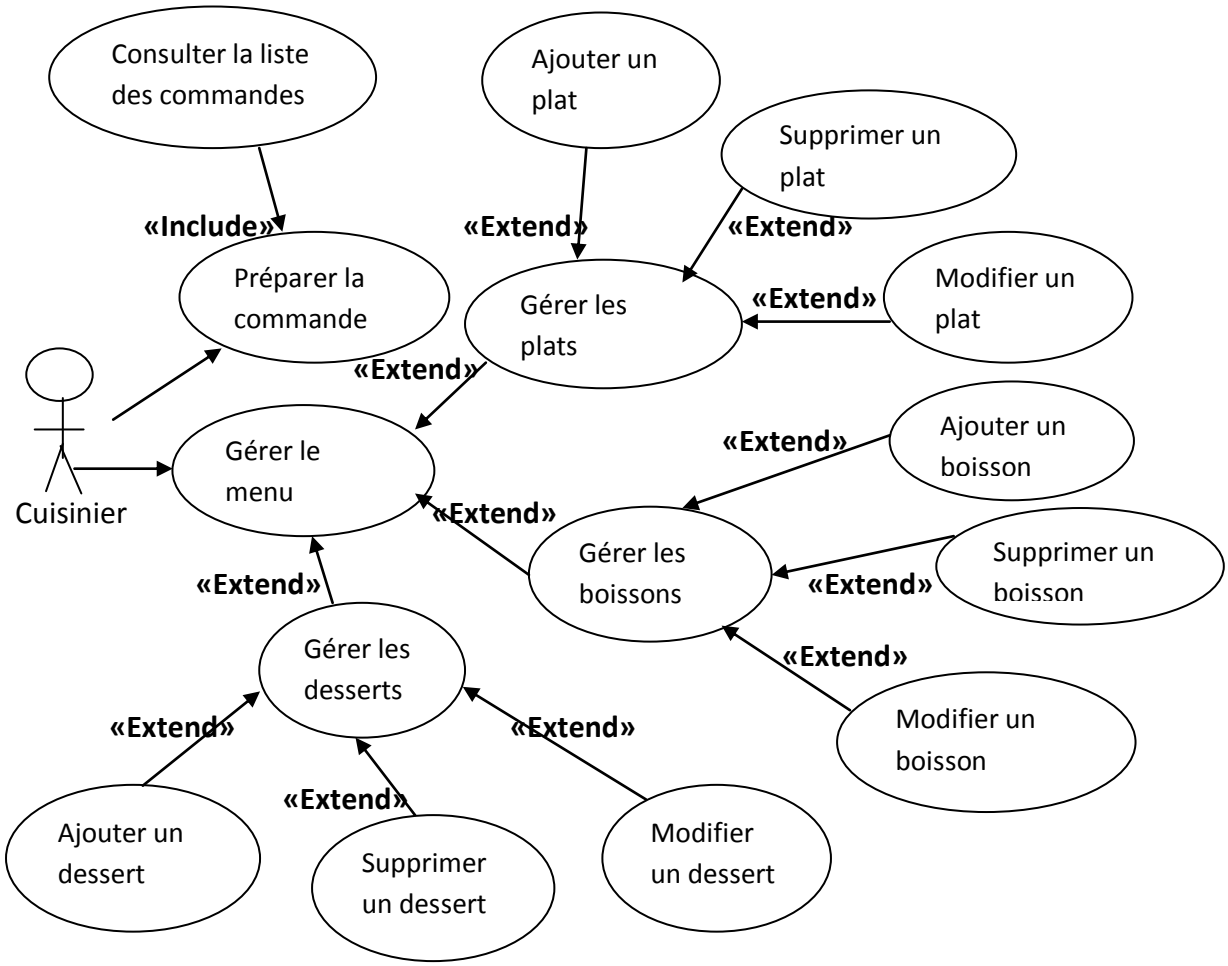


Diagramme des cas d'utilisations pour le cuisinier.

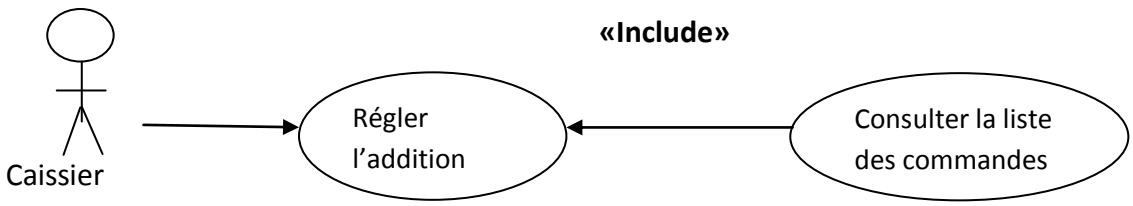


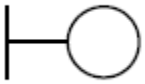
Figure III.3: Diagramme des cas d'utilisations pour le caissier.

B. Deuxième étape : Conception:

La conception est la dernière étape avant la réalisation technique de la plate forme de l'application, elle consiste donc à réaliser le modèle qui va être implémenté, un modèle ou les contenus sont classés et hiérarchisés. on apporte plus de détails à la solution et on cherche à clarifier des aspects techniques,
Tout comme l'étape d'analyse, UML utilise différents diagrammes pour la conception, et dans notre cas on va élaborer des diagrammes de séquence ainsi que les diagrammes de classes pour les cas d'utilisation étudié et un diagramme de classe générale.

B.1 Diagramme de séquence :
L'ensemble des classes d'analyse peut être scindé en trois (03) catégories d'objet :

1. Les objets interface :
Un objet interface représente l'interface entre l'acteur et le système.
L'icône :



2. Les objets entités :
Ils sont des objets décrits dans un cas d'utilisation et qui se trouvent dans d'autre cas d'utilisation tels que le plat, la commande.
L'icône :

3. Les objets de contrôle :
Ils représentent les activités des processus du système.
Ils dirigent les activités des objets entités et interface. Ces objets sont obtenus en extrayant les verbes des cas d'utilisation.
L'icône :

Les diagrammes qui suivent obéissent aux règles suivantes :
Les acteurs n'interagissent qu'avec les objets interfaces.
Les objets contrôles interagissent avec les autres objets y compris les instances d'objets contrôles.
Les objets entités n'interagissent qu'avec des objets contrôles.
Nous allons présenter l'analyse de quelques cas d'utilisation :

1. Modifier un plat.
2. Ajouter une boisson
3. Ajouter une commande.
4. Supprimer la commande encaissée.

Cas d'utilisation «Ajouter une commande » :

Ce cas d'utilisation contient les objets suivant :

Objet interfaces :

Fenêtre commande.

Le formulaire d'ajout de commande.

La fenêtre de confirmation.

Objet entités :

Commande.

Objet contrôle :

Valider.

Diagramme de séquence :

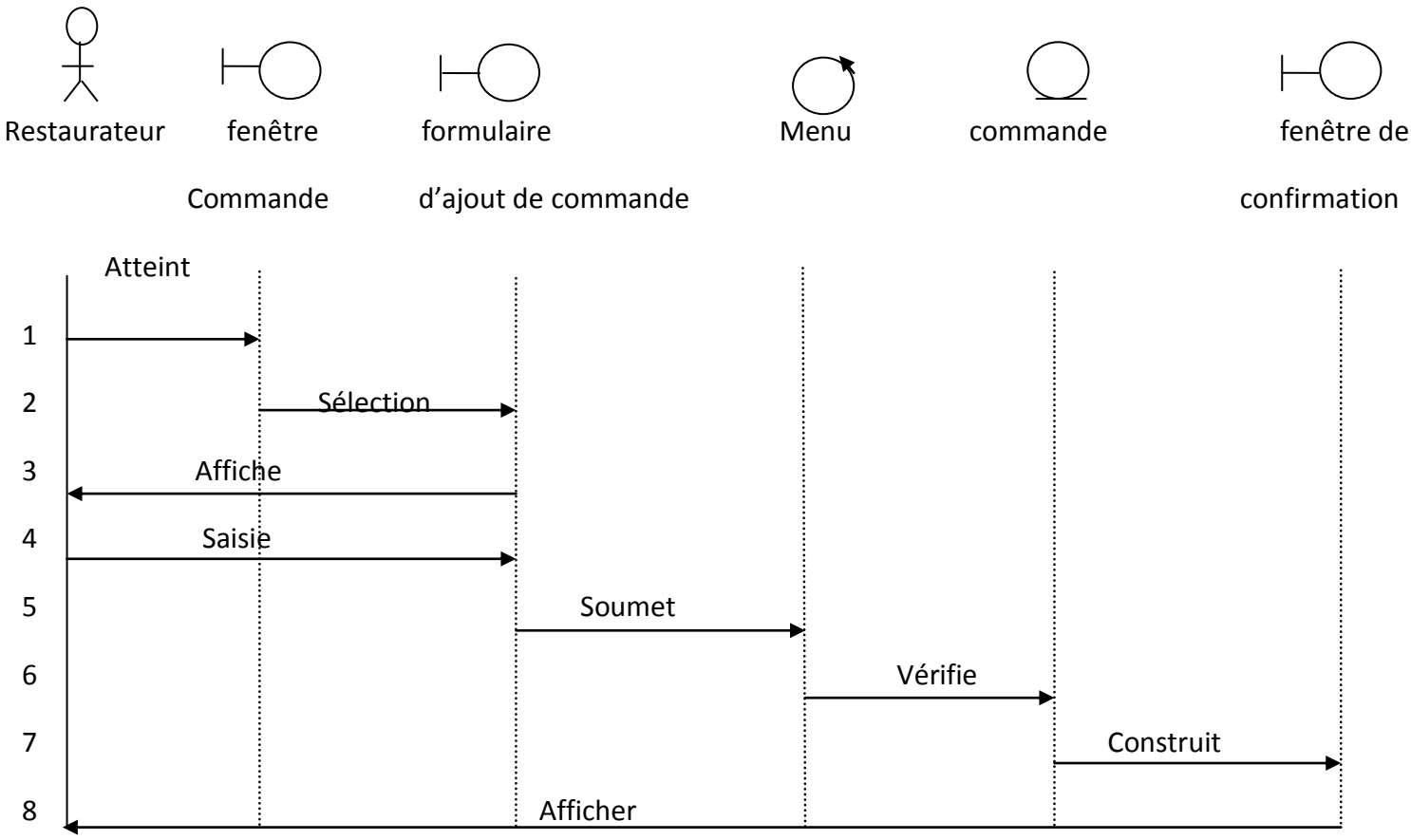


Figure III.4: Diagramme de séquence du cas d'utilisation «Ajouter une commande».

1. Le restaurateur atteint la fenêtre « commande » puis sélectionne « nouvelle commande ».
2. Le système lui affiche le formulaire d'ajout.
3. Le restaurateur choisit et valide.
4. Le système construit un message de confirmation.

Cas d'utilisation « Ajouter une boisson » :
Ce cas d'utilisation contient les objets suivants :
Objets interface :
La fenêtre des boissons.
Le formulaire d'ajout d'une boisson.
La fenêtre de confirmation
Objet entités.
Boisson.
Objet contrôles :
Valider
Diagramme de séquence :

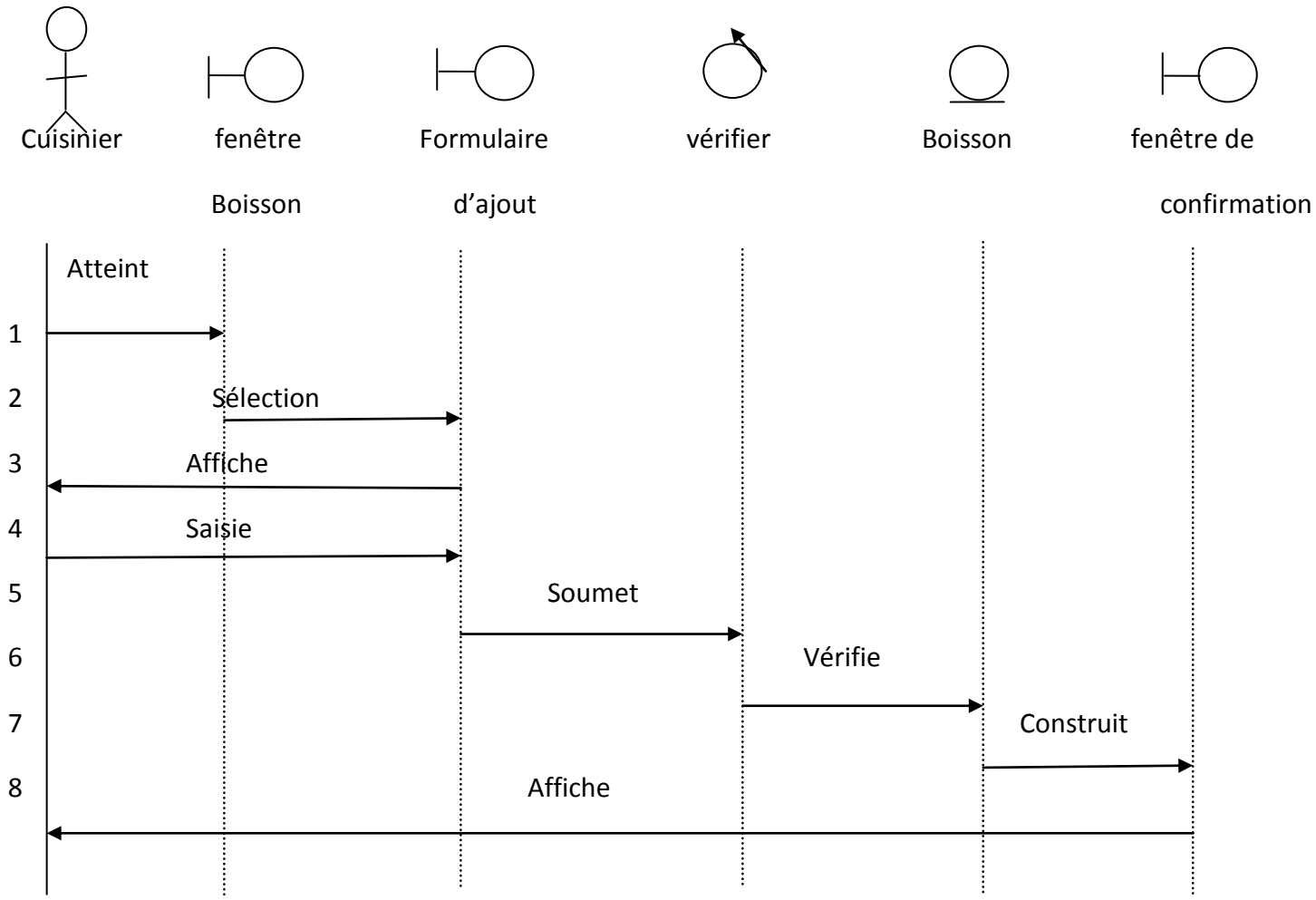


Figure III.5: Diagramme de séquence du cas d'utilisation «Ajouter une boisson».

1. Le cuisinier atteint la fenêtre boisson puis sélectionne« nouvelle boisson ».
2. Le système lui affiche le formulaire d'ajout.
3. Le cuisinier remplie et valide.

Cas d'utilisation «Préparer la commande» :

Ce cas d'utilisation contient les objets suivant :

Objet interfaces :

La fenêtre commande.
La fenêtre liste des commandes.
La fenêtre détail commande.

Objet entités :

Commande.

Objet contrôle :

Diagramme de séquence :

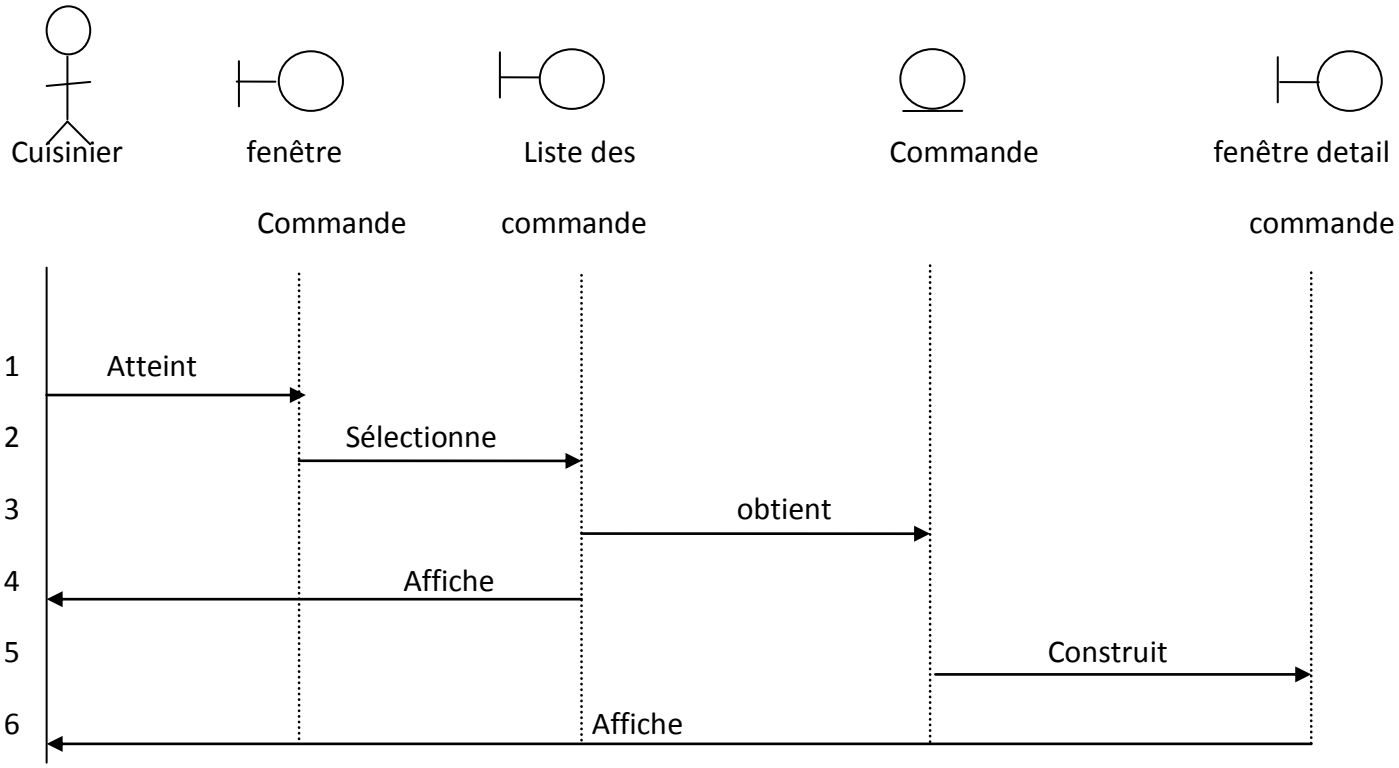


Figure III.6: Diagramme de séquence du cas d'utilisation «Préparer la commande».

1. Le cuisinier atteint la fenêtre commande puis sélectionne liste des commande.
2. Le système lui affiche la liste des commandes.
3. Le cuisinier sélectionne la commande à préparer.
4. Le système affiche la commande en détail.

Cas d'utilisation «Régler l'addition» :

Ce cas d'utilisation contient les objets suivant :

Objet interfaces :

Fenêtre commande.

La fenêtre de la liste des commandes.

La fenêtre détail de la commande.

Objet entités :

Commande.

Objet contrôle :

Diagramme de séquence :

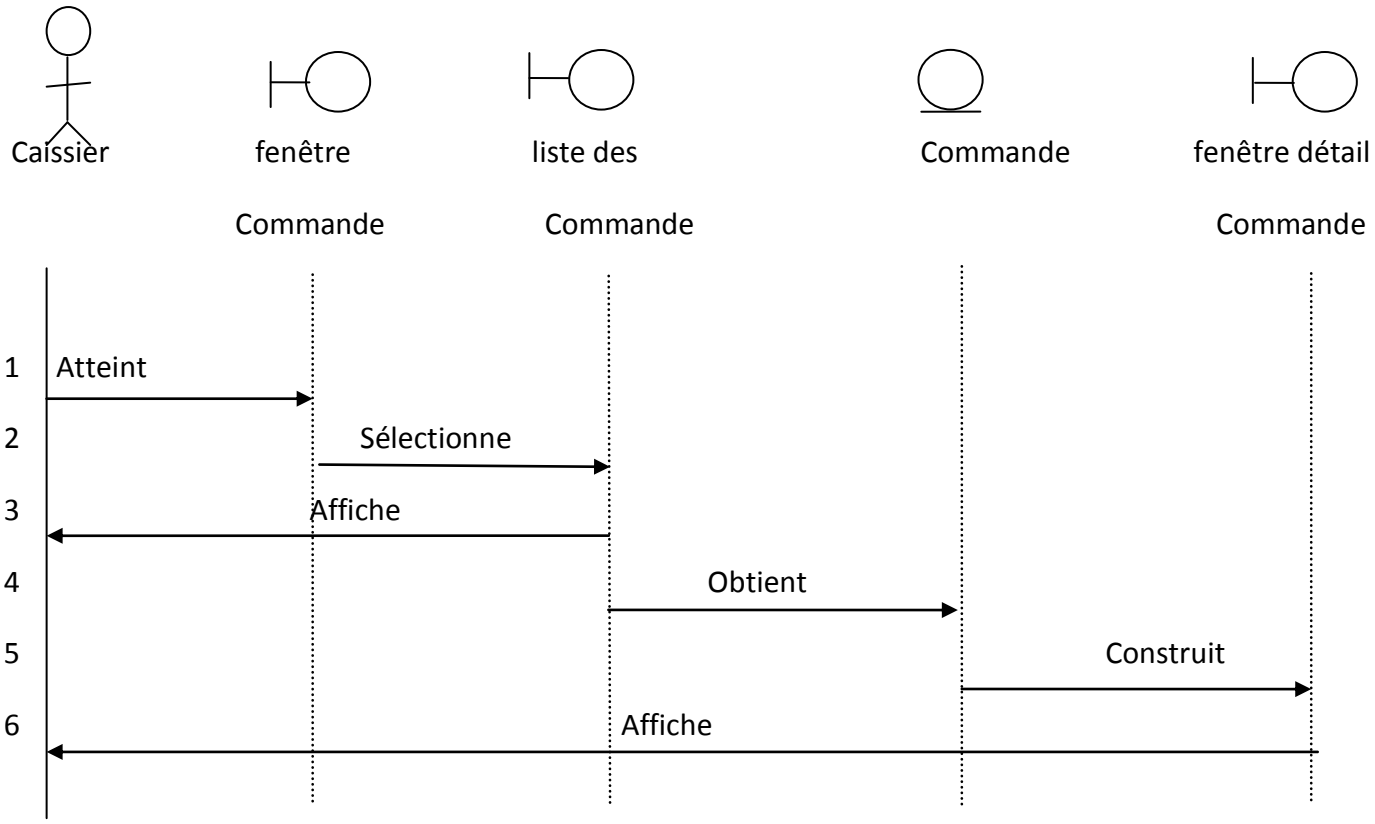


Figure III.7: Diagramme de séquence du cas d'utilisation «Régler l'addition».

1. Le caissier atteint la fenêtre commande puis sélectionne la liste des commandes.

2. Le système lui affiche la liste des commandes.

3. Le caissier sélectionne la commande à régler.

B.2 Diagrammes de Classes (Class Diagram):

Le diagramme de classes est considéré comme le plus important de la modélisation orientée objet, il est le seul obligatoire lors d'une telle modélisation.
Alors que le diagramme de cas d'utilisation montre un système du point de vue des acteurs, le diagramme de classes en montre la structure interne. Il permet de fournir une représentation abstraite des objets du système qui vont interagir ensemble pour réaliser les cas d'utilisation.
Il s'agit donc d'une vue purement statique car on ne tient pas compte du facteur temporel dans le comportement du système. Le diagramme de classes modélise les concepts du domaine d'application ainsi que les concepts internes créés de toutes pièces dans le cadre de l'implémentation d'une application. Il permet aussi de modéliser les classes du système et leurs relations indépendamment d'un langage de programmation particulier.

- **Diagramme de classe du cas d'utilisation «Préparer la commande» :**

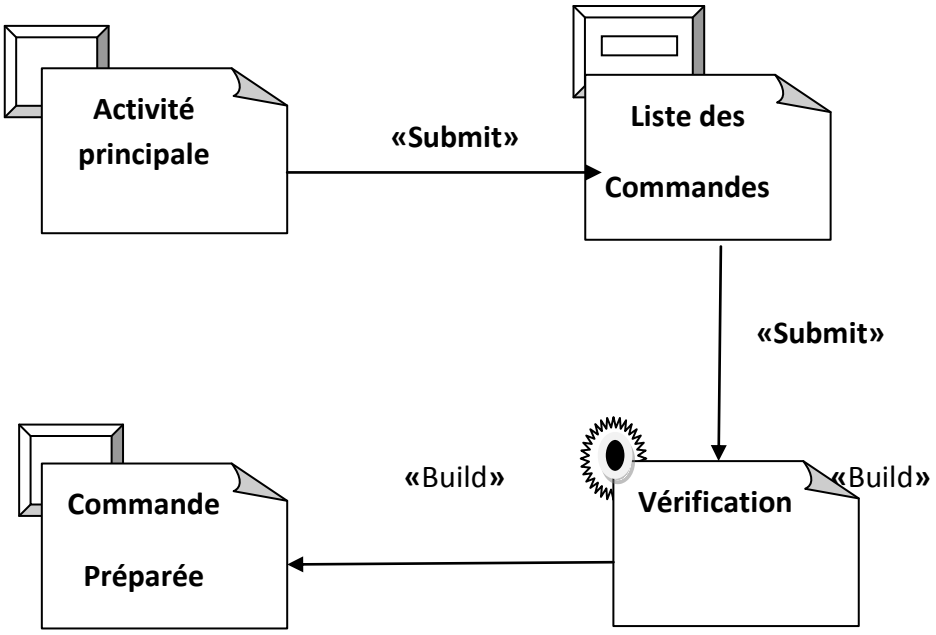


Figure III.8 : Diagramme de classe du cas d'utilisation «Préparer la commande»

- **Diagramme de classe du cas d'utilisation «Ajouter une boisson » :**

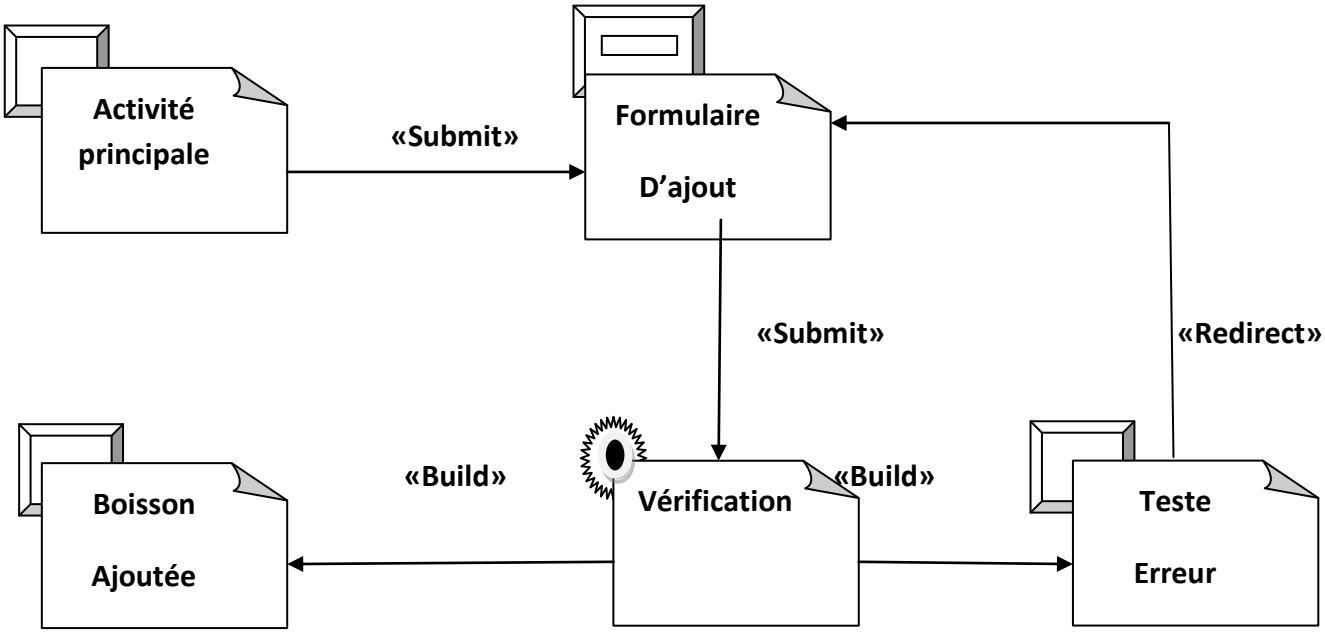


Figure III.9 : Diagramme de classe du cas d'utilisation «Ajouter une boisson »

- **Diagramme de classe du cas d'utilisation «Ajouter une commande » :**

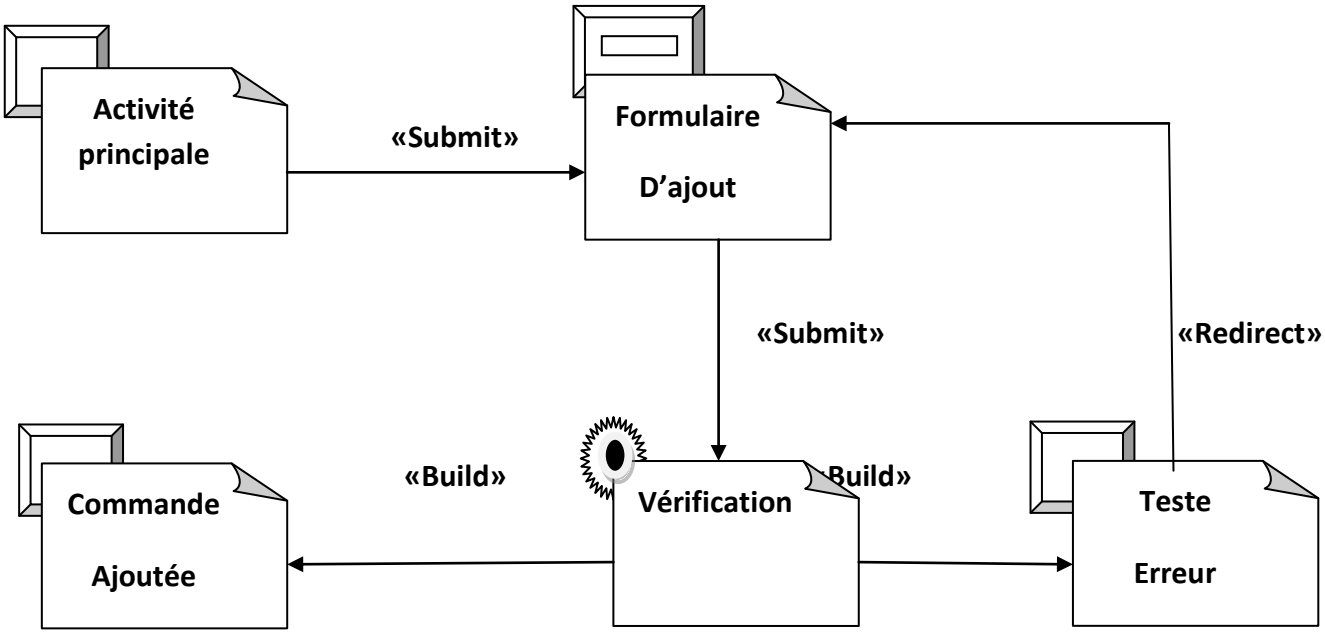


Figure III.10 : Diagramme de classe du cas d'utilisation «Ajouter une commande »

- **Diagramme de classe du cas d'utilisation «Régler l'addition» :**

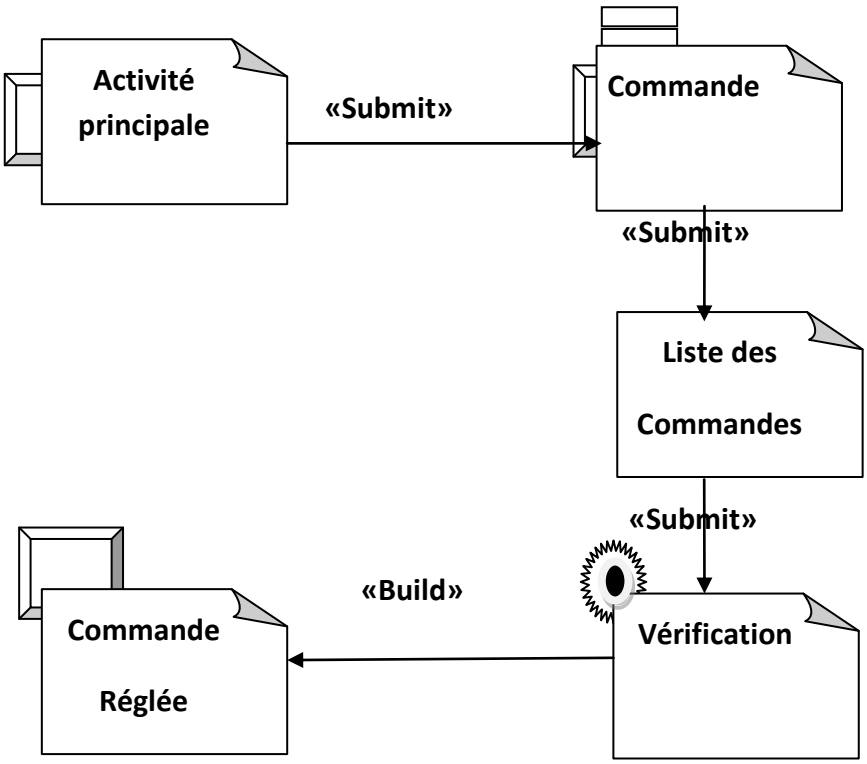


Figure III.11 : Diagramme de classe du cas d'utilisation «Régler l'addition»

B.3 Diagramme de classe globale :

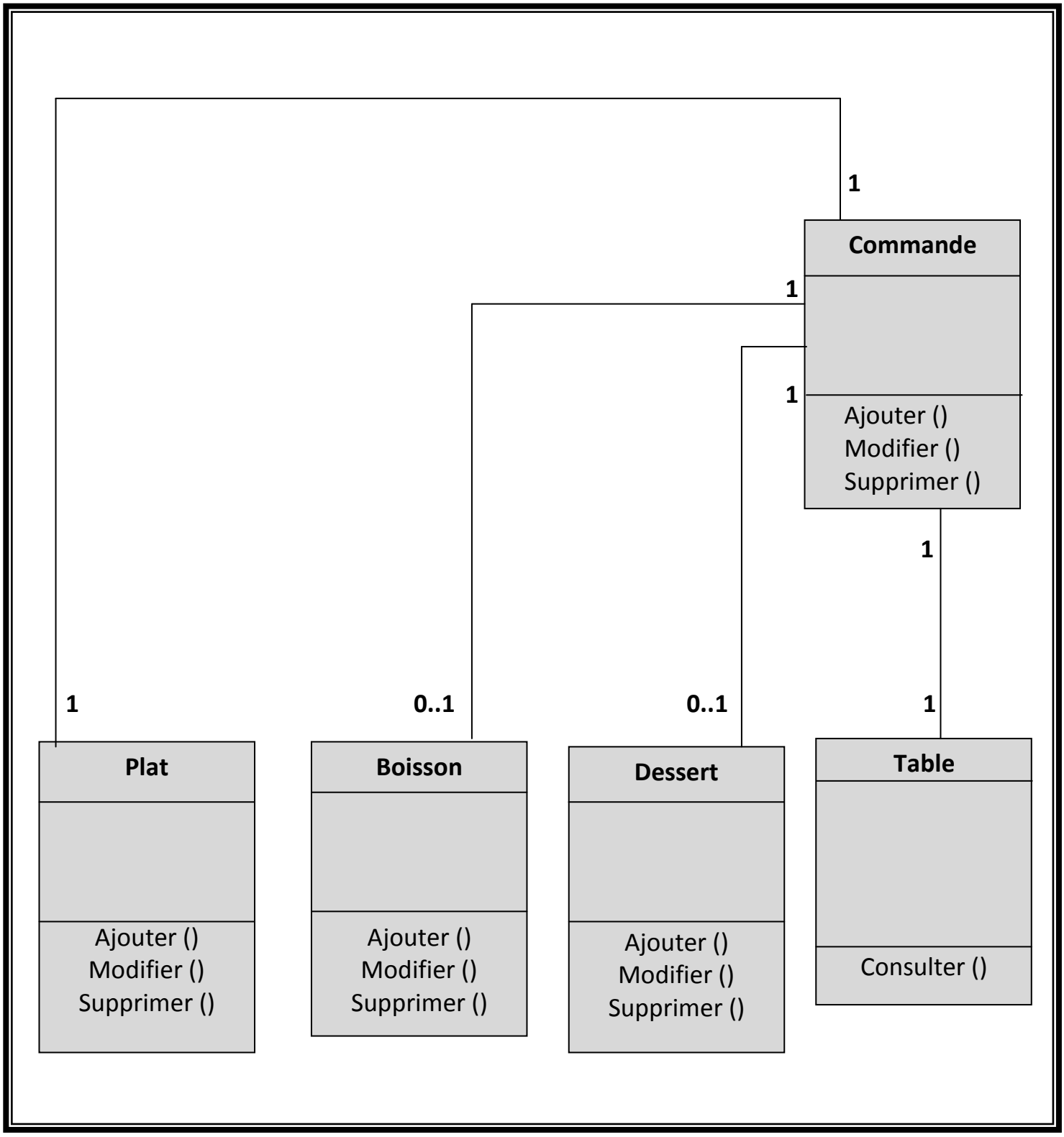


Figure III .12 : Diagramme de classe globale.

B.4 Règles de passage du diagramme de classe au MLD:

Dans ce qui suit, nous allons présenter les différentes règles de passages, qui nous ont servis lors de l'élaboration du modèle logique des données.

· Affecter une table à chaque classe.

· Une association « un à plusieurs » engendre la migration de la clé primaire de la table mère à la table fille.

· Une association « plusieurs à plusieurs » est représentée par une table ayant pour clé primaire la concaténation des clés primaires des deux tables associées.

B.5 Modèle relationnel :

Le modèle relationnel représente l'information dans une collection de relations, et dans notre cas, et dans ce qui suit, on va déterminer l'ensemble des informations manipulées par notre application.

Plat (IDplat, descriptionplat , Photo , prixplat , quantiteplat)
Commande (**IDcom** , IDplat , idboisson ,IDdes ,IDtable ,Total)
Boisson (**idboisson,** libellboisson , typeBoisson , prixboisson).
Dessert (**IDdes ,** descriptiondes , photodes , Prixdes , Quantitedes)
Table (**IDtable,** nbrpersonne)

B.6 Codification :

Toutes les informations manipulées par notre application doivent être stockées dans différentes tables d'une base de données, les noms attribués à ces tables et à leurs différents champs doivent respecter certaines règles.

Table «Plat» :

Nom	*Type de données*	*Description*	*Clef*	*Observation*
IDplat	Integer	Identificateur du plat	primaire	
descriptionplat	Text	Description du plat		
Photo	Image	Photo du plat		
prixplat	Reel	Le prix du plat		
quantiteplat	Integer	La quantité disponible		

Table «Commande» :

Nom	*Type de données*	*Description*	*Clef*	*Observation*
IDcom	Integer	Identificateur de la commande	primaire	
IDplat	Integer	Identificateur du plat	Etrangère	
idboisson	Integer	Identificateur de la boisson	Etrangère	
IDdes	Integer	Identificateur du dessert	Etrangère	
IDtable	Integer	Identificateur de la table	Etrangère	
Total	Reel	Le total a payer		

Table «Boisson» :

Nom	*Type de données*	*Description*	*Clef*	*Observation*
idboisson	Integer	Identificateur de la boisson	primaire	
libellboisson	Text	Nom de la boisson		
typeBoisson	Text	Type de boisson		
prixboisson	Reel	Le prix de la boisson		

Table «Dessert» :

Nom	*Type de données*	*Description*	*Clef*	*Observation*
IDdes	Integer	Identificateur du dessert	primaire	
descriptiondes	Text	Description du dessert		
photodes	Image	Photo du dessert		
Prixdes	Reel	Le prix du dessert		
Quantitedes	Integer	La quantité disponible		

Table «Table» :

Nom	*Type de données*	*Description*	*Clef*	*Observation*
IDtable	Integer	Identificateur de la table	primaire	
nbrpersonne	Text	Nombre de personne		

Conclusion :

Dans ce chapitre, nous avons présenté une démarche de modélisation pour développer notre application. Nous avons d'abord commencé par la spécification du cas d'utilisation dans un premier temps, suivi avec une étude de conception de notre application.

La réalisation du projet sera détaillée dans le prochain chapitre.

Chapitre 4

REALISATION.

I. Introduction :

La réalisation de notre application s'appui sur les étapes d'analyse et conception que nous avons présenté précédemment. Nous structurons ce chapitre ainsi : nous présentons d'abords les outils utilisés pour mettre en œuvre notre application à savoir :Windev mobile et le serveur HFql.

Enfin nous donnons quelques interfaces du site développé.

WinDev Mobile :

WinDev Mobile est un AGL (Atelier de génie logiciel) il vous permet de développer des applications dans tous les domaines.

WinDev Mobile est un outil de développement complet qui intègre tous les outils nécessaires au cycle de réalisation d'une application.

Contrairement a d'autres langages développement traditionnel il n'est pas nécessaire de chercher et d'ajouter des modules pour pouvoir concevoir, tester, installer une application.

Le L5G (langage 5éme génération) de WinDev Mobile, le Wlangage, vous étonnera par sa simplicité.

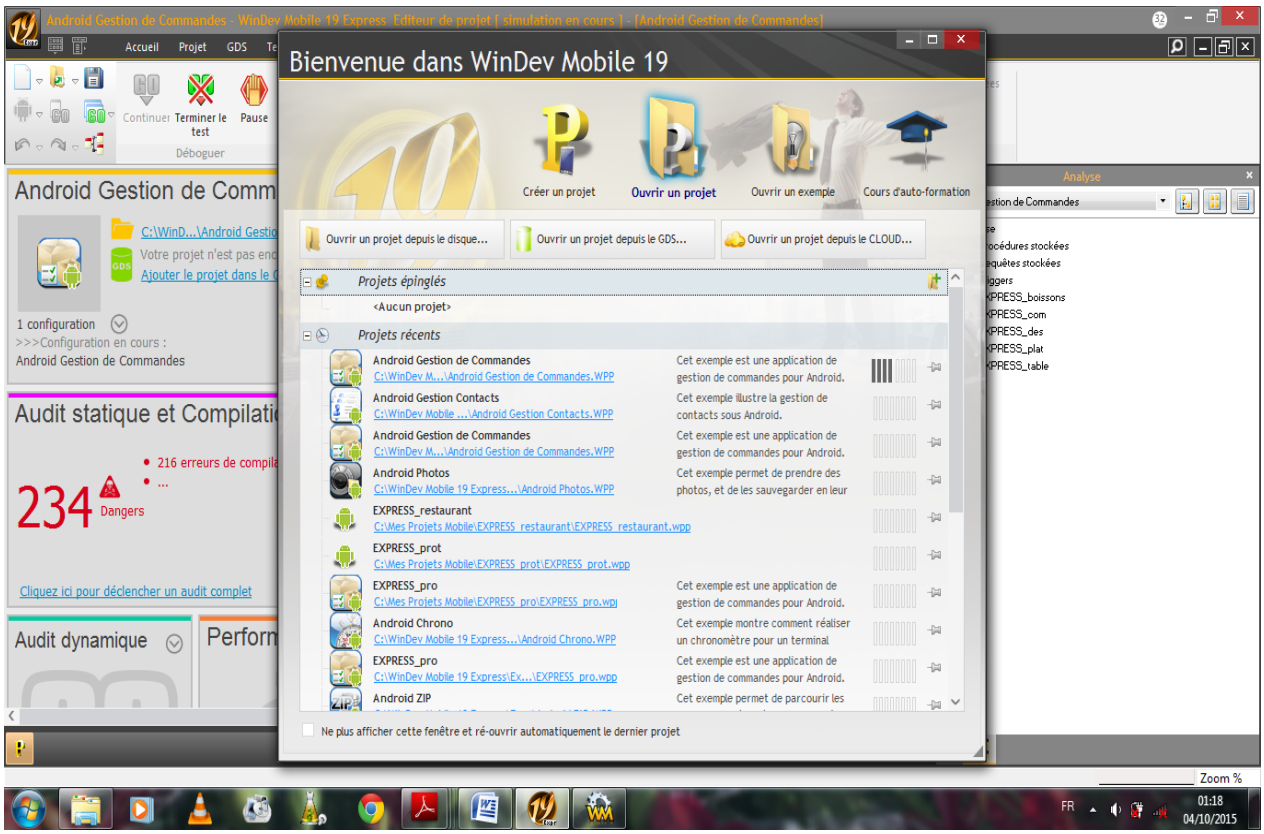


Figure IV.1: Interface Windev mobile

HFSQL

Sous Android deux types de base de données sont disponible :

- ✓ HFSQL (mode classique et mode client/serveur)
- ✓ SQLite

Base de données type HFSQL

- **HFSQL classique**

En mode HFSQL classique les fichiers de données sont stockés sur le matériel (Smartphone ou tablette).Dans ce cas l'application est autonome il n'est pas nécessaire de d avoir une connexion WIFI ou 3G

Les données sont stocké dans la mémoire du matériel la taille maximal de stockage dépond de la capacité mémoire de matériel

- **HFSQL client/serveur**

En mode HFSQL serveur aucune donnée n'est stocké sur le materiel les donnée sont stockées sur une machine sur la quelle est stocké un server HFSQL

Pour accéder a cette machine est donc la base de données il est nécessaire d'avoir activé un moyen de communication avec le server dans l'application mobile WIFI ou 3G afin de se connecter via le réseau ou internet

Les temps de réponse vont bien entendu dépondre de la qualité du réseau WIFI ou internet et du volume des données demandés

L'accès aux données se fera a l'aide des fonctions Wlangage Hxxx et /ou de requêtes SQL

- **Base de données SQlit :**

SQlit est un format de base de données libre reconnu sur un système android cette base de données est stocké dans la mémoire de l'appareil elle est accessible depuis windev mobile, via l accès natif SQlit

L'accès aux bases de données se fait a l aide des fonctions Wlangage Hxxx et/ou requêtes SQL.

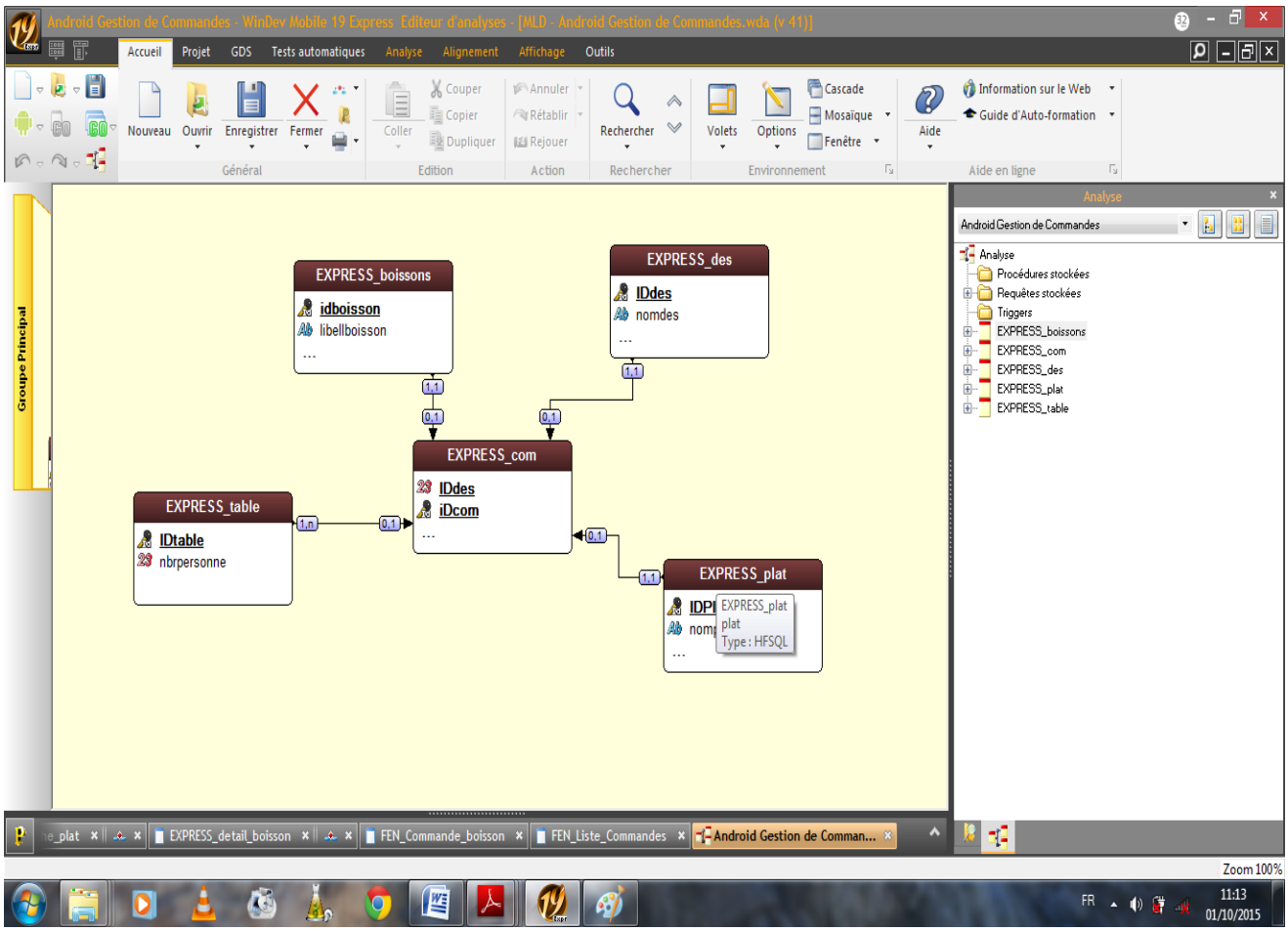


Figure IV.2: Interface Hfsql

Présentation de quelques interfaces de notre application :

Figure IV.3: Fenêtre d'accueil

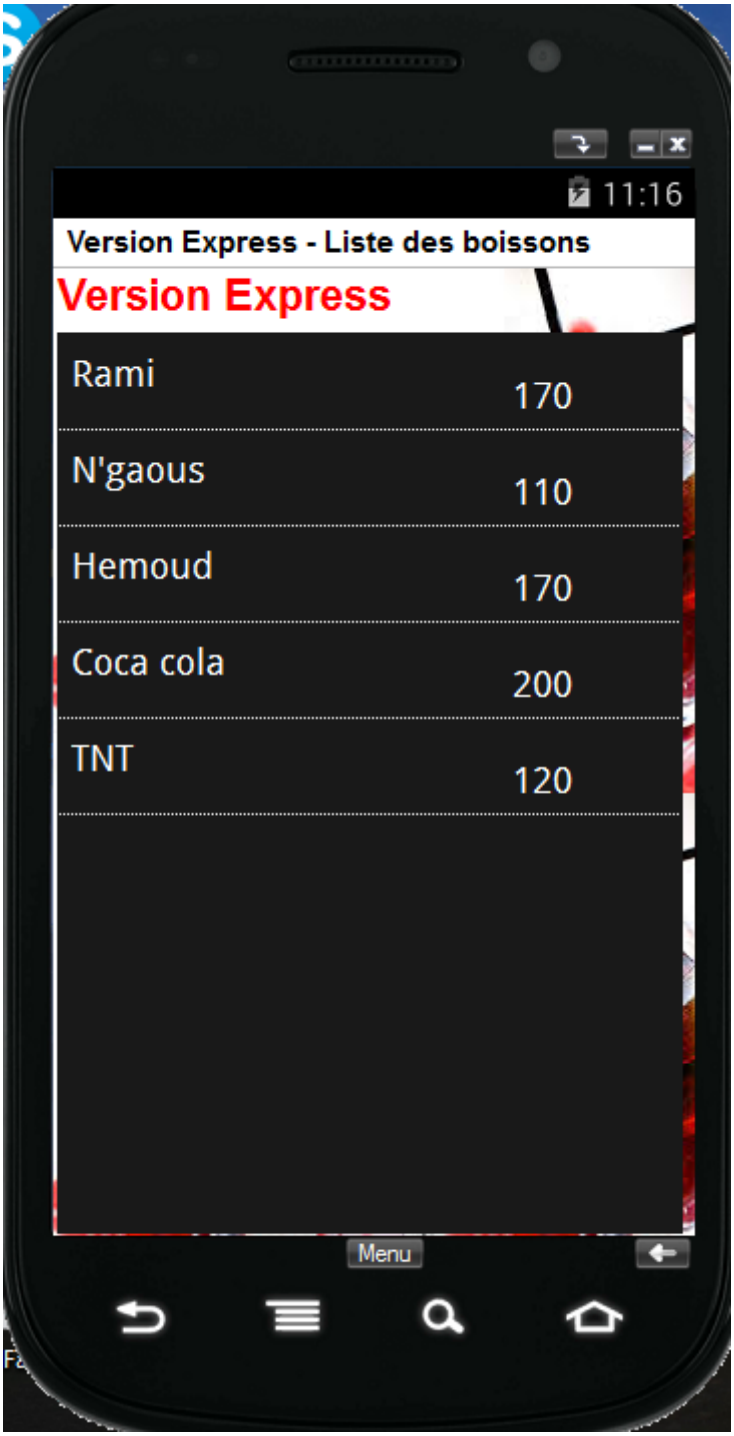


Figure IV.4 : Liste des boissons

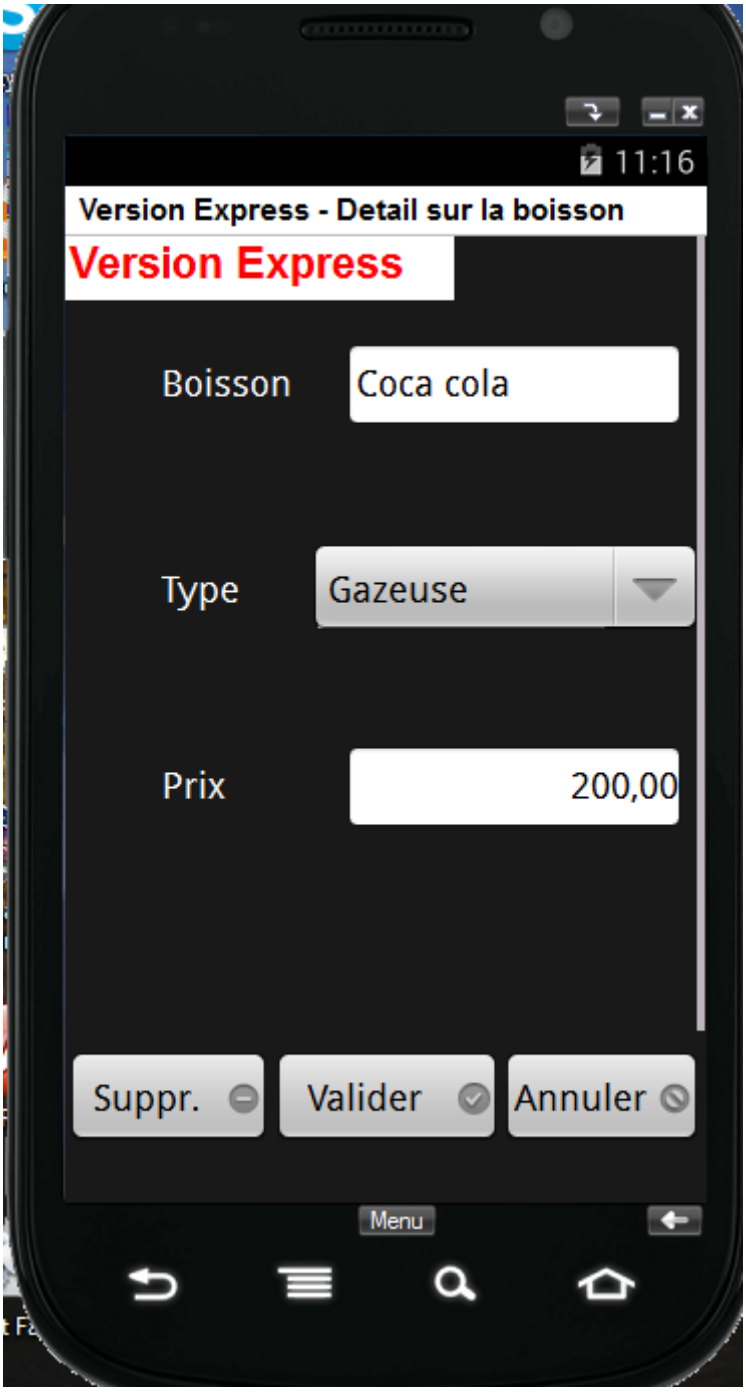


Figure IV.5: Détail boisson

Figure IV.6 : Ajouter une commande

Conclusion :

Ce dernier chapitre a été consacré à la présentation de l'étape réalisation de notre application. Ainsi, nous avons présenté l'environnement et les outils utilisés, pour implémenter notre projet, les langages de programmation et les logiciels de création d'application Android. Puis le fonctionnement de notre application a été décrit en présentant plusieurs interfaces commentées.

Conclusion générale

Au cours de ce projet, nous avons développé une application de commande à distance utilisant la technologie Android, technologie très prometteuse et très populaire avec laquelle un très grand nombre d'applications peuvent être envisagées, que ce soit sur des tablettes ou sur des terminaux mobiles compatibles andoid. L'objectif de notre travail est la création d'une application servant à faire une commande dans un restaurant. Cette application est destinée à être déployée sur un téléphone portable compatible Android. Pour montrer l'interaction de notre application avec ce système nous avons simulé notre application sur un smartphone. Ce travail nous a permis de faire un premier pas dans le monde d'Andoid, en particulier la plate-forme Windev mobile, qui est destinée aux terminaux mobiles à faible puissance, comme les téléphones portables et les assistants personnels (PDA), et pour nous, c'est quelque chose de totalement nouveau.. En ce qui concerne le terminal mobile, notre application peut être utilisée sur tout type de smartphne et tablette.

[GUY 05] : « Les réseaux informatique », Edition 2005, [GUY PUJOLLE].

[TAN 99] : [ANDREW TANNENBAUM, LES RESEAUX, Prentice Hall, 1999].

[FRL] : « cours d'introduction à TCP/IP »,[FRANCOIS LAISSUS].

[DIG 02] : [DI GALLO Frédéric], « intégration des systèmes client/serveur », cours du cycle d'approfondissement, CNAM Aix-en-Provence 2001-2002

[HAR 99] : Robert RFALI, Dan HARKEY et Jery EDWARDS, traduction de François

LEROY et Jean-Pierre GOUT, Client/serveur guide de service. Edition original : JOHN WILLY&SONS, INC 1999. Traduction : Vuilbert 1999

[GRB 00] : « Guide de l'utilisateur UML 2000) » GRADY BOOCH, JAMES RUMBAUGH

[JOG 02]: « UML pour la modélisation des systèmes d'information »JOSEPH GABAY 2002

Webliographie:

[1] http://forum.pcsoft.fr/group.awp?forum=pcsoft.fr.windevmobile 15/09/2015

[2] http://www.pcsoft.fr/st/telec/index.html 03/09/2015

[3] http://fr.wikipedia.org/wiki/WinDev_Mobile 03/09/2015

[4] http://openclassrooms.com/courses?q=uml 12/06/2015

[6] www.learningtree.fr/courses/fr323.htm 04/06/2015