République algérienne démocratique et populaire

Ministère de l'enseignement supérieur et de la recherche scientifique

UNIVERSITE SAAD DAHLEB BLIDA -1-

Faculté de science

Département d'informatique

Mémoire Master Ingénierie Logiciel

La Réalisation des Adaptateurs de médiation pour la composition Hétérogène des Applications Mobiles

Réalisé par :

- ✓ M elle ZARZAR SALIHA
- ✓ Melle RAIS SARAH

Encadré par :

Mme DJADAR AFRAH

Soutenu devant le jury composé de :

Présidente :	Mme ABED	Professeur	Université de Blida 1
Examinatrice :	Mme ARKAM MERIEM	MAA	Université de Blida 1
Promotrice :	Mme DJEDDAR AFFRAH	MCB	Université de Blida 1

Promotion : 2018/2019

Remerciements

Après avoir rendu grâce à « Allah » le tout puissant et miséricordieux, qui nous a donné la force et la patience d'accomplir ce modeste travail, Nous tenons à remercier vivement tous ceux qui, de près ou de loin ont participé à la rédaction de ce mémoire. Il s'agit plus particulièrement de :

Madame Djeddar .A notre promotrice pour sa disponibilité, sa rigueur scientifique, son précieux conseil et son sens d'écoute et d'échange durant toute la période du travail.

Nos vifs remerciements vont également aux membres de jury pour l'intérêt qu'ils ont porté à notre recherche en acceptant d'examiner notre travail et de l'enrichir par leurs propositions.

Enfin, nous tenons également à remercier toutes les personnes qui ont participé à la réalisation de ce modeste travail.

Dédicace

Je dédie ce mémoire

A ma très chère ***Maman Amel***

A mon très cher ***Papa Boualem***

Aucune ne dédicace, aucun mot ne pourrait exprimer à leur juste valeur la gratitude et que je vous porte.

Je mets entre vos mais, le fruit de longues années d'études, de votre amour et de votre tendresse, de longs jours d'apprentissage.

Chaque ligne de ce mémoire chaque mot et chaque lettre vous exprime la reconnaissance, le respect, l'estime et le merci d'être mes parents

A mes sœurs **Maroua** *et* ***Kawther***

A mon fiancé Lotfi

A ma chère tante ***Sabrina***

A ***ma grand-mère****, que dieu la couvre dans son paradis*

A ma grand-mère ***Saliha***

A ***Sara****, ma chère amie et mon binôme*

Je vous dédie ce travail en témoignage des liens solides et intimes qui nous unissent et pour leurs soutiens, encouragements en vous souhaitant un avenir plein de succès et de bonheur.

SALIHA

Dédicace

Je dédie ce travail comme preuve de respect, de gratitude, et de reconnaissance :

A toute ma famille, en particulier à mes chers parents et mon mari;

Pour m'avoir toujours soutenue et encouragée à chaque étape de ma vie.

A ma sœur et mon frère ;

A qui j'ai mené la vie dure durant la réalisation de mon travail.

A toutes mes amies ;

Mes amitiés les plus sincères ainsi que mes souhaits de réussite et de prospérité…

Sarah KAIS

Résumé :

Le nombre ainsi que les différents types d'applications qui sont disponibles sur une variété de plateformes pour les appareils mobiles sont indéniablement explosifs. Ce phénomène est susceptible d'encourager l'utilisateur d'un appareil mobile d'attendre de nouvelles applications selon ses propres besoins en se basant sur celles qui sont déjà existantes où la même application obtenue soit disponible sur à peu près tout type de plateformes même quand il y a des différences entre les plateformes ainsi que dans les applications qui fonctionnent sur ces plateformes- à savoir personnaliser le comportement de l'application souhaitée selon les différents environnements d'exécution et les besoins des utilisateurs.

Pour cause des exigences des utilisateurs et aussi du contexte de l'appareil mobile cible, un développeur se trouve parfois obligé de combiner des entités logicielles hétérogènes (différentes forme d'implémentation : composants, service..etc). Cependant, composer les entités logicielles contextuelles qui sont correspondantes aux besoins identifiés conduit forcément à traiter plusieurs types de composition en terme de collaborations - hétérogènes/homogènes.

L'objectif de ce travail est de réaliser les médiateurs d'adaptation nécessaires pour éliminer cette hétérogénéité et donc permettre une composition consistante. Faire face à la coordination hétérogène, nécessite l'intégration des médiateurs exogènes en fournissant la structure concrète d'une nouvelle entité *Component-of-services* ainsi que les médiateurs endogènes en fournissant la structure concrète au service de médiation proposé. Ce dernier nécessite de construire une bibliothèque de services de médiation selon les transformations requises des données échangées afin d'appeler par la suite celui qui est nécessaire pour assurer la compatibilité des données transmises.

Mots clés : Médiateurs, Entités logicielles, Hétérogénéité, Composition

Abstract :

The number as well as the different types of applications that are available on a variety of platforms for mobile devices is undeniably explosive. This phenomenon is likely to encourage the user of a mobile device to wait for new applications according to his own needs based on those already existing where the same application obtained is available on almost any type of platform even when there are differences between the platforms as well as in the applications that run on these platforms - namely to customize the behavior of the desired application according to the different environments of execution and the needs of the users.

Because of the requirements of the users and also the context of the target mobile device, a developer is sometimes obliged to combine heterogeneous software entities (different form of implementation: components, service..etc). However, composing the contextual software entities that correspond to the identified needs inevitably leads to processing several types of composition in terms of collaborations - heterogeneous / homogeneous.

The objective of this work is to realize the adaptation mediators necessary to eliminate this heterogeneity and thus allow a consistent composition. Dealing with heterogeneous coordination requires the integration of exogenous mediators by providing the concrete structure of a new Component-of-services entity as well as the endogenous mediators by providing the concrete structure to the proposed mediation service. The latter requires building a library of mediation services according to the required transformations of the exchanged data in order to s ubsequently call that which is necessary to ensure the compatibility of the transmitted data.

Keywords: Mediators, Software Entities, Heterogeneity, Composition.

Listes des abréviations

SDK : Software Développement Kit

HTML : HyperText Markup Language

IT : Information Technologie

SOA : Architecture Orientée Service

Table des matières

Chapitre 1 : Généralité sur les applications mobiles

Chapitre 2 : Processus de composition des applications mobiles

Chapitre3 : L'Approche d'adaptation pour la composition hétérogène des applications mobiles.

Liste des figures

Liste des tableaux

Introduction générale

L'utilisation de plus en plus fréquente des technologies mobiles nous amène à faire face à de nouveaux défis afin de satisfaire les utilisateurs. Ces derniers souhaitent pouvoir utiliser leurs applications favorites sur n'importe quel appareil mobile. De plus, ils souhaitent que les applications puissent être automatiquement personnalisées en fonction de leur position, la luminosité ou toute autre information de contexte. Cependant, nous devons faire face aux principales caractéristiques de ces systèmes : la variation du contexte, la mobilité et la limitation des ressources des appareils. Pour cause des exigences des utilisateurs et aussi du contexte de l'appareil mobile cible, un développeur se trouve parfois obligé de combiner des entités logicielles hétérogènes (différentes forme d'implémentation : composants, service...ect…). Cependant, composer les entités logicielles contextuelles qui sont correspondantes aux besoins identifiés conduit forcément à traiter plusieurs types de composition en terme de collaborations - hétérogènes/homogènes. L'interaction de ces entités logicielle entre elles nécessite un service de médiation pour transformer le format des données. Si le code d'une telle activité existe, le développeur applique la notion de la réutilisation de l'existent sinon il peut le développer sous forme d'un service ou d'un composant.

L'objectif de notre travail c'est implémenté une application mobile Self-Scanning en se basant sur un modèle architecturale détaillé via la réutilisation de l'existent ou bien création d'une nouvelle entités, donc notre travail s'inscrit dans le cadre d'adaptation des entités à composer l'une avec l'autre pour que l'application mobile souhaitée fonctionne correctement en assurant trois point principales :

- ✓ Le déplacement des données entre la source et la cible.
- ✓ Le mappage des données entre différents formats.
- ✓ L'exécution séquentielle et conditionnelle de l'enchaînement des opérations.

Pour cela notre mémoire est organisé comme suit

- **Chapitre 01 :** Ce chapitre sera consacré pour les appareils et les applications mobiles,

En représentent ses Caractéristiques, statistique sur marché et aussi les systèmes exploitation mobile Par la suite, nous mettons l'accent sur la description du contexte commençant par la définition de ce concept, sa relation avec les applications mobiles et les appareils mobiles en présentant quelques approches de modélisation qui sont dédiées pour décrire le contexte

- **Chapitre02 :** Dans un premier temps, nous introduisons les deux types de représentation les plus adoptées pour décrire le noyau fonctionnel d'une telle application. Puis, nous décrivons les tâches nécessaires pour obtenir une application composite tout en présentant dans un premier lieu le processus de composition. A ce stade, nous abordons les différents types de composition. Ensuite, nous introduisons un cadre comparatif entre la composition qui porte sur les services et l'autre qui porte sur les composants tout en mettant l'accent sur les problèmes d'hétérogénéité posés lors de la composition les multi-paradigmes émergents. Et enfin nous présentons les travaux de recherches articulés autour de la problématique de composition d'applications mobiles. Cette représentation a pour objectif de mettre en relief leurs apports et leurs manques afin de préciser les axes d'étude de notre proposition.
- **Chapitre03 :** Dans ce chapitre nous allons particulièrement intéresser au nouveau paradigme proposé en détaillant les défis de l'hétérogénéité comme nous allons traiter explicitement les contributions élaborées dans ce manuscrit. Ce chapitre présentera en détail l'approche adaptation proposé pour l'application mobile souhaité fonctionne correctement.
- **Chapitre04 :** Ce dernier chapitre sera réserver sur l'implémentation de notre application mobile qui est basé sur l'approche d'adaptation et enfin nous clôturons ce mémoire par une conclusion générale résumant les points essentiels de notre travail et dégageons quelques perspectives envisagées pour notre application mobile.

Chapitre 1 : Généralité sur les applications mobiles

Chapitre 1 Généralités sur les applications mobiles

1.1Introduction

Depuis quelques années, le marché de la téléphonie mobile est en plein essor et une part de plus en plus importante de la population possède un téléphone ou une tablette ayant la capacité de calcul d'un ordinateur. De cela, l'accès à internet a connu un changement majeur, de sorte que les mobiles sont devenus les principaux points d'accès pour l'utilisation d'internet, et grâce à l'évolution de la technologie, les mobiles ne sont pas utilisés juste pour appeler ou jouer à des jeux, mais avec les Smartphones par exemple nous pouvons planifier notre journée complète, consulter nos mails, faire des conférences téléphoniques, se connecter en utilisant un réseau social et effectuer une foule d'autres activités ; tous cela en utilisant ce qu'on appelle ' les applications mobile '. Les applications mobiles ont un impact primordial sur notre quotidien, elles nous permettent par exemple de réduire la charge de travail, substituer le format papier en format électronique, diminuer les coûts téléphoniques et nous offre d'innombrables autres services. [1]

Ce chapitre sera divisé en trois grandes parties. La première constituera une revue des notions de base relatives aux périphériques mobiles, tandis que la seconde sera dédiée aux applications mobiles et la troisième partie est destinée à la notion de contexte.

1.2 Les appareils mobiles

1.2.1 Définition

Un appareil mobile, traduction littérale du terme anglophone "mobile device", est un ordinateur portatif utilisable de manière autonome lors d'un déplacement. L'appareil est de taille et de poids réduits pour permettre un usage mobile. Il est doté d'une batterie bénéficie souvent d'une connexion Bluetooth ou même wifi pour fonctionner de manière autonome. Il se présente typiquement comme un écran possédant une interface tactile ou un clavier miniaturisé. Il peut s'agir de tablettes tactiles, de Smartphones, de téléphones mobiles, d'assistants personnels (PDA), d'assistants de navigation personnel, etc. [1]

1.2.2 Les différents types des appareils mobiles :

Il est possible de regrouper les appareils mobiles en fonction de leur ergonomie et de leur usage. Nous avons choisis les plus utilisés qu'on détaille ci-dessous :

- **Les Smartphones**  ***:***

– Définition des Smartphones

Smartphone ou " téléphone intelligent " est un téléphone mobile qui possède des fonctions Proches à celles d'un assistant numérique personnel grâce à un système d'exploitation, possédant des fonctionnalités précises comme l'accès à Internet, la bureautique, la gestion de carnet d'adresses, le multimédia (MP3, vidéo, photos,. . .) , les jeux et sont mêmes équipés de GPS (Système Global de Positionnement).[2]

- **Les tablettes tactiles**  :

– Définition des tablettes

Tablette (de l'anglais tablet, plaque) est le nom donné à une famille d'ordinateur portables dépourvus de clavier à touches et munis d'un écran tactile, de la même dimension qu'une feuille A4 ou plus petits. L'écran tactile est toujours multipoints, donc capable de détecter plusieurs touchés simultanés.

-- Utilisation des tablettes

Ces ordinateurs sont essentiellement tournés vers l'utilisation d'Internet : consultation de pages Web, lecture de journaux en ligne ou de livres électroniques ou messageries. Leurs dimensions excluent l'intégration d'une mémoire de stockage mécanique, comme un lecteur-graveur de DVD ou un disque dur [2].

- **Les assistants personnels (pdas**)

Assistant personnel ou PDA (Personal Digital Assistant) ou ordinateur de pocheen français est un équipement électronique bureautique de poche utilisé principalement pour ses fonctions d'agenda, de répertoire téléphonique et de bloc-notes ,mais les avancées technologiques ont permis de lui adjoindre des fonctionnalités multimédia, telles que le dictaphone, le lecteur de mp3, d'images, de vidéos, auxquels s'ajoutent des programmes qui le transforment en outil de navigation associéà un GPS, par exemple. Il s'utilise avec un stylet [2].

1.2.3 Caractéristiques :

Aujourd'hui, les Smartphones et tablettes sont fabriqués par des constructeurs différents(ex: les plus populaires sont *Samsung*, *Apple* et *LG*) où chacun d'entre eux fournit des architectures matérielles différentes : des écrans différents en terme de la résolution et de la taille, des processeurs différents en terme de la vitesse, des caméras différentes en terme de la qualité des images obtenues, côté connectivité (GPS, Wifi), les espaces de stockages, l'énergie de la batterie, etc. .. [3].

- **Ecran :**

La taille d'un écran est exprimée en pouces (1 pouce = 2,54 cm), elle mesure la diagonale d'un écran. Les Smartphones ont généralement un écran entre 3,5 et 5,5 pouces. La taille de l'écran est un élément déterminant lors du choix d'un appareil puisqu'elle a un impact sur la taille du Smartphone. Un écran plus grand facilite la lecture et est plus confortable pour jouer ou regarder un film. Cependant, les plus grands modèles sont difficiles à manipuler à une main et surtout, ne peuvent pas toujours se glisser dans une poche. Autre critère important : la résolution, c'est-à-dire le nombre de points composant une image. Plus la résolution est élevée, meilleure est la qualité d'affichage d'un écran [3].

- **Autonomie :**

C'est une des parties essentielles d'aspects, elle représente le potentiel de la batterie d'un Smartphone en termes de la consommation énergétique. L'autonomie diffère en fonction des tâches que nous effectuons sur les Smartphones. L'exécution d'une telle application mobile qui consomme beaucoup d'énergie implique que la vie de la batterie ne va pas durer beaucoup de temps. Donc c'est une question de taille de la batterie qui est mesurée en mAh. En dépendance avec cette capacité, l'autonomie est exprimée en fonction du temps par heures/minutes. La capacité de la batterie et donc l'autonomie dans les différents usages possibles (tel que la communication, la navigation web, la lecture vidéo, etc.) se diffère d'un Smartphone à un autre [3].

- **Mémoire :**

Les smartphones ont une mémoire sur laquelle vous pouvez stocker vos applications, vos photos, votre musique… Elle peut varier fortement d'un modèle à l'autre : les modèles de base ont généralement 4 gigas alors que pour les modèles plus couteux, cela peut monter à 64 gigas. Si vous pensez stocker des photos, des vidéos et des fichiers musicaux, n'oubliez pas de vérifier si vous pouvez ajouter une carte mémoire (ex. : une carte microSD) pour augmenter la capacité de stockage [3].

- **Puissance :**

Comme les ordinateurs, les smartphones ont un processeur. Plus celui-ci est puissant, plus les applications s'exécutent rapidement. C'est un critère important pour ceux qui aiment jouer sur leur smartphone. Généralement, les smartphones bon marché ont un processeur moins puissant ou plus ancien [3].

1.2.4 Limites

Le développement d'applications pour périphériques mobiles se confronte à de différentes limites relatives au périphérique lui-même. On peut citer : la taille d'écran, la saisie des données et la stabilité du réseau [2].

- **La taille d'écran**

Elle est réduite et varie d'un appareil à un autre. Ce qui limite largement le nombre d'informations que l'on peut afficher [2].

- **La saisie des données**

La saisie des données dans un périphérique mobile semble être relativement pénible. Dans le cas des téléphones mobiles, il est impératif d'évaluer la difficulté rencontrée lors d'une simple opération de saisie (exemple : la saisie d'un SMS) sans citer même des cas plus complexes comme le remplissage d'un formulaire. Certains PDAs disposent de mini claviers ou de logiciels de reconnaissance de caractères, n'empêche que la saisie reste lente et fastidieuse [2].

- **L'instabilité du réseau**

Les périphériques étant dotés d'une connexion Wi-Fi, les utilisateurs sont confrontés de fréquentes déconnexions et risquent, par conséquent, de ne pas être toujours connectés [2].

1.3 L'application mobile :

1.3.1 Présentation des applications mobiles:

Une application mobile est un logiciel applicatif transportable et autonome, développé pour être installé sur un appareil électronique mobile. Elle est identifiée par un ou plusieurs programmes téléchargeable de façon gratuite ou payante depuis un magasin d'applications "Application Store ", permettant d'accéder à un contenu homogène et exécutable à partir du système d'exploitation du Smartphone. Les applications mobiles permettent en général un accès plus pratique, rapide et efficace à des sites en version mobile ou web.[4]

Pour télécharger une application sur un téléphone mobile, il existe différentes possibilités:

- transfert depuis un ordinateur via un câble de connexion,
- à partir d'un service mobile,
- via une boutique logicielle accessible depuis un téléphone mobile (App Store d'Apple, Windows Market Place, Nokia OVI, AndroidMarket, etc.)

- le cas échéant l'application est dite native ; elle est déjà dans le téléphone lors l'achat du téléphone (l'opérateur ou le fabricant l'a ajouté comme fonction de base) [5].

Figure 1.1- Un portable ANDROID et ses applications [5]

1.3.2 Histoire des applications mobiles

La naissance des applications mobiles remonte à la fin du vingtième siècle, Leurs industries ont commencé quand l'phone d'Apple et leur magasin iTune ont été introduits en 2007.C'était aussi le moment ou les utilisateurs ont commencé à utiliser leurs appareils mobiles pour surfer sur internet, Seules quelques personnes regardaient leurs courriers électroniques, mais avec l'phone le taux de consommateurs a augmenté et les téléphones mobiles ne sont plus utilisés que pour des appels téléphoniques, mais pour beaucoup plus que ça, puisque le nombre de nouvelles applications mobiles s'accroit de jour en jour et ces applications nous offrent plusieurs services tel la consultation du courrier électronique, la téléphonie-IP etc. [6]

1.3.3 Différents types d'applications mobiles

On peut classifier les applications mobiles selon quatre différents types ; applications natives, web applications, applications hybride et applications flash ; qu'on définit comme suit [7] :

1.3.3.1 Application native

Les applications natives sont des logiciels qui ont été développé spécifiquement pour une plate-forme mobile ou plus exactement pour des systèmes d'exploitation utilisé par les Smartphones et tablettes (iOS, Android, etc.), en utilisant un SDK propre à elle. Les applications ainsi crées sont ensuite téléchargeables depuis une plateforme dédiée au système, généralement un magasin d'application comme 'Androïde Market' de Android ou 'App store' de Apple.

Le fait de développer une application native permet généralement d'utiliser toutes les fonctionnalités liées au système d'exploitation visé (GPS, accéléromètre, appareil photo, etc.) et permet également de proposer des applications généralement plus riches que les web applications en HTML5. Une fois téléchargées et installées directement sur le mobile. Cette installation se faisant soit au travers d'un téléchargement via Internet soit par déploiement depuis un ordinateur connecté au mobile.

Les tests pour vérifier le comportement de ces applications nécessitent des compétences techniques spécifiques et des appareils très couteux [7].

1.3.3.2 Application Web

Une application mobile Web est une application développée en HTML, accessible et exécutable par le biais d'un navigateur Internet pour téléphone mobile. Elle utilise le navigateur du Smartphone et ne nécessite pas forcément de télécharger l'application. Les applications mobiles Web complètent les applications natives qui sont développées spécifiquement pour un système d'exploitation et qui doivent être téléchargées et installées par la mob nautes. Elles s'adressent donc à l'ensemble des utilisateurs de mobiles, et non à une population spécifique utilisant une marque bien précise. Toutefois, les applications Web doivent être testées pour chaque navigateur, résolution et taille d'écran, à l'instar de n'importe quel site Web [7].

1.3.3.3 Application hybride

L'application hybride est une application pour mobiles qui combine des éléments HTML5 sous forme d'application mobile Web et des éléments d'une application native permettant l'utilisation des fonctionnalités natives des Smartphones et d'être distribuée en tant qu'application sur les stores des systèmes mobiles (App Store, Play Store, etc.).

Plusieurs stratégies sont alors possibles, selon que l'on place le curseur plus du côté natif ou plus du côté HTML :

- ne réaliser que certains écrans voir même que certains composants d'IHM en HTML.
- réaliser toutes les écrans en HTML mais garder la logique applicative en code natif, notamment les effets de transitions entre écrans et la gestion du scrolling.
- réaliser les écrans en HTML, et les transitions / scrolling en JavaScript. Le code natif peut alors se cantonner à quelques composants techniques très ciblés. De la même manière selon les applications .Le logique métier peut être codé en JavaScript ou bien en code natif.

Quelques exemples d'applications hybrides :

- LinkedIn
- Microsoft Bing pour mobiles

1.3.3.4 Applications Flash

Comme pour les applications HTML5/JavaScript, une application Flash est une solution de développements multiplateformes, qui permet d'accéder à la plupart des ressources des Smartphones et tablettes : multitouch, accéléromètre, GPS, bases de données locales SQLite. Ces applications embarquent le runtime AIR dans leur code. On a par exemple Flash Builder / Flex en version 4.5 permettent de packager des applications Android, iOS, et BlackBerry Tablet OS [7].

1.3.4 Caractéristiques d'application mobile

Des contraintes techniques qu'il est nécessaire de prendre en compte lors de la conception d'une application mobile [7]:

- Tailles d'écrans variables, pouvant dans certains cas être assez réduite
- Possibilité limitée de saisie de données
- Puissance du processeur, pouvant être limité sur les premiers smartphones
- Tailles de la mémoire pouvant varier
- Autonomie du smartphone
- Débits variables de la bande passante Internet.

L'application mobile à réaliser, et c'est un point crucial à ne pas négliger
Une application mobile doit respecter certaines règles :

- Utiliser des images petites et légères
- Utiliser des éléments facilement accessibles
- Maitriser l'utilisation du javascript pour économiser la batterie
- Adapter le mode de saisie des informations.

Les Solutions mobiles adaptées aux exigences et contraintes des clients Mobile reposent sur cinq domaines de compétences :

- **Techniques** iOS, Android, Windows, Objective-C, Java, C#, XAML,
- **Architecture :** Performance, fiabilité, Intégration, sécurité, Evolutivité,..
- **Design & Ergonomie**
- **Fonctionnelles** spécifiques à la mobilité
- **Démarche Projet** et aux conseils relatifs aux projets de mobilité.

1.3.5 Domaines d'application mobile

Avec les possibilités matérielles incorporées aux terminaux (caméra, GPS, gyroscope, …), les applications Smartphones et Tablettes peuvent intégrer des fonctionnalités spécifiques et dédiées pour les utilisateurs, permettant ainsi d'enrichir le spectre fonctionnel et imaginer des usages non couvert jusqu'à présent par les systèmes d'information

- Géolocalisation, Itinéraires
- Scan de Code barre, Flash, QR Code
- Réalité augmentée
- M-commerce, Paiement mobile
- Push et notification
- Gestion de documents, dématérialisation, Workflow
- Analyse d'Audience
- Gestion et Sécurisation de parc et de déploiement de terminaux mobiles [7].

Figure 1.2 : domaines d'application mobile [7].

1.3.6 Eléments de comparaison entre les applications mobile [7]

- **Coûts de mise en œuvre**

Développer une application native pour plusieurs plateformes mobiles peut coûter très cher, de par la multitude de langages et technologies mises en œuvre.
Selon le nombre de plateformes cibles, une technologie web ou même hybride sera souvent moins coûteuse. De plus il sera souvent plus simple de disposer de développeurs maîtrisant les technologies web, que les diverses plateformes mobiles.

- **Qualité, rapidité des applications**

Difficile de rivaliser avec les applications natives, Celles-ci seront presque toujours plus rapides.
Mais cela dépend fortement du type d'application en jeu et de ses fonctionnalités.

- **Publication et mises à jour**

Une importante contrainte des applications natives est que celles-ci doivent être approuvées avant diffusion sur leur store respectif (sauf pour Android), ce qui peut s'avérer long et contraignant. Une application hybride permet de limiter ce désagrément, et un web App de s'en affranchir complètement.
Le même problème se pose pour les mises à jour, il n'est souvent pas possible de diffuser un patch correctif en urgence ou même rapidement sur une application native.

- **Monétisation**

Les magasins d'applications permettent très facilement de vendre les applications, mêmes si Apple, Google et consorts prélèvent leur part sur les prix de vente, généralement autour de 30%.
Même si les stores d'applications web comment timidement à apparaître, leur usage est encore très restreint. Les stores servent aussi de moteur de recherche et de vitrines pour les applications, et permettent ainsi de les mettre en avant et de les faire découvrir.

1.3.7 Les meilleures applications mobiles en 2019

Le nombre de téléchargements d'applications mobiles est actuellement en forte hausse. Cette tendance va de pair avec la vente des Smartphones [8].

➢ **Dans la catégorie réseaux sociaux et messagerie**

Catégorie très prisée des utilisateurs, les réseaux sociaux voient leur nombre de téléchargement explosé cette année. Voici donc le peloton de tête des applications les plus téléchargées !

- **Whats App** 

WhatsApp est plus un service de messagerie qu'un réseau social. Il vous permet de créer des groupes de discussion de plusieurs contacts dans lesquels tous les messages instantanés, les photos et les vidéos sont partagés simultanément à tous les participants.

- **Facebook Messenger** 

Facebook Messenger est une application de messagerie instantanée dédiée qui vous permet de communiquer avec vos amis Facebook sans utiliser votre forfait téléphonique.

- **Tiktok** 

TikTok est un outil de montage et de publication de vidéo clips musicaux au sein de sa communauté. Le principe : fait ta vidéo, ajoute une musique, des émoticônes, des animations et poste-là pour que toute la communauté de TikTok la voit.

- **Instagram**

TikTok est à la vidéo, ce qu'Instagram est à la photo.

➢ **Dans la catégorie vidéo**

- **Youtube**

Youtube qui remporte la palme de l'application la plus téléchargée dans la catégorie vidéo.

- **Netflix** 

L'application Netflix pour Android et iOS vous donne accès à un immense catalogue de séries et de films.

➢ **Dans la catégorie musique**

- **Sportif**

SPOTIFY PREND CEPENDANT UNE PETITE LONGUEUR D'AVANCE, CERTAINEMENT GRACE A SA BIBLIOTHEQUE MUSICALE PLUS ETENDUE ET AUX RECOMMANDATIONS PERSONNALISEES.

➢ **Dans la catégorie GPS**

- ***Google maps*** 

Google Maps vous guide et vous indique les boutiques et services autour de vous, si vous en avez besoin.

- **Waze** 

Waze occupe la seconde place des meilleures applications GPS, avec sa communauté très active qui vous alerte en temps réel, des accidents susceptibles de vous ralentir et d'augmenter votre temps de trajet.

1.3.8 Systèmes d'exploitation mobiles

1.3.8.1 Définition

Tout comme un ordinateur dispose d'un système d'exploitation, les téléphones mobiles se composent également d'une plateforme qui contrôle toutes leurs fonctionnalités. Ceci est connu comme un système d'exploitation mobile. Généralement connu sous le nom d'OS (Operating System) mobile, il s'agit d'un système d'exploitation qu'exploite un appareil mobile tel qu'un Smartphone, une tablette tactile, etc. Il contrôle et coordonne toutes les opérations de base du téléphone mobile comme les options d'écran tactile, Bluetooth, Wi_, appareil photo, etc. et assure la liaison entre les ressources matérielles, l'utilisateur et les applications (traitement de texte, jeux vidéo, etc.).
On trouve plusieurs systèmes d'exploitation sur les mobiles, alors nous choisissons plus intéressant : iOS, Android, A chaque OS mobile correspondue technologie et un « store » où les applications peuvent être téléchargées, de manière gratuite ou payante : iOS (Apple), Android (Google Play)[9].

1.3.8.2 IOS d'Apple

- **Ios**(anciennement **iPhone OS**) est le système d'exploitation mobile développé par Apple pour l'iPhone. Il est dérivé de Mac OS X dont il partage les fondations (le kernel hybride XNU basé sur le micro-noyau Mach, les services Unix et Cocoa, etc.). iOS comporte quatre couches d'abstraction, une couche « Core OS », une couche « Core Services », une couche « Media » et une couche« Cocoa »Le système d'exploitation occupe moins d'un demi-gigaoctet (Go) de la capacité mémoire totale de l'appareil [7].

- **L'App Store**

Est la plateforme de téléchargement des applications mobiles iPhone (similaire au Google Play) Selon les cas, les applications obtenues sur l'App Store sont gratuites ou payantes [7].

➢ Caractéristique

Logicielle de l'phone est caractérisée par :

Le Base Band : C'est donc un micro logiciel autonome qui s'occupe en temps réel de toutes les interactions avec les périphériques de communication de l'appareil : Bluetooth, Wi-Fi et GSM.

Le Boot Loader : C'est une partie du Base Band, dont le rôle principal est d'assurer le démarrage de l'iPhone, de contrôler son activation, et sa compatibilité avec la carte SIM insérée

Le firmware : il s'agit d'un logiciel interne de l'appareil, cette fois responsable de la gestion de sa partie systémique (l'écran, le clavier tactile, etc.).

Le SeckPack : C'est une partie de la mémoire flash de l'appareil contenant entre autres des informations sur le verrouillage de celui-ci.

➢ Versions

- **La version 2.0**d'iOS a été dévoilée à l'occasion d'une conférence le 6 mars 2008,
- **La version 2.2.1** a été la dernière avant la sortie, le 17 juin 2009, soit un an après laversion 2.0, de la mise à jour 3.0 qui a apporté son lot de fonctionnalités tel que le copier-coller,
- **La version 4.0** du firmware, présentée le 8 avril 2010,
- **La version 5.0** du firmware a été annoncée le 6 juin 2011 qui apporte la présentation du système **Icloud et la 6.0** le 11 juin

1.3.8.3 Androïde de Google

➢ Androïde

Android est un système d'exploitation open source utilisant le noyau Linux, pour terminaux mobiles conçu par Android, une startup rachetée par Google, et annoncé officiellement le 5 novembre 2007. D'autres types d'appareils possédant ce système d'exploitation existent, par exemple des téléviseurs, des radio-réveils, des autoradios et même des voitures. [10]

- **Google Play**

Anciennement dénommé AndroidMarket, est le magasin en ligne de Google. Celui-ci permet de télécharger des logiciels, des livres, des films ou de la musique, payants ou non. Il est aussi possible de les noter et de les commenter. En septembre 2011, il y avait plus de520 000 applications sur AndroidMarket, dont 65 % sont gratuites.

AndroidMarket a été remplacé par Google Play Store le 6 mars 2012.

- **Caractéristiques**

- Android est un système d'exploitation fondé sur un noyau Linux.
- il comporte une interface spécifique, développée en Java, toutefois il est possible depasser outre cette interface, en programmant ses applications en C, mais le travail deportabilité en sera plus important.
- la majorité des périphériques Android sont basés sur l'architecture ARM.
- Android a été conçu pour intégrer au mieux des applications existantes de Googlecomme le service de courrier Gmail, celui de cartographie, Google Maps, ou encore Google Agenda, Google Talk, YouTube. Un accent particulier est mis sur la géolocalisation avec Google Latitude et la météo correspondant à la ville la plus proche disponible sur le menu principal.

- **Version**

Android est régulièrement mis à jour

La première version d'Android a été distribuée fin 2007. Depuis, de nombreuses versions ont vu le jour. La version la plus récente est la 4.2.1 Jelly Bean.

Version	Codename	API	Distribution
2.3.3 - 2.3.7	Gingerbread	10	0.3%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	0.3%
4.1.x	Jelly Bean	16	1.1%
4.2.x		17	1.6%
4.3		18	0.5%
4.4	KitKat	19	7.8%
5.0	Lollipop	21	3.6%
5.1		22	14.7%
6.0	Marshmallow	23	21.6%
7.0	Nougat	24	19.0%
7.1		25	10.3%
8.0	Oreo	26	13.4%
8.1		27	5.8%

Figure 1.3 : version d'Android [10].

1.4 Contexte et sensibilité au contexte

1.4.1 Contexte

La notion du contexte en informatique est ancienne et remonte aux années soixante où elle a été déjà exploitée par différents domaines, tels que les systèmes d'exploitation, la théorie des langages et l'intelligence artificielle. Dans les systèmes d'exploitation par exemple, un contexte caractérise l'ensemble minimal d'informations sur une tâche en exécution (processus) et permet de retourner à l'exécution de cette tâche après l'occurrence d'une interruption et l'exécution du programme de traitement de cette dernière [11,12]. Différents chercheurs se sont intéressés à éclaircir la notion de contexte pour l'informatique ubiquitaire. En effet, il est difficile de trouver d'une manière claire et unique, une définition valable dans tous les travaux impliquant cette notion. On cite ces définitions selon leurs ordres chronologiques comme suit [11,12] :

- En 1994, plusieurs approches au mot contexte ont été introduites par divers chercheurs :

- Schilit et Theimer, introduisent l'expression context aware pour désigner la localisation et l'identité des personnes et des objets à proximité ainsi que les modifications pouvant intervenir sur ces objets [13].
- Schilit et all [14], ont donné une définition plus vaste ; selon eux le contexte englobe plus que la localisation de l'utilisateur puisque d'autres centres d'intérêt peuvent aussi changer du contexte. Tels que les effets de la lumière, le niveau du bruit, la connectivité réseau, le coût de la communication et la bande passante de la communication.

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

- Brown, Bovey et Chen [15] présentent des éléments de base flexibles : la localisation, l'ensemble des objets dont l'utilisateur a besoin, le temps et l'orientation partiale (direction) afin de caractériser le contexte.
- Ryan, Pascoe et Morse [16], annoncent que : "les éléments du contexte sont : la localisation de l'utilisateur, l'environnement, l'identité et le temps".
  - En 2000, Chen et Kotz [17], déterminent le contexte comme un : "ensemble des états environnementaux et paramètres qui déterminent le comportement d'une application ou dans lequel un événement de l'application se déroule et ayant un intérêt pour l'utilisateur".
  - En 2001, Dey [18], affirme qu'un contexte désigne toute information utile pour caractériser la situation d'une entité qui peut être un objet ou une personne.

1.4.2 La sensibilité au contexte

La notion de sensibilité au contexte "context-awareness" concerne l'utilisation du contexte dans les applications. Elle caractérise la capacité d'un système à s'adapter aux changements du contexte [11,19].

Plusieurs définitions ont été présentées par plusieurs chercheurs :

- Selon Schilit, Adams et Want [20], premiers chercheurs à avoir évoqué le terme sensibilité au contexte en 1994 dans leurs travaux sur un système de localisation, la sensibilité au contexte est définie comme l'aptitude d'une application à

s'adapter au contexte de son exécution selon : la localisation, l'ensemble des personnes à proximité, les machines ,les équipements accessibles, de même que les changements de ces objets dans le temps.

- En1995, Brown[21]a défini la sensibilité au contexte dans son travail relatif à un guide touristique comme toute application qui prend en compte le contexte de l'utilisateur.
- En couplant et en élargissant les définitions de Brow et Dey(2001) [22], on tire aussi la définition suivante : "les applications sensibles au contexte sont des applications dont la structure et le comportement varient en fonction du contexte. Elles utilisent les observations de contexte pour fournir des informations et des services pertinents pour l'utilisateur".

1.4.3 Vers l'adaptation

Une définition du context-aware plus orientée vers l'adaptation au contexte est donnée par Brown [23]. Il dit qu'une application sensible au contexte doit automatiquement extraire de l'information ou effectuer des actions en fonction du contexte utilisateur détecté par les capteurs.

D'un côté, une application peut juste se concentrer sur la détection, la perception et l'interprétation des informations contextuelles de l'environnement d'exécution. Ce qui démontre que l'utilisation simple du contexte n'implique pas une modification de son comportement. Par exemple, les applications dans les travaux [24,25] font juste des connaissances sur les éléments de l'environnement de l'utilisateur.

D'un autre côté, les applications peuvent changer dynamiquement leur comportement en fonction du contexte de l'environnement d'exécution [26,27].En conséquence, nous avons abouti à une définition beaucoup plus orientée vers notre axe de recherche pour les deux notions d'adaptation et d'auto-adaptation. Afin de pouvoir obtenir des applications adaptatives, le développeur acquiert des connaissances sur l'environnement d'exécution pour garantir si les contraintes d'exécution d'une application mobile sont conformes au contexte d'exécution de l'appareil mobile cible, i.e. l'adaptation du contenu en fonction du terminal (cf. Figure 1.4). Par conséquent, l'adaptation est le fait de s'assurer que l'application souhaitée pourra fonctionner correctement dans un tel environnement mobile[28].

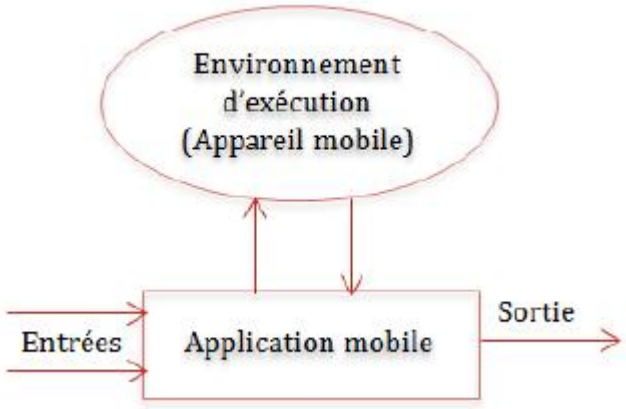


Figure 1.4 Application mobile Adaptable [28].

Dans le cas du changement des variables du contexte d'exécution, l'application change son comportement de telle sorte où les contraintes d'exécution associées au nouveau comportement doivent être à leur tour conforme à l'environnement d'exécution. Dans ce cas, l'application acquiert des connaissances sur le contexte d'exécution courant, i.e. prévoir les changements du contexte. Ici nous parlons des applications auto-adaptables (cf. Figure 1.5). La boucle de contexte exprime la modification des valeurs des variables du contexte où l'application va subir à un changement de comportement en fonction de ce nouveau contexte[28].

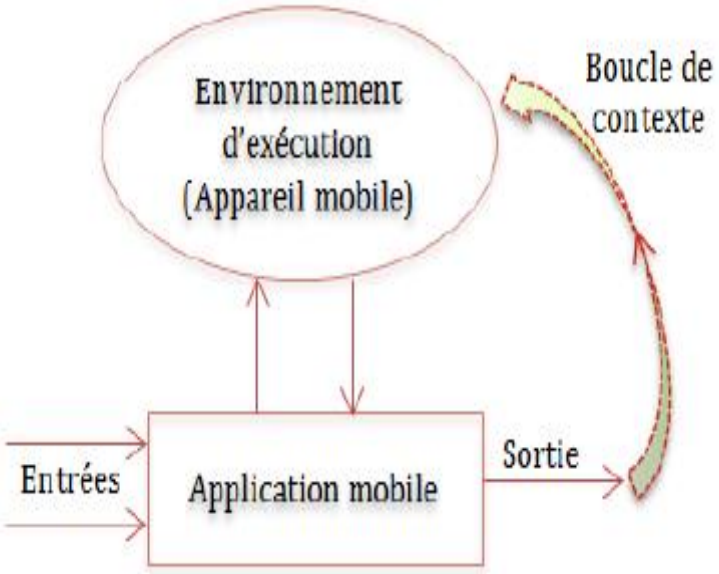


Figure 1.5 application mobile auto-adaptable [28].

1.5 Conclusion

Dans ce chapitre nous avons passé en revue quelques informations théoriques sur les notions clés liés à notre travail incluant des définitions présentées dans la littérature pour chacune des appareils mobiles, des applications mobiles et de contexte.

Nous avons abordé au premier temps une définition sur les appareils mobiles, Ainsi, nous avons introduit ses caractéristiques, ses limites.

On a présenté aussi quelques définitions dédiées aux applications mobiles. Ainsi, nous avons présenté ses caractéristiques, ses types, ses domaines ses plateformes de téléchargements existantes en arrivant par la suite à une étude statistique décrivant les chiffres de téléchargements des applications en générale.

Nous avons essayé de comprendre ce qui est le contexte pour les applications mobiles et son intérêt pour garantir un meilleur fonctionnement. Dans le chapitre qui suit, nous aborderons toutes les notions et les différentes techniques ainsi que les travaux existants qui sont liés à notre travail qui reflètent un socle théorique pour le processus de composition proposé [29] utilisé dans notre travail.

Chapitre 2 : Processus de composition des applications mobiles

Chapitre 2 Processus de composition des applications mobiles

2.1Introduction

Le monde connaît une avancée considérable dans l'utilisation des appareils téléphoniques portables (mobiles) grâce aux applications mobiles, ces dernières sont capables de satisfaire les besoins actuels des utilisateurs avec de nombreuses fonctionnalités et en orant plusieurs services. Mis à disposition de l'utilisateur, celui-ci peut avoir besoin d'utiliser plusieurs applications à la fois selon son contexte d'utilisation afin d'avoir leurs propres applications qui répondent à ces exigences. Face à ce constat le développeur se trouve parfois obliger de combiner différentes applications ou des entités logicielles déjà existantes afin de pouvoir tirer le meilleur parti de leurs fonctionnalités et d'avoir finalement une nouvelle application comblant les besoins des utilisateurs. Par conséquent, les applications mobiles sont presque devenues infinies ce qui explique le besoin de composer de nouvelles applications par la réutilisation de l'existant pour combler s'il y a des exigences sans avoir à reprogrammer.

Une application peut être construite par composition d'entités logicielles existantes où ces entités peuvent être soit des composants, des services ou des applications elles-mêmes. Les auteurs dans [30] ont déclaré que cette réutilisation nécessite que chaque service utilisé soit au préalable décrit par son fournisseur. L'avantage de la réutilisation est un gain de temps et d'investissement lors de la conception d'une nouvelle application. Par conséquent, un des leviers permettant d'augmenter la rentabilité ainsi que la rapidité de développement est de favoriser la réutilisation de l'existant. Cette réutilisation peut se faire grâce aux mécanismes de composition d'applications.

2.2 Noyau fonctionnel

Nous nous abordons dans ce travail sur les représentations de deux approches importantes pour faciliter la tâche de développement et d'un autre côté pour favoriser la réutilisation de l'existant. Deux types de représentations prédominent alors : l'approche utilisant des services (SOSE : Service-Oriented Software Engineering) et l'approche basée sur des composants (CBSE : Component-Based Software Engineering).

2.2.1 L'approche à base de composant

L'approche à base de composants est largement utilisée pour construire des systèmes complexes en s'appuyant sur les caractéristiques de modularité et d'autonomie pour le choix des éléments et la caractéristique de la compossibilité pour relier et attacher ces élément dans un bloc cohérent. [31].Ces approches reposent sur des langages de description d'architecture qui aident à la conception et la définition de la configuration, ces langages viennent compléter la notion de composant en permettant la description d'architectures.

De nombreux langages de ce type sont développés au sein des universités et des instituts de recherche. On peut citer les projets : Wright [32], ACME [33] de l'Université de Carnegie-Mellon,xADL[34] de l'Université de Californie ,Rapide de l'Université de Sanford[35]. Ils fournissent différentes définitions d'un noyau de concepts maintenant largement adopté : en tout premier lieu, la notion même de composant, mais aussi les notions de connecteurs, de ports, de rôles, etc. qui expriment les services fournis et requis par les composants et les interactions entre eux à travers les interfaces. Ces notions se présentent sous forme graphique, textuelle, ou les deux à la fois. Tous ces langages cherchent à améliorer la réutilisabilité des composants et des connecteurs en séparant le calcul et la coordination. [36]

Une première définition des composants est donnée par Szyperski [37]**.** Celle-ci se focalise sur la composition, les interfaces contractuelles et le déploiement des composants. On se place dans un contexte d'assemblage ou ces entités (composants) ont déjà été développées et où l'on voudrait les assembler pour former une application. Les composants spécifient des interfaces sous forme de contrats, et peuvent être déployés séparément.

Définition est celle de la spécification UML 2.0 [38].Un **composant** est une partie modulaire d'un système qui encapsule son Contenu (boites noirs) et dont la manifestation est remplaçable dans son environnement. Un composant définit son comportement en termes d'interfaces fournies et requises. En tant que tel, un composant sert comme un type, dont la conformité est définie par ses interfaces fournies et requises (incluant à la fois leur sémantique statique et dynamique) .Celle-ci met l'accent sur la modularité, la substitution et le typage. Ici, on se place dans un contexte de modélisation architecturale ou l'on perçoit le système comme étant un ensemble d'unités modulaires et substituables. Chaque unité (composant) est typée par ses interfaces, qui spécifient ces services requis et fournis. Plusieurs modèles à composants, comme par exemple Fractal [39] et CCM [40], utilisent un langage de description d'architecture(ADL) et un langage de description d'interfaces (IDL). L'ADL permet de décrire explicitement la structure de l'application en termes de composants. L'IDL permet de définir les fonctions fournies pour chaque interface, et donc les services que les composants offrent les uns aux autres. Parmi les interfaces fournies par les composants, les modèles définissent souvent des interfaces d'introspection et de contrôle. Les interfaces d'introspection révèlent des informations internes au composant comme sa structure, les fonctionnalités offertes, son état, etc. Les interfaces de contrôle varient d'un modèle à un autre et peuvent concerner la gestion du cycle de vie ou la configuration (valeurs de paramètres, choix de services systèmes utilisés, etc.) des composants. Afin de construire une application à base de composants, les composants sont interconnectés à l'aide de leurs interfaces Certains composants sont donc assemblés par composition de composants [57] [41](Voir figure 2.1).

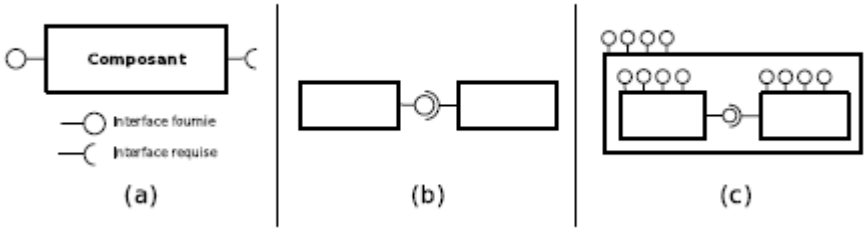


Figure 2.1 Interfaces, connexions et composition des composants [41].

2.2.2 L’approche à base d’un service

Une deuxième approche pour la construction d’applications et la représentation de son noyau fonctionnel est basé sur l’Architecture Orientée Services [42]connue sous l’acronyme SOA (Services OrientedArchitecture), apporte beaucoup de nouveautés au monde des systèmes d'information et à l'informatique en général. Les entreprises opèrent de plus en plus leurs applications métiers par le biais de cette architecture. Il s’agit d’un paradigme fondé sur la description de services et sur la description de leurs interactions.

L'objectif d'une architecture orientée services est de décomposer une fonctionnalité en un ensemble de fonctions basiques (services), et de décrire finement le schéma d'interaction entre ces services. Le résultat de cette communication peut consister en un simple retour de données ou en une activité (coordination de plusieurs services)[43].

L’interaction dans une architecture orientée services, s’articule autour de trois grands rôles producteur de service (fournisseur de service), un consommateur de service (client) et le répertoire de services (registre de stockage des services ou annuaire).
Le producteur a pour fonction de déployer un service sur un serveur et de générer une description de ce service. Cette dernière précise à la fois les opérations disponibles et leur mode d’invocation. Cette description est publiée dans un répertoire de services. Les consommateurs peuvent découvrir les services disponibles et obtenir leur description en lançant une recherche sur un répertoire.
Ils peuvent ensuite utiliser la description du service ainsi obtenue pour établir une connexion avec le fournisseur et invoquer les opérations du service souhaité (Figure2.2)[43].

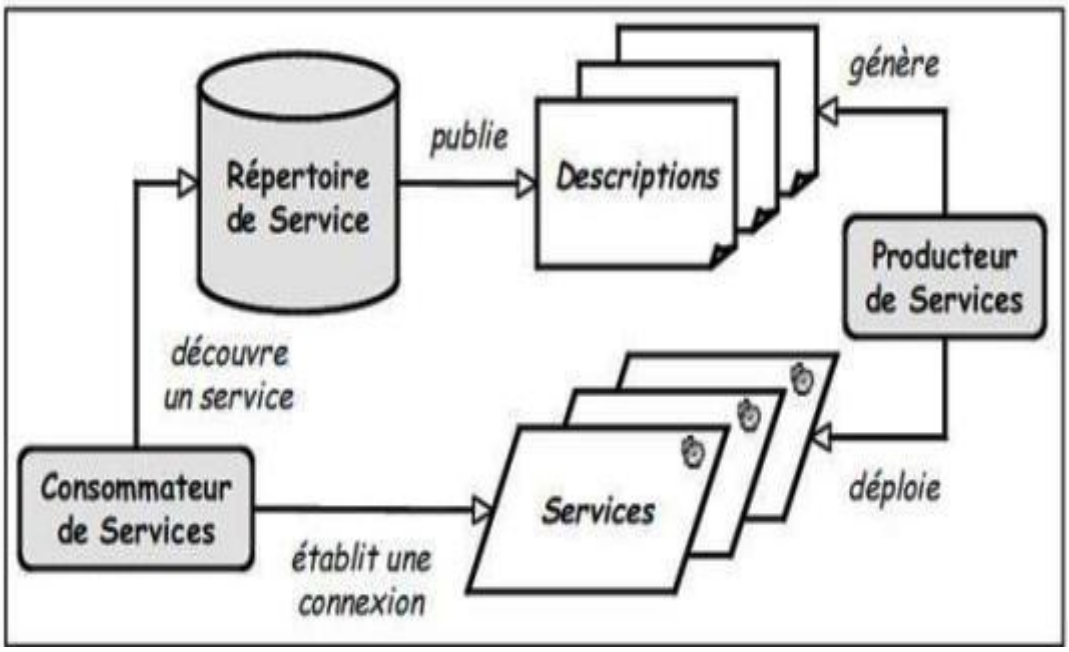


Figure 2.2 : Les interactions dans une architecture orientée services [43].

Aujourd'hui le SOA peut nous aider à mieux réutiliser nos investissements existants d'IT ainsi que les nouveaux services, c'est pourquoi le fait que les industriels développent SOA n'est pas simplement une mode. En effet, SOA est une solution métier qui est basée sur une entité technique nommée Service, elle se place plutôt dans la continuité logique des multiples tentatives de distribution des traitements, de répartition des données, d'intégration des applications, d'homogénéisation du système d'information etc. SOA permet d'intégrer les investissements IT plus facilement en utilisant des interfaces bien définies entre les services ,c'est-à-dire le fournisseur permet l'accès à son service à travers une interface, le client désigne une personne, un serveur ou une autre application qui accède au service et l'invoque à travers l'interface proposée [42].

Selon *Thomas Erl* [42], un service correspond à un ensemble d'opérations qui sont groupée logiquement et qui sont capables d'effectuer des unités reliées de travail, Une exigence de base pour la réalisation des objectifs stratégiques de SOA, c'est que les services doivent être intrinsèquement composables. Comme un moyen de réaliser ces objectifs, le paradigme de conception de SOA est donc naturellement porté sur la capacité de la composition flexible [43].

L’adoption de la SOA a été grandement facilitée par l’émergence opportune de la technologie des services web et leurs standards bien définis. Il y a huit principes à respecter et à suivre dans toute conception et déploiement d’une architecture SOA [42]Réutilisation, Couplage-lâche, *Stateless*, Autonomie, Interopérabilité, Abstraction, Description dans un contrat, Découverte et Composition.

Une déclinaison sur internet des services sont les web Services. Plusieurs langages ont été abordés pour décrire ce type d’entité logicielle. WSDL (Web Services Description Langage) [54] est un langage basé sur XML dédié pour la description des services web qui respectent la spécification WS*. En plus de l’utilisation du WSDL par les services web comme langage XML de description, ces services sont aussi basés sur SOAP (Simple Object Access Protocol) [45] comme un protocole de communication utilisant lui-même HTTP comme protocole de transport.

2.3 Processus de composition

Nous abordons au premier temps une définition générale de la composition d’applications de la part Christian Brel dans son travail de recherche [47] : « La composition d’applications est le moyen de combiner plusieurs morceaux d’applications existantes pour en construire une nouvelle. Cette composition réutilise tout ou une partie des applications existantes. ». Dû au fait qu’une application peut être représentée par des services ou des composants, plusieurs définitions ont été proposées pour la notion de composition.

Suite à l’étude de différents travaux sur la composition de services Web (Alonso et al. ,2004 [48], Benatallah et al ., 2005[49], Srivastava et al., 2005[50], Yang et al., 2004[51] nous retenons la définition de [49] qui nous paraît la plus générale. Dans [49], les auteurs considèrent la composition de services Web comme étant un moyen efficace pour créer, exécuter, et maintenir des services qui dépendent d’autres services.

Un service possède l'avantage d'être composable avec d'autres services. D'ailleurs, cette caractéristique constitue l'une des forces de ce concept. La composition de services a été définie dans [52]comme étant "*la capacité d'offrir des services à valeur ajoutée en combinant des services existants offerts par différentes organisations*". Le service ainsi résultant du processus de composition de services est appelé *service composite*. La composition de services peut être accomplie en combinant soit des services élémentaires soit des services composites.

De cette manière, les services composites sont définis d'une manière récursive comme une agrégation de services élémentaires et de services composites [53].

En effet, la composition d'applications est un processus complexe qui fait intervenir un ensemble d'étapes intermédiaires. Ces étapes se répartissent suivant les préoccupations classiques liées à la réutilisation d'entités logicielles existantes commençant par l'identification des entités qui seront composées puis la réalisation effective de cette composition. Par conséquent, le processus de composition d'applications peut être divisé en trois étapes : la découverte, la sélection et la composition d'entités logicielles. Vis à nos objectifs dans ce manuscrit, nous proposons un processus de composition global constitué de cinq étapes comme illustré dans la Figure 2.3[29].

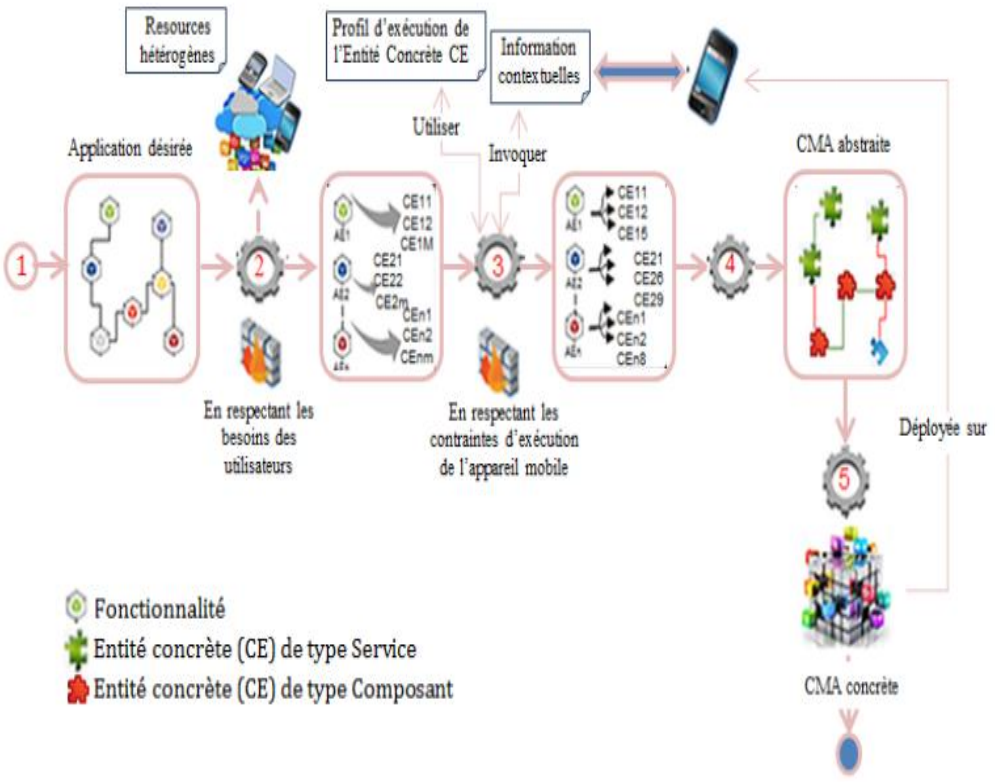


Figure 2.3 :Aperçu générale du processus de composition global proposé [29].

Ce processus vise à effectuer la tâche de composition en commençant par l'identification de l'ensemble des fonctionnalités souhaitées tout en ignorant comment elles seront implémentées ainsi que les différentes dépendances entre ces fonctionnalités. Ces dépendances décrivent les ordres d'invocation ainsi que les données échangées entre elles. Après, il associe chaque fonctionnalité identifiée aux entités logicielles concrètes correspondantes qui peuvent être utilisées pour l'implémenter et qui peuvent se trouver dans différents emplacements (ex. internet, emplacement locaux, d'autres appareils, etc.). L'étape suivante permet de sélectionner parmi les entités correspondantes trouvées les plus appropriées qui sont adaptables au contexte courant de l'appareil mobile. La tâche de sélection est basée sur les conditions d'exécution des entités logicielles et la description contextuelle de l'appareil mobile cible. Ensuite, il vise à composer les entités contextuelles sélectionnées tout en incluant les adaptateurs nécessaires dans le cas d'une coordination hétérogène. La dernière étape consiste à générer l'application mobile composite concrète vers une plateforme spécifique sur laquelle elle sera exécutée [29].

Nous allons maintenant aborder une définition spécifique pour chacune de ces tâches : la découverte, la sélection et la composition ainsi que quelques travaux apportés pour chacune d'entre elles.

2.3.1 La tâche de découverte

La découverte est un point clé nécessaire pour le développement d'applications par la réutilisation de l'existant. Elle désigne le processus qui permet d'identifier les entités logicielles qui peuvent répondre aux besoins exprimés par l'architecte. Ce processus de découverte repose, selon le cas, soit sur une recherche dans un *entrepôt* ou un *registre local* ou *public*, i.e. chercher des composants ou des services, soit sur une découverte globale sur le *web*, i.e. chercher des services web. La tâche de découverte sert à comparer les descriptions d'entités logicielles dont il a accès avec les exigences fonctionnelles et non fonctionnelles qui lui sont fournies afin de déterminer les entités candidates. Ces dernières sont les entités existantes d'après leurs descriptions peuvent répondre aux exigences[54]. Afin d'avoir cette liste des candidates, le processus de découverte doit passer par deux étapes essentielles :

- L'accès aux entités logicielles disponibles[55,56].

- La description de ces entités et les besoins puis la détermination d'une solution comme adéquate [57,58].

Cependant, nous abordons une définition plus convenable aux applications mobiles pour la notion de découverte : « Le processus de découverte vise à chercher et choisir les entités logicielles correspondantes pour chaque fonctionnalité désirée en les téléchargeant via internet où certaines d'entre elles sont gratuites tandis que d'autres sont payantes. Ce processus exploite l'ensemble des fonctionnalités identifiées qui représente l'application mobile désirée afin de trouver les différentes entités logicielles qui satisfont les exigences des utilisateurs décrites par ces fonctionnalités. » [29].

Dans le contexte d'une application qui a besoin d'exécuter une fonctionnalité implémentée comme un service Web par plusieurs fournisseurs, la découverte fait référence au processus de recherche des services Web implémentant la fonctionnalité souhaitée. Les registres UDDI sont les entités qui servent d'appui à la découverte de services web pour les applications client. De cette façon une application interroge un registre UDDI pour les fournisseurs d'un service Web. Parmiles solutions proposées pour cette problématique : OWL-S (Ontology Web Langage- Service), Web Services Dynamics Discovery et Web Service Discovery Architecture[59].

2.3.2 La tâche de sélection

Pour une telle composition, le résultat de la tâche de découverte est un ensemble d'entités logicielles correspondantes pour chaque besoin défini ou fonctionnalité désirée. Les entités trouvées mappées à la même fonctionnalité sont équivalentes fonctionnellement mais elles peuvent varier en plusieurs aspects non fonctionnels. La seconde étape de la composition correspond au processus de sélection où ce dernier sert à identifier parmi les entités candidate scelles les plus appropriées aux besoins de l'application à construire. En général, ces entités candidates répondent tous aux conditions minimales d'acceptation posées par l'architecte de l'application. La sélection se base donc sur les préférences des utilisateurs ou d'autres critères/contraintes pour identifier celle avec la plus haute qualité et/ou la plus adaptée au contexte d'utilisation. Ces contraintes de sélection diffèrent selon le domaine d'application [29].

Plusieurs travaux ont été effectués pour cette tâche. Maintenant, nous présentons parmi eux quelques travaux proposés précisément pour les environnements mobiles :

Les auteurs [60] discutent la sélection de service pour le service composite .Ils ont effectué la tâche de sélection selon les informations dépendantes du service et les informations dépendantes du dispositif (ex. niveau de la batterie, la force du signal du réseau, etc.). Considérant que, dans [61] une infrastructure de logiciel appelé *AppSpotter* qui permet la composition dynamique et automatisée de composants logiciels d'une application mobile est proposée. Sur cette infrastructure un composant nommé *component selector* sert à extraire à partir des *SPL assets* (software component stored) les composants logiciels qui répondent à un ensemble de critères où ces derniers représentent les caractéristiques de l'appareil mobile : la plateforme, l'écran, le clavier, etc. La tâche de sélection présenté en ces deux travaux [61,62]est entraîné par les caractéristiques de l'appareil mobile mais les critères de sélection ne sont pas les mêmes.

Cependant, nous avons abouti à une définition plus adéquate pour le processus de sélection : « Afin d'assurer le déploiement correct et le bon fonctionnement d'une application mobile composite, nous avons besoin dans un premier temps de s'assurer que ses entités logicielles constituantes sont adaptables au contexte courant de l'appareil mobile où l'application composite va être déployée. La phase de sélection correspond à une opération de filtrage qui vise à sélectionner parmi toutes les entités logicielles qui implémentent une même fonctionnalité désirée celles les plus adaptées aux informations contextuelles de l'appareil mobile. Nous disons qu'une entité logicielle est conforme à l'appareil mobile cible si toute les contraintes d'exécution de cette entité sont satisfaites. » .[29]

2.3.3 la tâche de composition

Notre travail repose principalement sur la tache de composition qui est destiné de composer une application mobile en utilisant des entités logicielles préexistant. Plusieurs travaux ont été proposés pour considérer cet axe de recherche parmi eux nous citons celui de Rosa et Lucena [61] et celui de Furno et Zimeo [62]. Les auteurs dans ces travaux de recherche traitent la tâche de composition en utilisant soit des composants soit des services selon les informations contextuelles de l'environnement d'exécution. Ces travaux limitent les objets de composition à un seul type et l'application composite sera dédiée seulement pour être exécutée dans un environnement spécifique, limiter l'utilisation de l'application.

Pour cause des exigences des utilisateurs et aussi du contexte de l'appareil mobile cibles, un développeur se trouve parfois forcé de combiner des entités logicielles hétérogènes comme nous allons traiter dans ce manuscrit.

Un exemple de composition est : une combinaison de deux services web nommés *GeoIPService* et *GlobalWeather* dont l'objectif est de retourner à partir de la localisation les conditions météorologiques courantes.

Le résultat de cette composition est un service composite de météorologie nommé *MyMeteo*. Ce composite agit comme suivant : le service*GeoIPService* obtient l'emplacement courant de l'utilisateur, après le service *GlobalWeather*fournit l'utilisateur par les conditions météorologiques courantes en se basant sur les cordonnées obtenues par le premier service.

Par conséquent, l'étape de composition correspond à l'établissement de la collaboration entre les entités logicielles qui ont été sélectionnées. Le résultat final est la construction de la composition d'entités encapsulées sous la notion d'une entité composite [29].

Il s'agit des compositions de services à travers les orchestrations de services pour avoir un service composite, des compositions dans les approches à base de composants à travers les assemblages de composants afin d'arriver finalement à un composant composite ou également la composition des entités hétérogènes.

2.3.3.1 Collaboration entre service :

Pour accomplir une tâche complexe, il est parfois nécessaire de composer plusieurs services plus simples en les connectant adéquatement. Les services échangent des messages pour se synchroniser dans leur exécution et se procurer des données. Une telle collaboration entre services est appelée composition de services. La composition de services web et donc le processus consistant à combiner des services existants pour former de nouveaux services.

La collaboration entre service s'effectue suivant deux types de schémas de collaboration, l'orchestration et la chorégraphie[59].

- **L'orchestration**

Dans l'orchestration le processus principal (orchestrateur) prend le contrôle du déroulement de la composition en coordonnant les différentes opérations des différents services web. À aucun moment les autres services servant à la composition n'ont connaissance de cette composition (Figure 8): ils remplissent leur rôle de service sans se soucier si un client humain ou applicatif interagit avec eux.

L'orchestration est une méthode très utilisée, et pour plusieurs raisons :

- Il est relativement simple d'écrire un processus centralisé gérant l'invocation de sous services ;
- Dans le cas d'une réutilisation de services web basiques, seule la partie centrale est à développer [59].

- **La chorégraphie :**

La chorégraphie ne repose pas sur un processus principal. Chacun des services intervenant dans la composition sait exactement ce qu'il doit faire, quand il doit le faire et avec qui. Donc ils ont tous une connaissance plus ou moins globale du processus métier dans lequel ils se retrouvent (Figure 2.4). Contrairement à la méthode basée sur l'orchestration, la chorégraphie demande nettement plus de développement et de test : il faut développer chaque service Web pour qu'il participe correctement dans la composition qui l'utilise. Cela dit, en utilisant des méthodes de développement et des stratégies bien pensées, l'intérêt de la méthode réside dans la décentralisation des traitements [59].

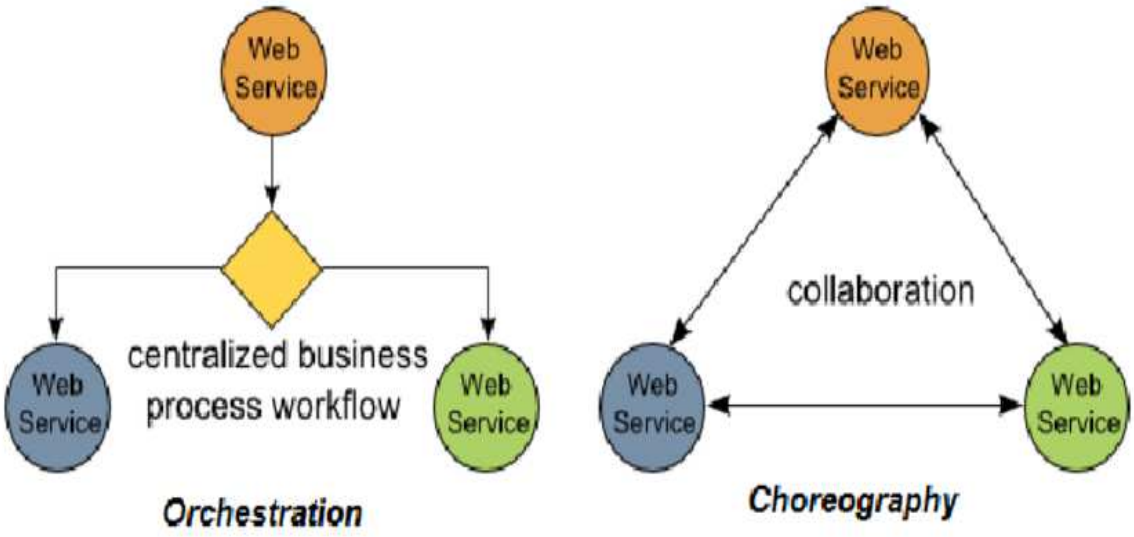


Figure 2.4 orchestration et Chorégraphie de service web [59].

Il y a une différence importante entre l'orchestration et la chorégraphie de services web. L'orchestration se base sur un processus métier exécutable pouvant interagir avec les services web internes ou externes. L'orchestration offre une vision centralisée, le processus est toujours contrôlé du point de vue d'un des partenaires métier.

La chorégraphie est de nature plus collaborative, chaque participant impliqué dans le processus décrit le rôle qu'il joue dans l'interaction.

L'approche la plus simple à notre avis pour les Architecture Orientées Service est l'orchestration. En effet, le but principal de la composition est la réutilisation de services (basiques ou composites) sans modifier ceux-ci. Ces services peuvent être hébergés par une autre compagnie et nous n'avons alors aucun moyen de contrôle sur ceux-ci.

Le processus principal doit alors pouvoir s'adapter aux différentes erreurs possibles (exp : non disponibilité d'un service, annulation, etc..). Initialement les standards se sont intéressés soit à l'orchestration soit à la chorégraphie. Récemment, un nouveau standard combine les deux stratégies [63].

2.3.3.2 Assemblage de composant

L'approche à composants est relativement récente dans l'histoire du génie logiciel, cette approche promeut la construction d'applications à partir de l'*assemblage* de *composants* [64]. De façon simpliste, un composant peut être décrit comme étant une 'brique' logicielle préfabriquée qui est conçue pour être composée, c'est à dire assemblée, avec d'autres composants [65]. Un composant est *réutilisable*, c'est à dire qu'un même composant peut être employé dans la construction de différentes applications, et sa réutilisation ne nécessite, à priori, pas de connaissances sur l'implémentation. L'approche à composants est fondée sur l'idée que le développement et l'assemblage de composants peuvent être réalisés de façon totalement séparée par des acteurs différents et dans des emplacements différents. Un *modèle composant* définit les caractéristiques des composants, de leurs assemblages et est accompagné par un support d'exécution.

Le modèle à composants permet de réaliser le développement et l'exécution d'applications à base de composants.

L'interaction entre composants reflète un prédicat d'interaction nommé connecteur.

Ce dernier sert à spécifier qu'un *composant 1* interagit avec *composant 2* où un service fourni par un *composant 1* via son interface fournie sera un service requis par le *composant 2* via son interface requise.

Un connecteur est un bloc de construction destiné pour exprimer les interactions entre composants ainsi que les règles qui gouvernent cette interaction. Autrement dit, les connecteurs sont des entités architecturales qui relient un ensemble de composants et agissent en tant que médiateurs entre eux.

Par exemple, les connecteurs peuvent prendre des formes simples d'interaction, comme des appels de procédure ou l'émission d'évènements ou des valeurs/variables. etc. Ces connecteurs peuvent également représenter des interactions complexes ou être associés avec des médiateurs dans le cas d'une coordination hétérogène au niveau des données échangées entre composants[66,67].

Les connecteurs peuvent ne pas correspondre à des unités de compilation comme pour les composants.

L'approche à composants est motivée d'un côté par des arguments économiques, tels que la réduction du temps et du coût de développement, ou bien la spécialisation des acteurs intervenant dans le cycle de vie du développement [68], et d'un autre côté par la proposition de solutions pratiques à des limitations d'approches telle que l'approche orientée objet [69]. L'approche à composants promeut notamment l'emploi de la composition au lieu de l'héritage comme mécanisme de réutilisation, le support pour la livraison et le déploiement indépendant des composants ainsi que la prise en compte de l'existence d'aspects non-fonctionnels.

2.4 Assemblage de composants vs collaboration de services

La composition d'applications dans le contexte de l'ingénierie logicielle basée composant sou orientée services a pour objectif de maximiser la réutilisabilité en terme de modularité qui est directement issue de l'objet et d'améliorer le développement d'applications par assemblage de composants logiciels ou l'orchestration/chorégraphie de services.

CBSE et SOSE sont deux paradigmes très similaires qui se basent sur la construction d'applications à partir d'entités logicielles existantes, composants ou services [36]. Ils partagent aussi le même processus de développement global qui se compose des activités d'identification des entités logicielles (composant ou service) qui répondent aux besoins, puis de coordination de ces entités pour construire l'application composite finale. Par conséquent, ils reposent sur les mêmes notions de composition pour la construction de nouvelles entités dites composites à partir de l'existant afin de garantir une approche homogène où toute entité peut être vue comme un composant ou un service. Cette approche permet une manipulation des structures de description claires en terme de composition de composites Un point de différence importante entre l'assemblage de composants et l'orchestration de services c'est que dans la plupart des cas les composants cachent le flux de données à l'intérieur d'eux-mêmes. L'interaction entre ces entités logicielles constituantes de l'application à composer met en exergue une notion clé qui est le *couplage*. Brel dans [47] a identifié cette notion comme un des points de rupture clé entre les deux paradigmes CBSE et SOSE. Comme ils ont déclaré que réduire le couplage garantit un certain nombre de bénéfices intuitifs en termes d'isolation des erreurs, de facilitation d'ajouts et de retrait d'entités réutilisées, reconfiguration, etc.

Les types de composition présentés par ce cadre de travail mettent en place un couplage faible entre les entités constituantes qui permettent une grande réutilisation de celles-ci. La différence entre les deux mécanismes de composition c'est que l'orchestration de services sera explicite où les cheminements des données et leurs transformations doivent être définis au préalable. Contrairement à l'opération d'assemblage où ceux-ci peuvent être vus comme des boîtes noires et le cheminement des données n'est pas à priori connus.

2.5 La composition hétérogène : multi paradigme

Plusieurs travaux limitent la composition d'une application à base d'un seul type et l'application composite sera dédiée seulement pour être exécuté dans un environnement spécifique. Cependant, la création d'applications basées uniquement sur les services peut se révéler complexe et difficilement adaptatives puisque les services découverts ou plus exactement leurs interfaces doivent être connus au moment du la conception. Par ailleurs, une application basée uniquement sur les composants peut difficilement gérer les variations du contexte d'exécution et les communications entre dispositifs. La combinaison entre l'approche à base de composant et orienté service est nécessaire. Plusieurs études (exp équipe *INRIA)* ont été consacrées pour la gestion des hétérogénéités où plusieurs systèmes multi-paradigmes ont donc vu le jour (ex. SOA a.k.a. SOA 2.0 qui utilise les services et les événements, SCA [70], SLCA [71].

SCA (Service Component Architecture) est un modèle d'architecture à composants qui visent à simplifier la création des applications et systèmes par composition de services. Autrement dit, il correspond à la création d'applications orientées services à base d'assemblages de composants SCA. SCA accepte un grand nombre de langages de programmation tel que Java, PHP, C++, Cobol, BPEL, XSLT, SQL et XQuery, C#, chacun doté de son propre système de spécifications .L'implémentation dans SCA peut s'appuyer sur des services fournis par d'autres composants dont elle dépend. Ces dépendances sont appelées références. Elles sont associées à services qui peuvent être soit exposés par d'autre composants SCA soit exposé par des systèmes tiers (Web services, connecteurs JCA, …).

Cela prouve sa capacité de gérer l'hétérogénéité et la dynamicité nécessaire à l'orienté services [28].

Un autre modèle classé dans la catégorie des approches hybrides est SLCA (Service lightweight Components Architecture). SLCA est un autre modèle classé comme approches hybrides. Son objectif principal est de définir une architecture dynamique pour la composition de services en exploitant plusieurs paradigmes existants: architecture orientée services Web, assemblage léger de composants et Evénements [28].

2.6 La gestion d'heterogeneite des donnees echangees

Le défi de l'hétérogénéité ne peut concerner que les briques logicielles mais il peut également affecter les données échangées entre ces entités constitutives. En fait, plusieurs études ont été consacrées à l'aspect technologique de la gestion de l'hétérogénéité des données échangées.

- Hock-Koon a traité cette question pour son service composite proposé [72]. Il rassemble les préoccupations d'invocation qui reflètent le déclenchement de l'exécution des services constitutifs et la médiation qui représentent la capacité du composite à garantir une bonne compréhension des données échangées entre ses services constitutifs.

- Dans [55], les auteurs proposent une méthode de composition de services qui cible les problèmes d'hétérogénéité des données. Les services sont organisés dans un graphique représentant l'ensemble des compositions possibles. Lors d'une incompatibilité de données entre services, le système utilise le graphe de composition pour identifier une succession de services capables d'assurer les modifications nécessaires sur les données a priori incompatibles.

- Derdour dans ses recherches [66,67] s'est attaqué à ce problème pour les architectures de logiciels multimédias. Il a déclaré que le problème de l'hétérogénéité est basé sur le flux de données multimédias échangées (par exemple, image, son, texte). Compte tenu de ce fait, il a proposé un méta modèle MMSA (Architecture logicielle multimédia méta-modèle) pour les architectures multimédias, ce méta modèle permettant de décrire les systèmes multimédias comme un ensemble de composants qui gèrent différents types et formats de données multimédia et interagissent avec eux via des adaptateurs. Ce modèle permet de faciliter la tâche d'adaptation entre des supports de même.

2.7 Les approches de composition

Nous allons présenter quelques approches de composition déjà existantes touchant les notions clés liées à notre travail. L'objectif est de faire apparaitre plus précisément la problématique abordé, les motivations derrières ce travail et d'aborder les objectifs visés.

Les auteurs dans [73]ont décrit certaines issues liées à la composition de services dans un environnement mobile. Ils ont présenté un protocole de composition de services distribué pour les environnements mobiles qui prend en considération la mobilité, changement dynamiques dans la topologie de service et les ressources des dispositifs. Ils se sont concentrés principalement sur une architecture distribuée pour faciliter la tâche de composition mais en négligeant ses capacités d'adaptation.

Dans [74], les auteurs ont présenté une approche automatique pour la composition de services web tout en attaquant le problème de l'hétérogénéité des processus et les hétérogénéités des données en utilisant un planificateur et un médiateur de données. En revanche, les auteurs dans [72] étaient censés réifier les notions pertinentes des mécanismes de composition existants dans un méta-modèle de service composite. Ce méta-modèle définit toutes les fonctionnalités entrelacées et fournit une vision globale et explicite de la composition de service. En outre,

Cette approche permet la spécification du processus d'auto-composition dont la composite a la capacité de modifier dynamiquement son architecture et ses logiques de composition selon le contexte de l'environnement.

Dans [61], les auteurs ont créé une infrastructure de logiciel appelé *AppSpotter* qui rend possible la sélection et la composition dynamique et automatisé des composants logiciels pour la création d'applications mobiles. Ils ont proposé un mécanisme pour la sélection de composants logiciels en se basant sur les caractéristiques de l'appareil mobile. Dans [62], les auteurs ont présenté un modèle pour concevoir des services sensibles au contexte qui peut être exploité comme un domaine flexible pour générer automatiquement des compositions sensibles au contexte au moyen d'un outil spécifique. En outre, l'outil exploite une extension de la définition du problème qui comprend également la représentation du contexte pour étendre les services.

Chacune de ces approches tente de résoudre quelques problèmes particuliers de différents points de vue : la composition en terme de composants, de services ou d'entités logicielles hétérogènes ; l'environnement d'exécution : dispositif mobile ou autre (ex. les ordinateurs) ; les capacités d'adaptation en terme de construire des applications contextuelles ou non adaptatives ; comme illustre le Tableau présenté ci-dessous (cf. Tableau 2.1).

Ces multitudes d'approches avec ses caractéristiques souvent spécialisées n'ont pas une vision globale de la composition d'applications mobiles. Ces approches limitent la réutilisabilité de l'application composite pour un contexte spécifique qui ne pas pourra être celui qui est nécessaire, le contexte courant. Un besoin naissant est alors de pouvoir obtenir plusieurs versions de la même application composite désirée selon plusieurs informations contextuelles de différents appareils mobiles et donc la nécessité de prendre en compte les configurations hétérogènes offertes par eux ainsi que les ressources limitées et leur état d'exécution

	Objet de composition			Type de composition		Capacité d'adaptation		Approche définit au	
	Entité composant	Entité service	Entité hétérogène	Micro composition	Macro composition	Contexte courant	Non adaptative	Niveau architectural	Niveau application
Chakraborty et *al.*(2005)		x		x		x		x	
Zixin et *al.*(2007)		x			x		x	x	x
Hock-Koon and Oussalah (2010)		x			x	x		x	
Ricardo and Vicente (2011)	x			x		x			x
Furno et *al.*(2014)		x			x	x		x	x
processus de composition proposé par Djeddar et al.(2014)(sous processus d'adaptation)	x	x	x	x		x		x	x

Tableau 2.1 un cadre de comparaison d'approches de composition [28]

De plus, mis à la disposition de l'utilisateur, celui-ci peut avoir besoin de plusieurs applications pour combler ses propres exigences. Par conséquent, pour cause du contexte et des exigences des utilisateurs un besoin émergent est alors de combiner des entités logicielles déjà existantes tout en ignorant comment elles sont implémentées, indépendamment de leurs plateformes d'exécution, afin de tirer profit de leurs fonctionnalités. L'objectif est d'obtenir des applications mobiles comblant les exigences des utilisateurs et avoir des applications composites adaptables à son environnement d'exécution.

La dernière ligne du Tableau reflète le positionnement de notre processus utilisé, proposé par Djeddar et al [28] par rapport aux différentes approches de composition citées précédemment en fonction des différents critères proposés.

Le processus proposé [28] permet de traiter plusieurs types de composition en terme de collaborations - hétérogènes/homogènes - effectuées. Donc, il couvre plusieurs points d'hétérogénéité que ce soit en termes d'objets de composition ou d'environnement d'exécution.
Par conséquent, il permet la composition des applications mobiles adaptatives en utilisant des entités logicielles hétérogènes en commençant par la définition des différentes fonctionnalités souhaitées jusqu'à l'obtention d'un modèle architectural détaillé pour l'application composite incluant tous les adaptateurs nécessaires. En ce qui concerne les applications composites hétérogènes ce processus a introduit la structure conceptuelle pour les différents médiateurs proposés pour éliminer cette hétérogénéité et donc permettre une composition consistante.

Notre modeste travail est destiné à la réalisation de ces médiateurs proposés en utilisant l'architecture détaillé qui est obtenu par ce processus de composition.

2.8 Conclusion

Dans ce chapitre nous avons explicité au premier temps sur les représentations de deux approches importantes ; l'approche basée sur des composants et l'autre sur des services pour faciliter la tâche de développement et d'un autre côté pour favoriser la réutilisation de l'existant.

Dans ce chapitre nous nous sommes particulièrement intéressées au processus de composition, nous avons présenté plusieurs définitions proposées pour cette technique où cette dernière peut faire l'objet sur les composants ou bien les services. D'après ces définitions, nous avons présenté chacune de ces étapes tout en discutant quelques travaux apportés pour chacune d'entre elles.

Ce chapitre a présenté un cadre comparatif entre les objets de composition et aussi entre les techniques de composition proposées pour chacun d'entre eux ; pour y arrivera la fin à l'apparition des approches hybrides qui sert à tirer profit de plusieurs paradigmes existants ce qui conduit à l'émergence de la nouvelle notion *multi-paradigmes*. En plus de l'hétérogénéité du type des objets à composer, la tâche de composition présente un autre défi important qui est l'hétérogénéité qui porte sur les données échangées entre ces objets. Face à ce constat, nous avons présenté les travaux qui traitent cette issue. Nous avons conclu ce chapitre par une comparaison de quelques approches de composition liées à notre travail suivant un ensemble de critères bien définis.

Le chapitre qui suis sera consacré sur le nouveau paradigme de composition proposé [28] que nous allons l'utiliser dans notre travail, en déclarant toutes les défis que nous allons traiter et enfin nous mettent l'accent sur l'approche d'adaptation.

Chapitre3 : L'Approche d'adaptation pour la composition hétérogène des applications mobiles.

Chapitre 3 L'approche d'adaptation pour la composition hétérogène des applications mobiles

3.1 Introduction

le nombre d'utilisateurs des application mobile est en progression, grâces à la satisfaction d'un large éventail des besoins des utilisateurs, pour cela le développeur trouve que la combinaison entre les entités logicielles existants est apparu nécessaire pour composer une nouvelle application mobile ce qui favorise la réutilisation de l'existent ,cependant la tache de composition d'une application mobile est la satisfaction des exigences des utilisateurs ,par conséquent durant cette tache le développeur vise à faire face au défi de l'hétérogénéité qui porte sur différentes entités logicielles à composer et d'autre porte sur les données échangées,

ces derniers ceux qui reflètent notre axe de travail .Face à ce constat, Nous présentons une approche d'adaptation proposé pour la composition hétérogène des applications mobiles qui nous l'appris dans notre travail pour réaliser des médiateurs d'adaptation afin d'éliminer cette hétérogénéité et donc permettre une composition consistent.

Dans ce chapitre nous allons particulièrement intéresser au nouveau paradigme proposé [28] en détaillant les défis de l'hétérogénéité comme nous allons traiter explicitement les contributions élaborées dans ce manuscrit. Ce chapitre présentera en détail l'approche adaptation proposé pour l'application mobile souhaité fonctionne correctement.

3.2 Nouveau paradigme hybride propose

Un nouveau paradigme hybride proposé [29] est dédié au développement de logiciels pour les environnements mobiles à travers la réutilisation des entités logicielles existantes est dirigé par les exigences des utilisateurs et le contexte du périphérique mobile qui sera utilisé pour mettre en œuvre le produit final. Les entités logicielles concrètes sélectionnées en fonction du contexte d'exécution pour mettre en œuvre les fonctionnalités souhaités peuvent être hétérogènes à la fois en termes de formes de mise en œuvre ou d'échange de données entre elle. L'application résultante sera une application qui reflète un ensemble d'entités logicielles de types différents et/ou les entités échangés .

Dans notre travail nous utilisons le nouveau paradigme qui a été déjà proposé [29] pour composer une application mobile à base des entités logicielles de différentes formes d'implémentation (service, composant).

Ce nouveau paradigme permet la composition des applications mobiles adaptatives en utilisant des entités logicielles hétérogènes en commençant par la définition des différentes fonctionnalités souhaitées jusqu'à l'obtention d'un modèle architectural détaillé pour l'application composite incluant tous les adaptateurs nécessaires. Cette architecture représente un point de départ de notre travail pour réaliser ces adaptateurs de médiation

3.3 La description architecturale d'un nouveau paradigme

La tâche de composition présentée par le processus proposé est dirigée par les exigences des utilisateurs et le contexte de l'appareil mobile cible. Les entités logicielles concrètes choisies en fonction des besoins des utilisateurs et du contexte pour implémenter les fonctionnalités désirées peuvent être hétérogènes que ce soit au niveau de leur type ou de leur nature. A cet effet, un méta-modèle nommé *H*2 CMA-AD (Heterogeneous ou Homogeneous Composite Mobile Application - Architectural Description) qui a été déjà proposé pour décrire des applications mobiles composites hétérogènes aussi bien qu'homogènes au niveau architectural (Figure 3.1, les classes en blanc).

L'objectif de ce méta-modèle est de pouvoir représenter n'importe quelle application mobile quelque soient les détails d'implémentation de ses entités logicielles constitutives. Donc, ce méta-modèle décrit un formalisme qui nous permet d'élaborer une composition hétérogène. Ce formalisme sert à représenter l'application composite via les entités logicielles concrètes choisies pour implémenter les fonctionnalités requises. Un grand avantage de notre processus de composition est de ne pas limiter le choix technologique de ces entités concrètes. Cependant, le formalisme proposé vise à spécifier les points d'hétérogénéité dans le cas de coordinations hétérogènes. On outre, il nous donne la possibilité d'associer ces relations de composition avec des médiateurs afin de remédier les issues d'hétérogénéité [29].

Une entité logicielle est une fonction qui prend en entrée un ensemble de paramètres nécessaires pour qu'elle fonctionne et renvoie en sortie le résultat souhaité. Cette fonction doit être exécutée selon un ensemble de conditions que nous avons regroupées dans un profil d'exécution. Par conséquent, nous définissons une entité logicielle comme un quadruplé :

La fonction à exécuter, les données d'entrée et de sortie, le profil d'exécution. Suivant le méta-modèle illustré dans la Figure 3.1 une application mobile composite (C) est constituée d'un ensemble d'entités logicielles (C1) liées entre-elles via des connecteurs (C2) où :

- Chaque entité logicielle peut être un composant (C3), un service (C4) ou une entité logicielle composite (C5) (service composite, composant composite, etc.)
- Chaque entité logicielle possède des données d'entrée (C6) et de sortie (C7) et également un profil d'exécution (C8) contenant toutes les conditions nécessaires pour son exécution.
- L'échange des données entre les entités logicielles reliées est effectué via des ports fournis (C9) et des ports requis (C10) pour les entités de type composant et via des services fournis (C11) et des services requis (C12) pour les entités de type service.
- Les différentes relations entre les entités constitutives sont des liens de précédence (C13) et des liens d'utilisation (C14) où ces connecteurs seront attachés avec des *médiateurs endogènes* (C15) ou *exogènes* (C16) dans le cas d'une coordination hétérogène.
- L'application mobile globale est dédiée à être exécutée dans un environnement mobile spécifique (C17) avec un contexte d'exécution (C18) bien déterminé.

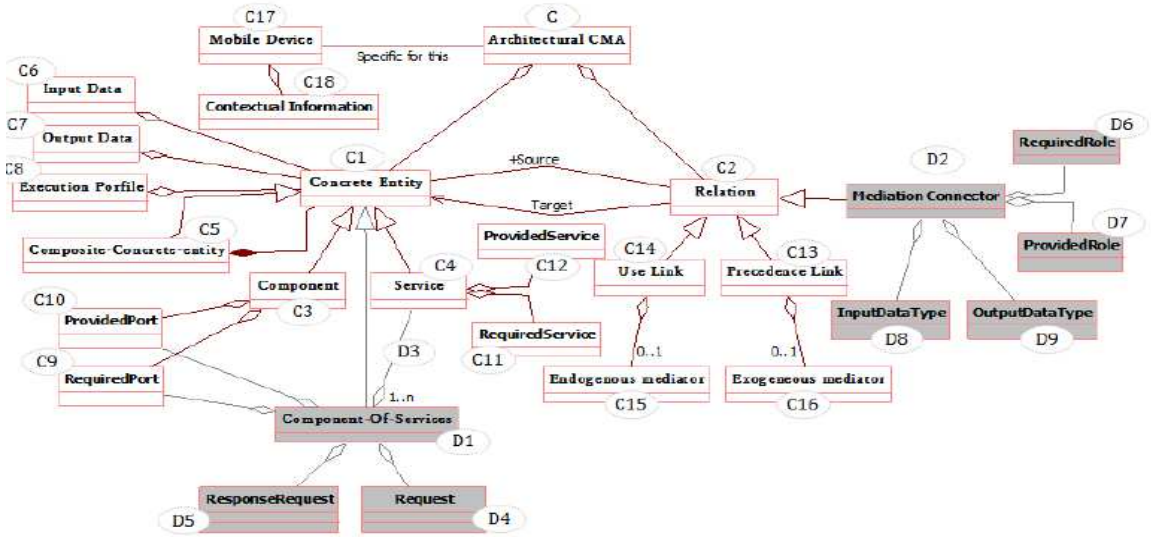


Figure 3.1 le méta model CMA-FD [29]

Cette description architecturale vise à représenter la CMA souhaitée avec des services ou des composants séparément ou avec des entités logicielles hétérogènes pour cause du contexte de l'appareil mobile et des exigences souhaités. Par conséquent, elle permet la création d'applications mobiles adaptatives comme une *composition endogène* ou *exogène*. Via ces deux types de composition nous pouvons avoir deux types d'applications :

- Une application homogèn*e* constituée d'un ensemble d'entités du même type manipulant des données du type homogène.
- Une application hétérogène reflétant un ensemble d'entités du différents types reliées entre-elles via des médiateurs et/ou les données échangées entre-elles nécessitent des transformations pour qu'elles soient compréhensibles. Lors de la tache de composition on trouve un défi d'hétérogénéité qui porte sur les entités logicielle et d'autre porte sur les données échangées [29].

❖ **L'hétérogénéité qui porte sur les entités logicielle :**

La tâche de composition à base d'un paradigme proposé porte un défi d'hétérogénéité entre les entités à composer. Cependant ces derniers ne peuvent pas communiquer directement en raison de l'incompatibilité de leurs interfaces de communication (interface hétérogène), on dit que ces entités logicielles associées ne sont pas du même type. À titre d'exemple, entité logicielle de type composant connectée à une autre de type service.

❖ **L'hétérogénéité qui porte sur les données échangées :**

Les données échangées entre les entités logicielles à composer nécessitent quelques transformations pour les rendre compréhensibles. Si les entités logicielles liées ne peuvent pas communiquer directement en raison du fait que les données échangées entre elles ne sont pas compréhensibles, nous affirmons que ces entités liées n'ont pas la même nature. À titre d'exemple, la fonctionnalité Aquire photo, qui permet d'acquérir une image de produit que nous souhaitons acheter, fournit une image de type JPG, tandis que la fonctionnalité lire code à barres, utilisée pour extraire le code à barre du produit à partir de l'image acquise, nécessite image de type WebP afin qu'elle puisse fonctionner correctement.

Grace à ces défis le processus proposé vise à adresser et traiter ces problèmes d'hétérogénéité en utilisant une approche d'adaptation pour éliminer cette hétérogénéité.

3.4 L'approche d'adaptation pour la composition mobile hétérogène

Cette approche proposée vise à adresser et traiter ces problèmes d'hétérogénéité. Faire face à la coordination hétérogène pour réaliser les médiateurs d'adaptation nécessaires afin d'éliminer cette hétérogénéité et donc permettre une composition consistante. Deux types de médiateurs doivent être réalisés :

- Les médiateurs endogènes : ce type de médiateurs est destiné à éliminer l'hétérogénéité entre deux entités logicielles de différentes natures qui ne peuvent pas communiquer directement.

- Les médiateurs exogènes : ce type de médiateurs vise à surmonter l'hétérogénéité entre deux entités logicielles qui n'ont pas le même type d'implémentation.

Cette approche d'adaptation traite le défi d'hétérogénéité selon quatre types de composition :

1. Composition hétérogène exogène : composition d'entités logicielles de différents types qui n'ont pas la même nature en attachant des médiateurs exogènes et endogènes.
2. Composition homogène exogène : composition d'entités logicielles de différents types qui ont la même nature en attachant des médiateurs exogènes.
3. Composition hétérogène endogène : composition d'entités logicielles du même type qui n'ont pas la même nature en attachant des médiateurs endogènes.
4. Composition homogène endogène : composition d'entités logicielles du même type qui ont la même nature par une relation simple et donc sans adaptation.

Type de composition	Problèmes d'hétérogénéité	Médiateurs proposés
Composition hétérogène exogène	Entités de différentes natures/Entités de différents types	Médiateurs endogènes/Médiateur Exogène
Composition homogène exogène	Entités de différents types	Médiateur exogènes
Composition hétérogène endogène	Entités de différentes natures	Médiateurs endogènes
Composition homogène endogène	Aucun problème d'hétérogénéité	Aucun médiateur

Tableau 3.1 : les types de composition traités par l'approche d'adaptation

3.5 Spécification des médiateurs proposés

L'approche d'adaptation n'est pas seulement destiné à la détection des points d'hétérogénéité mais aussi d'intégrer des médiateurs et de décrire leur fonctionnement pour remédier aux problèmes d'hétérogénéité posés lors de la composition .En premier lieu on spécifier les rôles des médiateurs d'adaptation intégrés pour l'application mobile composite souhaités :

D'une part, les médiateurs exogènes visent à assurer la communication entre deux entités logicielles de différents types. Vu que les entités logicielles ne peuvent pas communiquer en raison de leurs types d'implémentation hétérogènes, les médiateurs exogènes indiqué dans le modèle architectural sera remplacé par une entité de type composant nommé *Component-of-services* (D1) qui encapsule les entités de type services afin d'éliminer l'hétérogénéité entre les entités de différents types en construisant des interfaces de communications bien-formées communes afin de tirer profit de ses services mais juste en manipulant les entrées et les sorties nécessaires indépendamment de leurs détails d'implémentation. Plus précisément, dans le cas d'une coordination exogène entre un composant et un service, le service sera encapsulé (D3) dans la nouvelle entité *Component-of-services*. Ce dernier peut encapsuler un ou plusieurs services collaborés (Figure 3.2) où il joue le rôle du moteur défini dans l'orchestration de services. Il vise à déclencher l'exécution de l'entité de type service qu'il englobe en fournissant dans un premier temps les données requises obtenues via son port requis par le biais d'une requête déclenchée (D4) et de diffuser le résultat obtenu via son port fournis en déclenchant une autre requête de réponse (D5). Cela assure une communication compatible entre les entités sources et cibles via des interfaces communes.

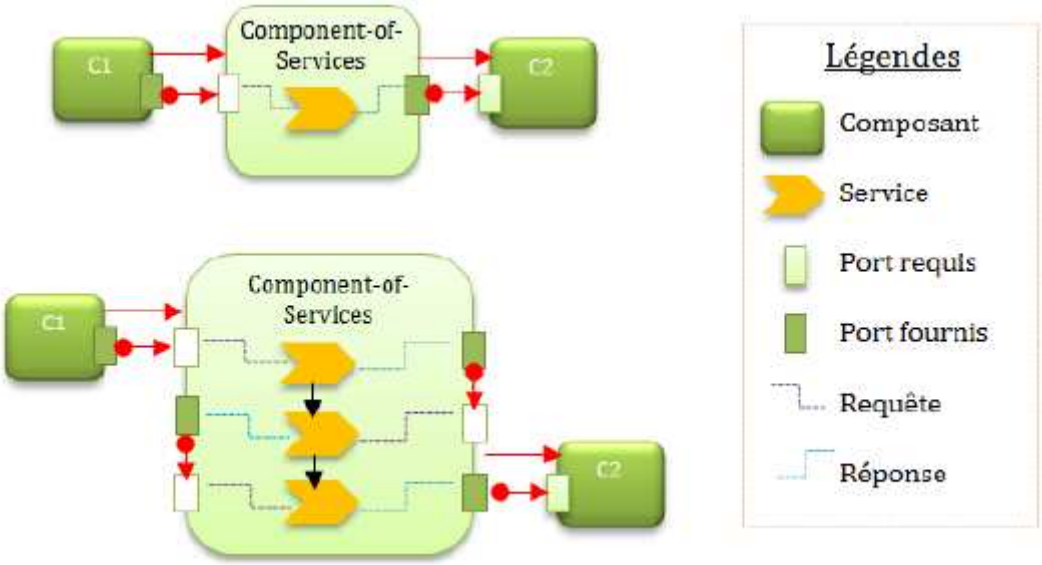


Figure 3.2 Spécification d'un *médiateur exogène[29].*

D'autre part, les médiateurs endogènes représentent les services de médiation qui seront sélectionnés pour assurer la transformation des données échangées qui sont hétérogènes. Dans ce cas-là, les liens qui sont attachés par ce type de médiateur seront des connecteurs présentant des interactions complexes. Ces connecteurs ont pour objectif de convertir les données échangées en appelant les services de transformation appropriés tout en utilisant ses rôles (D6) (D7) pour recevoir et diffuser ces données.

Le formalisme proposé représente ce connecteur comme une glu qui définit une fonction qui sert à chercher le service de médiation approprié (*searchMD*: chercher un service de médiation) en se basant sur les types des données échangées (D8) (D9) dans la bibliothèque de services de médiation ; et une autre qui sert à appeler le service de transformation trouvé (*callMD*: appeler un service de médiation) afin d'assurer la compatibilité des données échangées (Figure3.3).

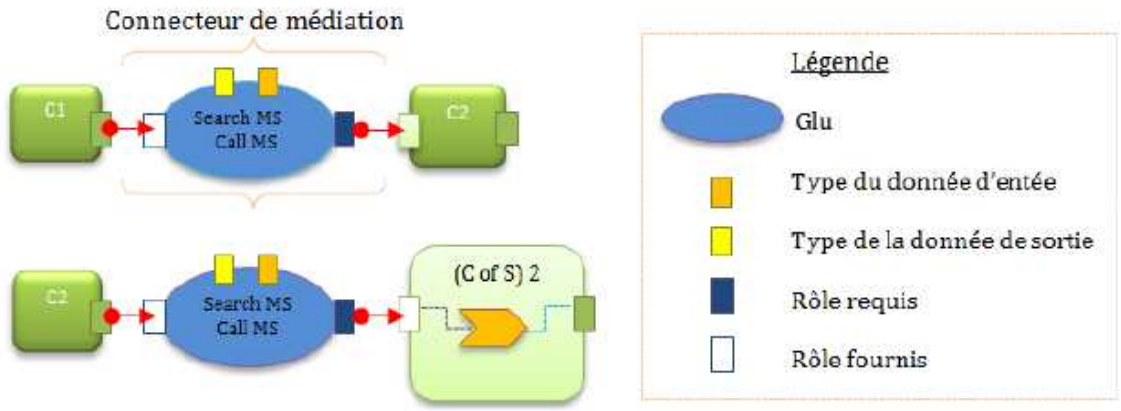


Figure 3.3 Spécification d'un médiateur endogène [29].

Dû au fait que les médiateurs exogènes vont éliminer l'hétérogénéité entre deux types différents des entités logicielles en encapsulant les services dans un composant spécifique, les liens d'utilisation vont alors ne relier que (component/component) ou (component/Component-of-services).

3.6 Le mécanisme d'adaptation

❖ **Encapsulation des services :**

Encapsulation fait à chaque fois où les liens de précédences détecte un *médiateur exogène* l'entité de type service dans une nouvelle entité nommée *Component-of-services* où :

- Le service requis par l'entité service sera une requête déclenchée par la nouvelle entité c'est-à-dire l'entité *Component-of-services* fait un appel à l'entité service tout en fournissant les données nécessaires pour son exécution qui sont obtenues via son port requis.
- Le lien d'utilisation qui relie l'entité source et l'entité service entre un port fourni et un service requis, sera un lien d'utilisation entre l'entité source et l'entité encapsulant entre un port fourni et un port requis.
- Le service fourni par l'entité service sera transmis vers le port fournis du composant encapsulant en déclenchant une requête de réponse.

- Le lien d'utilisation qui relie l'entité service et l'entité cible, entre un service fournis et un port requis, sera un lien d'utilisation entre l'entité encapsulant et l'entité cible, entre un port requis et un port fournis.

L'encapsulation peut dépasser un seul service dans le cas où le service encapsulé soit relié par d'autres services du côté cible. Dans ce cas-là, tous les services reliés seront encapsulés dans le même composant encapsulant jusqu'à ce que nous arrivions à une relation entre un service et un composant. Donc, l'encapsulation est le terme proposé pour refléter le regroupement d'un ou plusieurs entités services afin d'unifier les interfaces de communication entres les entités hétérogènes reliées. Les relations de composition entre les services encapsulés seront telles qu'illustrées sur la Figure 3.3. Le composant encapsulant fait à chaque fois un appel à un service selon l'ordre d'invocation défini via une requête en fournissant les données nécessaires obtenues via son port requis et reçois le résultat obtenu via son port fournis [29].

❖ Les services de médiations

Notre approche vise à gérer les liens de l'utilisation attachés par des médiateurs endogènes afin de les remplacer par des connecteurs complexes qui sont les connecteurs de médiation où :

- Un connecteur de médiation sera un lien de composition entre (component/component) ou (component/Component-of-services).
- Un port fournis de l'entité source sera relié avec le rôle requis du connecteur de médiation avec un lien d'utilisation.
- Un rôle fournis du connecteur de médiation sera relié avec le port requis de l'entité cible.

Le modèle architectural détaillé obtenu représente une description détaillée de l'application mobile composite incluant tous les adaptateurs nécessaires pour assurer le bon fonctionnement pour un contexte mobile spécifique où cette architecture associée par un profil d'exécution contenant toutes les informations contextuelle de l'appareil mobile cible.

3.7 Conclusion

Dans ce chapitre, nous avons abordé au premier temps présentation sur le nouveau paradigme hybride proposé pour la composition mobile hétérogène. Par la suite nous avons abordé une description architecturale de ce processus Vu que la composition dirigée par le contexte d'exécution et les besoins des utilisateurs peuvent soulever plusieurs problèmes d'hétérogénéité, en détaillant ces défis pour cela on a présenté l'approche d'adaptation qui est destiné à intégrer des médiateurs et de décrire leur fonctionnement pour remédier aux problèmes d'hétérogénéité posés lors de la composition.

Dans le chapitre qui suit, nous allons réaliser le mécanisme d'adaptation en abordant l'implémentation des points contributifs présentés dans le chapitre suivant.

Chapitre IV: Réalisation de l’approche d’adaptation proposée

Chapitre 4 Réalisation de l'approche d'adaptation proposée

4.1 Introduction

Le chapitre précédent a définit l'approche d'adaptation au niveau architectural pour la composition mobile hétérogène, nous avons introduit la structure conceptuelle pour les différents médiateurs proposés pour éliminer cette hétérogénéité et donc permettre une composition consistante. Cependant nous visons dans ce chapitre à définir cette approche au niveau applicative, pour ce faire nous devrons réaliser les médiateurs exogènes en fournissant la structure concrète de la nouvelle entité Component-of-services ainsi que les médiateurs endogènes en fournissant la structure concrète au service de médiation proposé. Ce dernier nécessite de construire une bibliothèque de services de médiation selon les transformations requises des données échangées afin d'appeler par la suite celui qui est nécessaire pour assurer la compatibilité des données transmises.

4.2 Environnement de développement

- **Android Studio**

- Android Studio est un nouvel environnement pour le développement et programmation entièrement intégré qui a été récemment lance par Google pour les systèmes Android, il a été conçu pour fournir un environnement de développement et une alternative à Eclipse qui est l'IDE le plus utilisé
- Android Studio permet de voir chacun des changements visuels que vous effectuez sur votre application et en temps réel, vous pourrez voir aussi son effet sur différents appareils Android.

- Android Studio offre aussi d'autres choses :
    - ✓ Un environnement de développement robuste.
    - ✓ Une manière simple pour tester les performances sur d'autres types d'appareils.
    - ✓ Des assistants et des modèles pour les éléments communs trouvés sur tous les programmateurs Android.
    - ✓ Un éditeur complet avec une panoplie d'outils pour accélérer le développement de votre application. [75]

- **Netbeans IDE** « l'environnent de développement intégré, libre, extensible et universel, qui constitue une plateforme permettant le développement des applications et prenant en charge de divers langages de programmation» on l'utilise pour le langage PHP, et est l'outil technique que nous avons utilisé pour implémenter notre application.

- **Xampp**

XAMPP est un ensemble de logiciels permettant de facilement créer une interface web interagissant avec une base de données SQL!

- X pour cross-plateforme (LAMPP pour Linux, WAMPP pour Windows,...)
- A pour Apache
- M pour MySQL
- P pour PHP
- P pour Perl

Il possède également PhpMyAdmin pour gérer plus facilement vos bases de données. [26]

- **PhpMyAdmin**

PhpMyAdmin est un outil logiciel gratuit écrit en PHP, destiné à gérer l'administration de MySQL sur le Web. Prend en charge une large gamme d'opérations sur MySQL tel que la gestion des bases de données, des tableaux, des colonnes, des relations, des index, des utilisateurs, des autorisations, etc. ses opérations peuvent être effectuées via l'interface utilisateur, alors il offre aussi la possibilité d'exécuter directement une instruction SQL. [76]

- **Apache**

Le logiciel libre Apache HTTP est un serveur HTTP créé et maintenu au sein de la fondation Apache. C'est le serveur HTTP le plus populaire du World Wide Web. Il est distribué selon les termes de la licence Apache.

4.3 Langages de développement

- **JAVA**

C'est un langage de programmation orienté objet, développé par Sun Microsystems. Il permet de créer des logiciels compatibles avec de nombreux systèmes d'exploitation (Windows, Linux, Macintosh, Solaris). Java donne aussi la possibilité de développer des programmes pour téléphones portables et assistants personnels. Enfin, ce langage peut être utilisé sur internet pour des petites applications intégrées à la page web (applet) ou encore comme langage serveur (JSP). [77]

- **PHP**

PHP (Hypertext Preprocessor) est un langage de scripts généraliste et Open Source, spécialement conçu pour le développement d'applications web. Il peut être intégré facilement au HTML. [78]

- **SQL**

Le SQL (Structured Query Language) est un langage permettant de communiquer avec une base de données. Ce langage informatique est notamment très utilisé par les développeurs web pour communiquer avec les données d'un site web. SQL.sh recense des cours de SQL et des explications sur les principales commandes pour lire, insérer, modifier et supprimer des données dans une base. [79]

- **XML**

Le XML (eXtensible Markup Language) est un langage informatique qui sert à enregistrer des données textuelles. Ce langage a été standardisé par le W3C en février 1998 et est maintenant très populaire. Ce langage, grosso-modo similaire à l'HTML de par son système de balisage, permet de faciliter l'échange d'information sur l'internet.

Contrairement à l'HTML qui présente un nombre finit de balises, le XML donne la possibilité de créer de nouvelles balises à volonté. [31]

- **JSON**

Le format de données JSON (JavaScript Object Notation) constitue le standard actuel pour les échanges de données sur le Web. Il s'agit d'une syntaxe pour décrire des informations structurées sous une forme proche des objets JavaScript.

- **Web service :**

Un **service web** est un protocole d'interface informatique de la famille des technologies web permettant la communication et l'échange de données entre applications et systèmes hétérogènes dans des environnements distribué. Il s'agit donc d'un ensemble de fonctionnalités exposées sur internet ou sur un intranet, par et pour des applications ou machines, sans intervention humaine, de manière synchrone ou asynchrone. Le protocole de communication est défini dans le cadre de la norme SOAP dans la signature du service exposé (WSDL).Actuellement, le protocole de transport est essentiellement http. [81]

4.4 Etude de cas : application mobile self-scanning

Afin d'expliquer notre travail, nous présentons un scénario de composition pour une application supermarché. Cette application reflète la composition d'un ensemble de fonctionnalités à la fois assez simple dans leur fonctionnement et assez convenable pour illustrer nos apports.

- **Avantage de self-scanning :**

Notre application mobile est à l'intérêt des clients dans un supermarché. Celle-ci fait plaisir au client pour ne pas être déçu devant le caissier pour le règlement de payement. En permettant à un client scanner le code-barres de ces articles choisis. Il peut en temps réel savoir toutes les informations de ces achats comme le nom de produit, le prix, et la date d'expiration. Aussi en un coup d'œil le client peut contrôler le montant total de ses achats avant de passer au caissier en évitant un manque d'une somme d'argent et de ne pas rendre certains produit aux rayonnage.

- **Scénario de composition**

Un utilisateur peut utiliser notre application a pour objectif de calculer le montant des produits achetés dans un supermarché dont les séquences des activités principales sont:

- Fun1 : Acquisition de la photo du produit à partir de la caméra du dispositif mobile (composant: AquirePhoto).
- Fun2 : Scanner le code-barres du produit (composant: Barcode Reader).
- Fun3 : Traduire le code-barres et obtenir les informations du produit: nom, prix, date d'expiration ... (composant : get-product-id).
- Fun4 : Calculer à chaque fois le prix total des produits achetés (composant : Purchasing Price).

.

Nous allons adopter ce scénario comme une petite étude pour prouver l'efficacité de l'approche d'adaptation proposée. Dans ce scénario nous avons une séquence des fonctionnalités de différents types donc on vise face à une coordination hétérogène.

Notre travail destine à implémenter ce scénario de composition en attachant l'approche d'adaptation pour éliminer le défi d'hétérogénéité des entités logicielles à composer, en se basant sur un modèle architecturale détaillé de cette approche.

- ***DESCRIPTION :***

- L'entité logicielle **Aquire Photo** est de type composant permet de capturer une photo de produit et fournis en sortis une photo de type Bitmap.
- L'entité logicielle **Read Barcode** et de type composant, cette entité permet de scanner le code barre de produit et qui est transformé par la suite en un identifiant ID. la communication entre deux entités logicielles de même type et de même nature. Donc ce type de composition est homogène endogène nécessite aucun médiateur.

L'entité logicielle **get-product-by-id** et de type service, reçoit une donnée fournis par l'entité composant **Read-Barcode** (ID de type chaine de caractère) c'est une donnée non compatible par rapport à une donnée de son service requis (entité cible). Donc ce type de composition est hétérogène exogène.

Un médiateur exogène qui vise à fournir des interfaces bien formées et compatibles pour pouvoir assurer la communication entre les entités exogènes sera remplacé par une nouvelle entité. Cette entité a été encapsulé par une nouvelle entité s'appelle **component of service** est considéré elle-même comme une entité concrète de type composant. Les services requis et fournis de l'entité service **get-product-by-id** encapsulé seront transformés respectivement en un port requis et un autre fournis pour l'entité component of service. La donnée d'entrée qui est représentée par le port requis sera transférée via une requête pour déclencher le service approprié. En conséquence, une réponse pour cette requête sera récupérée et transmise à un port fournis de la nouvelle entité.

Un médiateur endogène qui vise à assurer la compatibilité des données échangées sera remplacé par un connecteur de médiation. Ce connecteur possède un rôle requis et un autre fournis. Le rôle requis est dédié pour supporter la donnée à transformer qui est fournie par l'entité source tandis que le rôle fournis vise à extraire la donnée transformée via ce connecteur de médiation dont l'objectif est de transférer le résultat obtenu à l'entité cible. Comme dans le cas suivant :

Le connecteur de médiation doit connaître le type de donnée à transformer ainsi que le type requis pour pouvoir appeler le service de médiation approprié. Un connecteur de médiation sert à comparer ces types des données afin d'exécuter dans le cas de la non compatibilité deux méthodes nommée respectivement *SearchMD* et *Call MD*. La première méthode sert à chercher le service de médiation nécessaire pour assurer la compatibilité des données et la deuxième à l'exécuter. Entité **get-product-id** permet d'obtenir les informations du produit (nom, prix, date d'expiration).

- L'entité logicielle **Purchasing Price** est de type service permet de calculer la somme des prix des produits scannés. Cette entité est encapsulée dans la classe component of service, reçoit une donnée (prix de type entier) qui a été fournis par **get-product-id** pour faire la somme des prix. Celle-ci fournit par la classe component of service.

Le tableau 4.1 montre la syntaxe graphique proposée pour représenter ces éléments architecturaux afin de pouvoir représenter graphiquement le modèle architectural détaillé de la CMA souhaitée. Nous avons utilisé **EdrawMax** pour obtenir notre architecture suivant les règles d'adaptation proposées.

EdrawMaxest le fournisseur de services le plus croyable d'un logiciel de création de diagramme graphique pour certains des marques les plus reconnaissables du monde d'entier

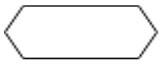
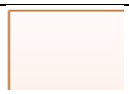
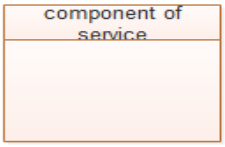
Nom de l'élément architectural	***Description de l'élément architectural***	***Forme graphique de l'élément architectural***
Entité service	Entité logicielle de type service	
Entité composant	Entité logicielle de type composant	
Component-of-services	Une nouvelle entité qui vise à encapsuler les entités de type service	
Requête Réponse	Sert à déclencher l'exécution d'un service tout en fournissant les données nécessaires • Diffuser la donnée obtenue au port fourni	
Type de la donnée d'entrée Type de la donnée de sortie	Représente le type de la donnée à transformer Représente le type vers lequel la donnée sera transformée	
Rôle requis Rôle fournis	Une interface qui supporte la donnée àtransformer • Une interface pour recevoir et garder la donnée transformée	

Tableau 4.1 – Les éléments architecturaux de l'architecture détaillée de la CMA

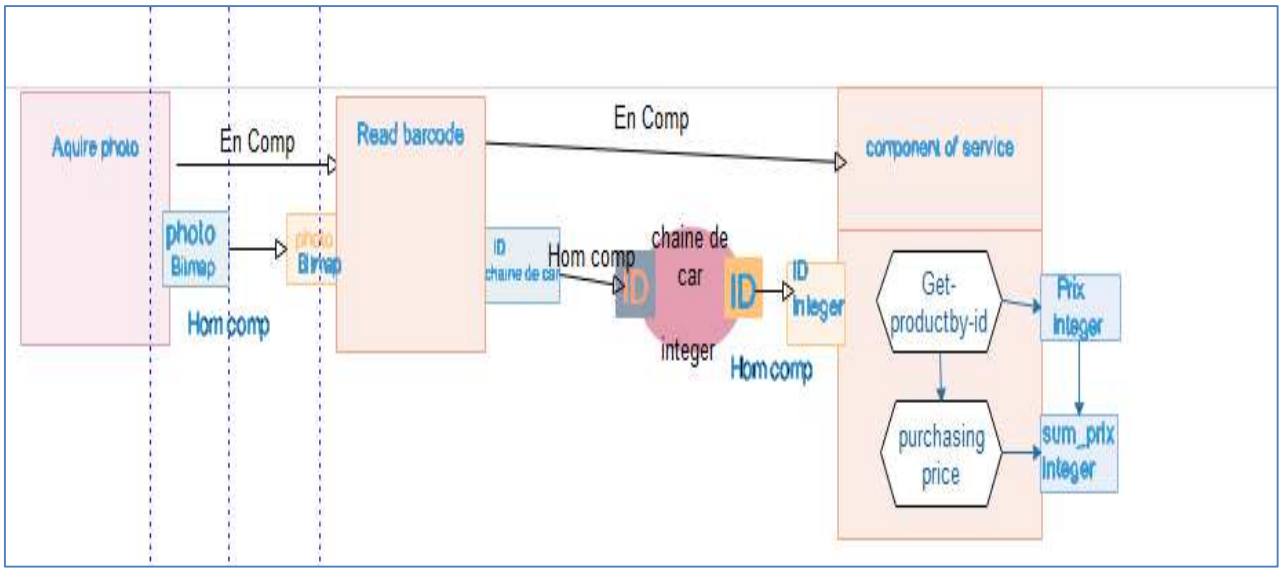
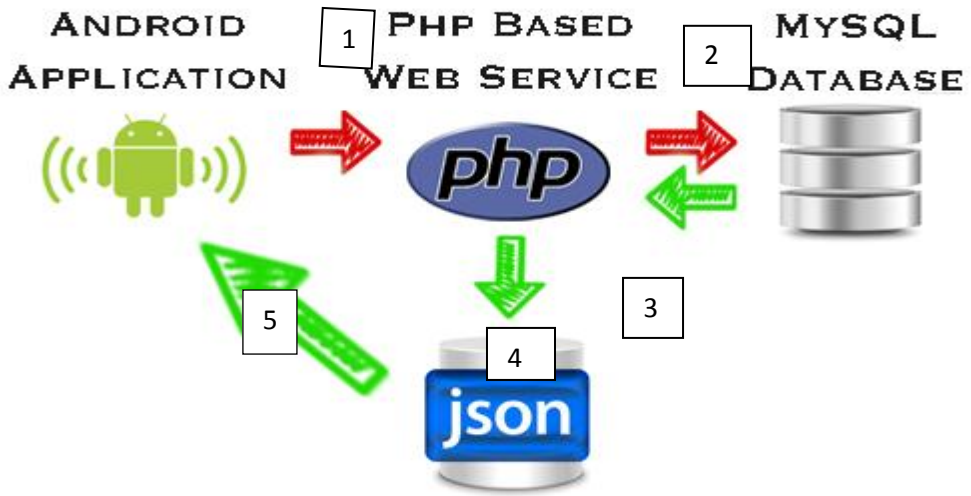


Figure 4.1 : Architecture détaillé de notre application mobile.

4.5 Implementation

4.5.1 Description du système

- **Accés au serveur :** il s'agit d'implémenter la partie logicielle qui va permettre de se connecter au serveur afin de communiquer avec la base de données



:

Figure 4.2 : architecture du système.

.

1. L'application envoie une requête HTTP au serveur avec l'adresse du script PHP dont elle a besoin. Le serveur se charge de trouver le script en question.

2. L'accès à la base de données se fait via les fichiers PHP (Web Services).

3. La base de données se charge d'insérer les données dans les tables (méthodes POST) ou bien de renvoyer le résultat d'une sélection (méthodes GET).

4. Le résultat retourné est en format *Java Script Object Notation* (JSON). Ce format permet de représenter de l'information structurée.

5. Le résultat est transféré à l'application. Il suffit ensuite de convertir le résultat pour ensuite le réutiliser (cette conversion est dite *parsing*).

4.5.2 Création de la base de données

L'interface « PhpMyAdmin » fournie avec Xampp permet de gérer la base de données :

La création de la base de données, la création des tables, la gestion des utilisateurs et leurs privilèges.

On a créé une base de donnés pour notre application qui contient deux tables.

- Produit (identifiant d'un produit, le nom, le prix et quantité).
- Utilisateur (identifiant, mot de passe, le nom, rôle)

4.5.3 Développement du système :

On a utilisé Netbeans avec le langage PHP pour implémenter des classes nécessaires suivantes :

4.5.3.1 La réutilisation de l'éxistent

- **Classe DAO :**

On a implémenté une classe DAO qui contient les requêtes SQL prédéfini ça veut dire le développeur peut appeler ces requêtes .S'il a besoin lors de développement de ces entités logicielles. Cette classe permet de connecter à une base de données.

- **Classe SALIHA DAO :**

La classe **SALIHADAO** est héritée de la classe DAO .Permet au développeur connecter à la base de donnée et utiliser les entités logicielles qui se trouvent dans la classe DAO.

- **Classe Product :**

Cette classe est héritée de la classe **SALIHADAO.** Cette classe contient deux paramètres : le nom de la table et identifiant du la table. Par exp un développeur veut créer une nouvelle table. Il accède à cette classe et il remplace dans les paramètres le nom de la table et identifiant par une nouvelle table, et il peut utiliser les requêtes de la classe DAO sur cette nouvelle table. L'intérêt de cette classe c'est pour aider le développeur à implémenter n'importe quelle application désirée sans la développer à zéro. Ça ce qu'on appelle **la** réutilisation de l'existent**.**

4.5.3.2 L'encapsulation des services :

- **Classe SALIHAAction :** c'est un composant contient les services qu'on a implémenté. Parmices services est le service **get-product-id**. Cette entité permet de récupérer les informations du produit .Classe SALIHAAction est-elle même une classe **component of service** laquelle encapsule ce service. Deuxième service est pour calculer la somme des prix des produits scannés par le composant scan code-barres, ce service est aussi encapsulé dans cette classe.

4.5.3.3 Les services de médiation *:*

- **Classe transférer donné :** c'est une classe qui contient des différents services de médiation ou bien les médiateurs endogènes qui permettent de transférer les données échangées qui sont non compatibles.

  On peut ajouter n'importe quel service de médiation nécessaire à cette classe qu'on le besoin pour régler les problèmes d'incompatibilités des données, avec un accès à cette classe et en faisant juste un appel de ce médiateur approprié .L'intérêt de cette classe assure la comptabilité des données échangées, ça veut dire on a réglé le problème d'hétérogénéité des données échangées entre les entités logicielles.

4.5.3.4 Création d'une bibliothèque :

On a créé une bibliothèque qui regroupe toutes les classes précédentes dans le cas où une classe a besoin d'une fonctionnalité qui existe dans une autre classe, il suffit seulement d'appeler cette bibliothèque pour accéder à une autre classe.

```php
<?php

include_once '../common/constant.php';
include_once '../common/transfer.php';
include_once '../DAO/DAO.php';
include_once '../DAO/DAOSALIHA.php';
include_once '../logique/SALIHAACTION.php';
include_once '../DAO/product/product.php';
```

4.5.3.5 Web service : on a implémenté deux web services :

- WS-Insert-Produit : c'est un web service pour insérer un nouveau produit dans la base de données à l'intérêt d'administrateur.
- WS-Get-Product : pour récupérer au client les informations des produits scannés

Le web service WS-Get-Product : c'est un web service qui sert à récupérer les informations d'un produit scanné, il reçoit un identifiant ID de type string ce type est incompatible par rapport à une donnée d'entré de service getProductById.

Donc le web service accède à la classe transfererdonné pour appeler par la suite à un médiateur endogène approprié qui sert à transférer un ID de type string à un ID de type int. Après il fait un appel à un service GetProductById qui est encapsulé dans un component of service SALIHAAction Ce web service accède aussi à la classe SALIHAAction pour appeler au service calculer-prix pour faire la somme des produits scannés.

Le web service récupère les résultats à l'application Android sous un format fichier JSON.

❖ **Scanner code-barres :**

Cette implémentation est faite sous un Android Studio pour implémenter le composant scan code-barres. On a utilisé un Zxing qui est une librairie Android, qui permet à notre application de reconnaitre le code-barres de produit.

Présentation des interfaces

❖ **Authentification d'un utilisateur :**

Une connexion via l'insertion du login et mot de passe dans les champs requis voir figure

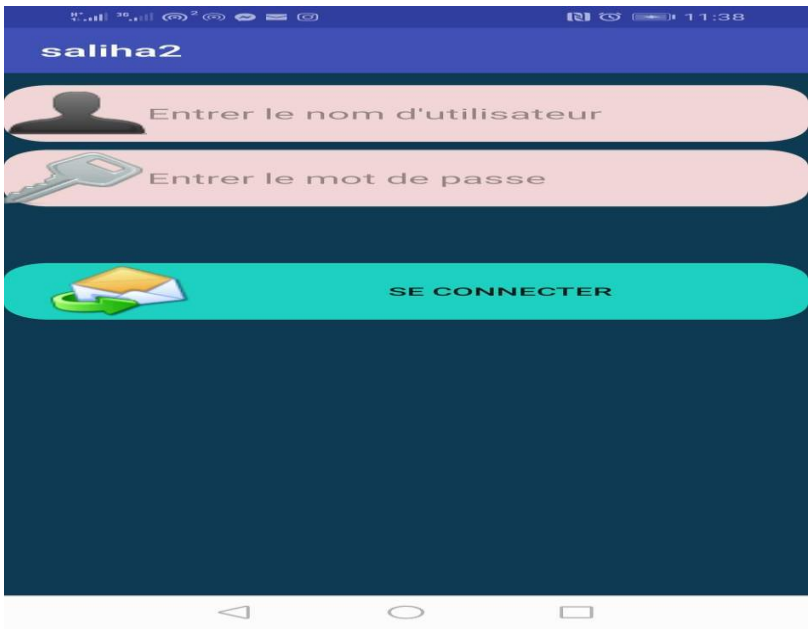


Figure 4.3 : interface d'authentification d'un utilisateur.

Après l'authentification, le client scanne le code barre d'un produit choisi et saisit le montant destiné pour les achats.

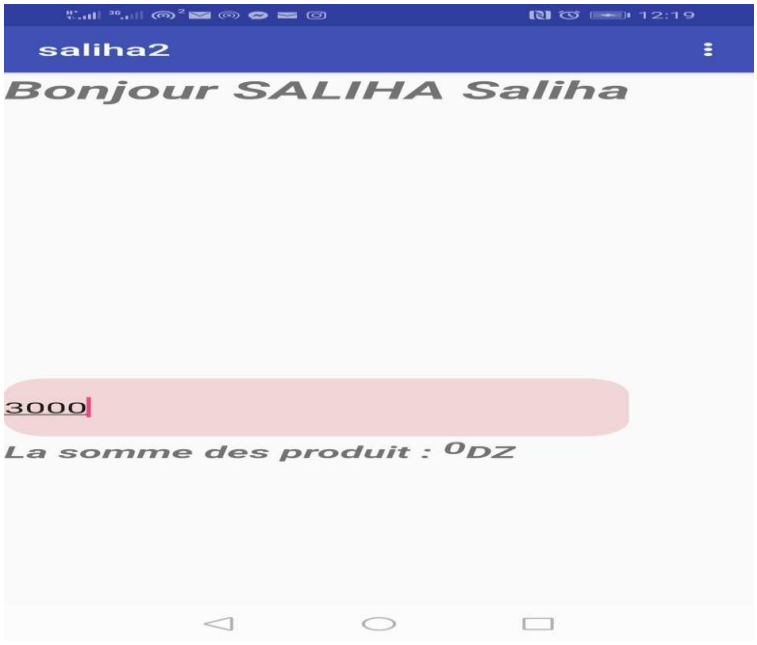


Figure 4.4 : interface d'insertion de montant.

❖ **Affichage les informations d'un produit**

Après le client scanne le code barre de son produit, il obtient l'interface contenant un affichage des informations de produit scanné comme illustré dans la figure suivante :

Figure 4.5 : **Affichage les informations d'un produit.**

4.6 Conclusion

Dans ce dernier chapitre on a présenté les outils de développement et les Langages utilisés pour implémenter notre application mobile qui est basée sur un modèle architectural détaillé via la réutilisation de l'existent. Cependant, nous avons proposé un scénario de composition comme une petite étude pour prouver l'efficacité du processus d'adaptation proposé. La phase d'implémentation est l'étape la plus importante de notre travail. Dans ce chapitre, nous avons décrit brièvement le processus de réalisation de notre application. En effet, nous avons achevé l'implémentation tout en respectant la conception élaborée.

Conclusion générale

Conclusion générale

Vu que la tâche de composition d'une application mobile est la satisfaction des exigences des utilisateurs, par conséquent, durant cette tache le développeur vise à faire face au défi d'hétérogénéité qui porte sur différentes entités logicielles à composer et d'autre porte sur les données échangées entre elle. Face à ce constat, ce projet de fin d'étude avait pour ambition de réaliser les médiateurs d'adaptation nécessaires afin d'éliminer cette hétérogénéité et donc permettre une composition consistante. Faire face à la coordination hétérogène :

- nécessite l'intégration des médiateurs exogènes en fournissant la structure concrète d'une nouvelle entité Component-of-services
- les médiateurs endogènes en fournissant la structure concrète au service de médiation proposé.
- Ce dernier nécessite de construire une bibliothèque de services de médiation selon les transformations requises des données échangées afin d'appeler par la suite celui qui est nécessaire pour assurer la compatibilité des données transmises.

Dans le contexte de notre travail l'approche d'adaptation utilisée a été déjà élaborée au niveau architectural pour les différents médiateurs proposés. Pour cela nous avons adopté un exemple comme une petite étude pour prouver l'efficacité de processus d'adaptation proposé. Cependant nous avons proposé un scénario de composition. Ce scénario est considérer comme une séquence de fonctionnalité de différents types d'implémentation (service, composant). Dans notre travail on a assuré des points suivants :

- ✓ Le déplacement des données entre la source et la cible.
- ✓ Le mappage des données entre différents formats.
- ✓ L'exécution séquentielle et conditionnelle de l'enchaînement des opérations. Perspectives :

Nous vison d'implémenter un algorithme à l'intérêt des développeurs qui permet de composer n'importe quelle application mobile en se basant sur un modèle architectural détaillé, cet algorithme permet d'Appeler des entités logicielles existantes nécessaires, Tout en appliquant des règles de coordination hétérogènes proposés.

Bibliographie

[1] BECHAR, SADELLI. Conception et Implémentation d'une Application Mobile pour les Services d'Aide aux Etudiants sous Android, Mémoire de Master2 de l'Université A. Mira- Bejaïa, juin 2013.

[2] BOUDJADJI, CHEKLAT. Conception et réalisation d'une application mobile sensible au contexte pour un musée, Mémoire de Master2 de l'Université A. Mira, Bejaïa, juillet 2013.

[3]Date consultation 04/2019
https://www.pmtic.net/sites/default/files/filemanager/memos/pmtic_env_num_systexpl_mobilesmartphones.pdf

[4] Clark, F. History of mobile applications. Date de consultation 04 2019. http://www.uky.edu/~jclark/mas490apps/History%20of%20Mobile%20Apps.pdf.

[5] Application embarquée ou service mobile. Date de consultation : 03 2019. http://www.mobileenfrance.com/2009/08/18/application-embarquee-ou-service-mobile-%E2%80%93-partie-1-de_nitions/.

[6] The Birth of Mobile Applications. Date de consultation 03 2019. https://skylineapps.wordpress.com/2012/01/16/the-birth-of-the-mobile-application

[7]Telli, Benjeddou.Une Application mobile : Une télécommande Bluetooth d'une souris optique, Mémoire de Master2 de l'université Kasdi Marbah Ouargla, septembte2013.

[8] Date de consultation 03/2019
http://www.leparisien.fr/guide-shopping/les-meilleures-appli-mobiles-en-2019-18-10-2019-8175429.php

[9] Date de consultation 03/2019
http://www.maisondugsm.com/a/encyclopedie/definition/4/systeme_d_exploitation.html

[10] Date de consultation. http://fr.wikipedia.org/wiki/Android.

[11] MIRAOUI, Moeiz. *Architecture logicielle pour l'informatique diffuse : modélisation du contexte et adaptation dynamique des services.* Thèse de doctorat, école technologie supérieure université du Québec, JUILLET 2009.
[12] SEBBAK, F. Architectur intergicielle semantique basé sur le standard OSGI pour les services robotiques, Mémoire de Magistère en Informatique, Université Abderrahmane Mirade Bejaïa, 2007.

[13] B-N.SHILIT et M.THEIMER. *Disseminating Active Map Information to Mobile Hosts*.
Network, IEEE, Vol.8, N°5, pp. 22-32, September/October 1994.
[14] B.SCHILIT, N.ADAMS et R.WANT.*Contextaware computing application.* Dans In Proceedings of the Workshop on Mobile Computing Systems and Applications: pages 8590, IEEE: Computer Society.1994.
[15] J.BROWN, D.BOVEY et X.CHEN. *Context-aware applications: from the laboratory to theMarketplace*, IEEE Personal Communications, vol. 4, no 5, p. 58-64, (October 1997)
[16] N.RYAN, J.PASCOE et D.MORSE *Enhanced Reality Fieldwork: The Contextaware Archaeological Assistant*, In Computer Applications in Archaeology, Oxford, UK.1997
[17] G.CHEN, D.KORT.*A Survey of Context-Aware Mobile Computing Research*, TR2000- 381, Dartmouth: Dartmouth College Computer Science, 2000.
[18] A-K.DEY. *Understanding and using context.* Personal and ubiquitous computing, vol. 5, p. 4-7.2001.
[19] T.CHAARI. *Adaptation d'applications pervasives dans des environnements multicontextes*,
Thèse de doctorat. L'institut national des sciences appliquées de Lyon. 28/09/2007.
[20] B.SCHILIT, N.ADAMS, N et R.WANT. *Context-Aware Computing Applications*, In- Proc. of the Workshop on Mobile Computing Systems and Applications (Santa Cruz, CA, Dec.1994), pp. 85-90.
[21] P-J BROWS, 1995. *The stick-e document: a framework for creating context-aware applications*. Electronic Publishing Origination, Dissemination and Design, vol. 8, no 2 (JUNE AND SEPTEMBER 1995), p. 259-272.

[22] Dey, A. K. et al. (2001)."A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications." Human-computer interaction, L. Erlbaum Associates Inc. 16(2), p 97-166. ISSN: 0737-0024.
[23] Brown, P. J. (1998)."Triggering information by context." Personal Technologies, Springer 2(1) :18-27. ISSN: 0949-2054.

[24] Pascoe, J. (1998).Adding generic contextual capabilities to wearable computers. Wearable Computers,1998. Digest of Papers.Second International Symposium on, IEEE. p. 92-99. ISBN: 0818690747.

[25] Pascoe, J. et al. (1998).Human-Computer-Giraffe Interaction-HCI in the Field. InWorkshop on Human Computer Interaction with Mobile Devices.

[26] Brown, P. J. et al. (1997). "Context-aware applications : from the laboratory to the marketplace." Personal Communications, IEEE 4(5) : 58-64. ISSN : 1070-9916.

[27] Cheverst, K. et al. (2000). Providing tailored (context-aware) information to city visitors. InAdaptive hypermedia and adaptive web-based systems, Springer. p. 73-85.ISBN : 3540679103.

[28] [67] Djeddar, A.Contribution développement d'un Processus de Composition d'Applications Mobiles Adaptatives à base d'Entités Logicielles Hétérogènes. Thèse de doctorat en informatique de l'Université de Larbi Tébessi de TÉBESSA, 20 mai 2016.

[29] Djeddar, A. et al. (2015). "Developing Context-aware Mobile Applications Using Composition Process based-on heterogeneous software entities." Journal of Advanced Computer Science and Technology Research 5(3): 93-103.

[30] Nickul, D. et al. (2007). "Service oriented architecture (SOA)and specialized messaging patterns." A technical White Paper published by Adobe Corporation USA.(En ligne). Disponible sur : <http://www.adobe.com/enterprise/pdfs/Services_Oriented_Architecture_from_Adobe.pdf>

[31] Manfred, B.et al.What characterizes a (software) component? Software - Concepts and Tools, 19(1):49–56, 1998.

[32] Allen, R. J. et al. (1998). Formal modeling and analysis of the HLA component integration standard .In ACM SIGSOFT Software Engineering Notes, ACM .Vol. 23, No. 6, p. 70-79. ISBN: 1581131089.

[33] Garlan, D. and D. E. Perry (1995). "Introduction to the special issue on softwarearchitecture." IEEE Trans Software Eng. Citeseer 21(4): 269-274.

[34] Dashofy, E. M .et A. Van Der Hoek (2002). Representing product family architectures
in an extensible architecture description language. In: Frank J. van der Linden, Software Product-Family Engineering, Springer: p. 330-341. ISBN: 978-3-540-21941-5.

[35] D. C. Luckham, J. Kenney, L. M. Augustin, J. Vera, D. Bryan and W. Mann. Specification and Analysis of System Architecture Using Rapide. IEEE Transactions on Software Engineering, pages 336-355, April 1995.

[36] Amirat, A .et Oussalah, M. “Towards an UML Profile for the Description of Software Architecture” Proceedings of International Conference on Applied Informatics (ICAI’09), pp. 226 – 232.

[37]Clemens Szyperski. *Component Software: Beyond Object-Oriented Programming*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[38] OMG. *Unified Modeling Language Specification 2.0: Superstructure*,
2005. OMG doc. Formal/05-07-04.

[39] Bruneton, E. et al. (2006)."Thefractal component model and its support in java.” Software-Practice and Experience, Wiley Interscience .London, New York 36(11): 1257-1284. ISSN: 0038-0644.

[40] Object Management Group .Corba component model 4.0 specification.
Specification Version 4.0, Object Management Group, April 2006.

[41] Carlos Hernán, P. Rojas.Une approche à base de composants logiciels pour l'observation de systèmes embarqués .Thèse de doctorat en informatique, Université de GRENOBLE, Juin 2011. P42.

[42]ERL, T. *Service-oriented architecture*, Prentice Hall New York2005.

[43] Krafzig,D., Banke, K .et Slama, D (2004). *Enterprise SOA: Service-Oriented Architecture Best Practices* Prentice Hall.

[44] ZHAO, L., REN, Y., LI, M. and SAKURAI, K. Flexible service selection with user-specific QoS support in service-oriented architecture. *Journal of Network and Computer Applications* 35, 3 (2012), 962-973.

[45] Christensen, E. et al. (2001). Web services description language (WSDL) 1.1. (En ligne).
Disponible sur : Http:// www.w3.org/TR/wsdl.

[46] Hadley, M. et al. (2003). "SOAP Version 1.2 Part 1 : Messaging Framework."
W3C REC REC-soap12-part1-20030624. (En ligne) June: 240-8491. Disponible sur :
http://www.w3.org/TR/soap12-part1.7

[47] Brel, C. (2013). Composition d'applications multi-modèles dirigée par la composition desinterfaces graphiques. Thèse de doctorat en informatique. Antipolis : Université Nice Sophia,
28 juin 2013. 204 p.

[48] Alonso, G. et al. (2004). Web services: Concepts, Architectures and Applications, Springer Verlag. p.320.ISBN: 3642078885.

[49] Benatallah, B., et al. (2005). Service Composition : Concepts, Techniques. In: Z. Stojanovicand A. Dahanayake, Service-Oriented Software System Engineering: Challenges and Practices, Idea Group. p. 48-66. ISBN: 1-59140-428-2.

[50] Srivastava, B., Koehler, J. Web Service Composition – Current Solutions and Open Problems. In:Procs of the 13th International Conference on Automated Planning and Scheduling (ICAPS 2003),Workshop on Planning for Web Services, June 2003, Trento, Italy.

[51] Yang, J., Papazoglou, M.P. Service Component for Managing Service Composition Life-Cycle .Information Systems, vol.29, n°2, pp.97-125, 2004.

[52]Casati, F. et Shan, M.-C (2001). Dynamic and adaptive composition of e-services, *Information Systems*,26(3), pp. 143-163.

[53]Khalaf, R. et al. (2003).Service-Oriented Composition in BPEL4WS. In: Proceeding of the12th International World Wide Web Conference (WWW), Alternate Paper Tracks, Budapest, Hungary. p. 27-28.

[54] Djeddar, A. et al. (2015). "Developing Context-aware Mobile Applications Using Composition Process based-on heterogeneous software entities." Journal of Advanced Computer Science and Technology Research 5(3): 93-103.

[55] Kalasapur, S. et al. (2007). "Dynamic service composition pervasive computing" Parallel and Distributed Systems, IEEE Transactions on 18(7): 907-918. ISSN: 1045-9219.

[56] Verma, K. et al. (2005). The METEOR-S approach for configuring and executing dynamic web processes. Technical Report TR6-24-05, The University of Georgia, Computer Science Department, LSDIS Lab, Athens, Georgia.

[57]Li, L. et al. (2009). Trust-oriented composite service selection and discovery. In: L. Baresi, C.Chi and J. Suzuki, Service-Oriented Computing, Springer Berlin Heidelberg: p. 50-67. ISBN: 978-3-642-10382-7.

[58] Suraci, V. et al. (2007).Context-aware semantic service discovery.In Mobile and Wireless Communications Summit, 2007. 16th IST, 1-5 July, IEEE. p. 1-5.
ISBN: 9638111666.

[59] BENTLEMSAN, K. "UNE APPROCHE ARCHITECTURE LOGICIELLE POUR LA COMPOSITION DE SERVICES WEB". Mémoire de Magister en informatique de l'Université Saad Dahleb de Blida, Juin 2010.

[60] Prochart, G. et al. (2007).Fuzzy-based support for service composition in mobile ad hocnetworks. In Pervasive Services, IEEE International Conference on,IEEE. p. 379-384. ISBN: 1424413257.

[61] Rosa, R. E. V. et V. F. Lucena Jr (2011).Smart composition of reusable software components in mobile application product lines., In Proceedings of the 2nd International Workshop on Product Line Approaches in Software Engineering ACM .p. 45-49.ISBN: 1450305849.

[62] Furno, A .et Zimeo, E (2014). "Context-aware composition of semantic web services." Mobile Networks and Applications, Springer 19(2): 235-248. ISSN: 1383-469X.

[63]Sonia JAMAL, Environnement de procédé extensible pour l'orchestration Application aux services web, thèse de doctorat, UNIVERSITE JOSEPH FOURIER Grenoble (France), Décembre 2005.

[64] Component Source. The evolution of components. Document en ligne,
05 2019. http://www.componentsource.com/.

[65] Meijler,T .et Nierstrasz,O. Beyond objects: Components. Dans *Cooperative Information Systems: Current Trends and Directions*. Academic Press,1997.

[66] Derdour, M. et al. (2010a). "Typing of adaptation connectors in MMSA approach case study: sending MMS." International Journal of Research and Reviews in Computer Science 1(4):39-49. ISSN: 2079-2557.

[67] Derdour, M.et al.(2010)."MMSA: metamodel multimedia software architecture." Advances in Multimedia. Hindawi Publishing Corporation, Article ID 386035.17 pages, 2010. Doi: 10.1155/2010/386035.

[68] Bass, L.et al. Volume i: Market assessment of component-based softwareengineering. Rapport technique CMU/SEI-2001-TN-007, Software Engineering Institute, Mai 2000.

[69] Wuyts, R .et Ducasse,S. Composition languages for black-box components. Dans *First OOPSLA Workshop on Language Mechanisms for Programming Software Components*, 2001.

[70] Beisiegel, M. et al. (2005). Service Component Architecture: Building Systems Usinga Service Oriented Architecture. (En ligne) Whitepaper. Vol. 1, 31 p. Disponible sur : http://www.iona.com/devcenter/sca/SCAWhitePaper109.pdf.

[71] Hourdin, V. et al. (2008).SLCA, composite services for ubiquitous computing. In Mobility'08,Proceedings of the International Conference on Mobile Technology, Applicationsand Systems, ACM, Singapore. p. 1-8. ISBN: 1605580899.

[72] Hock-Koon, A. and M. Oussalah (2010). "Composite service metamodel and auto composition."Journal of Computational Methods in Sciences and Engineering, IOS Press 10(1-2S2): 215-229. ISSN: 1472-7978.

[73] Chakraborty, D. et al. (2005). "Service composition for mobile environments." Mobile Networks and Applications, Springer-Verlag, New York, Inc. 10(4): 435-451.ISSN: 1383-469X.

[74] Wu, Z. et al. (2007). Automatic composition of semantic web services using process and data mediation. Technical Report, LSDIS lab, University of Georgia, February 28.Disponible sur: http://corescholar.libraries.wright.edu/knoesis/8.

[75] “https: //android-studio.fr.uptodown.com/Windows.” consulté le 06 Juin 2019.

[76] “https://www.phpmyadmin.net/.” consulté le 06 Juin 2019

[77] “http://www.futura-sciences.com/tech/definitions/internet-java-485/.” consulté le 08 Juin 2019.

[78] “http://php.net/manual/fr/intro-whatis.php.” consulté le 08 Juin 2019.

[79] “http://sql.sh/.” consulté le 08 Juin 2019.

[80] “http://glossaire.infowebmaster.fr/xml/.” consulté le 08 Juin 2019.

[81] https://fr.wikipedia.org › wiki › Service web consulté le 08 juin 2019.