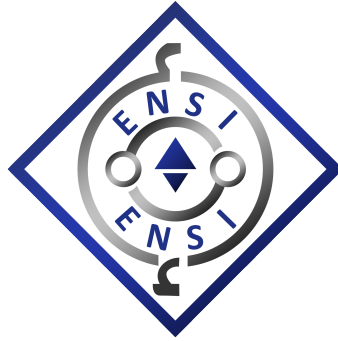


Ministère de l'Enseignement Supérieur,  
de la Recherche Scientifique  
Université de la Manouba  
Ecole Nationale Des Sciences De L'Informatique



---

**Rapport de Projet Conception et Développement**

SUJET :

***Applications Web, Mobile d'intermédiaire en bourse***

---

Réalisé par :

**Yassine MAALEJ**

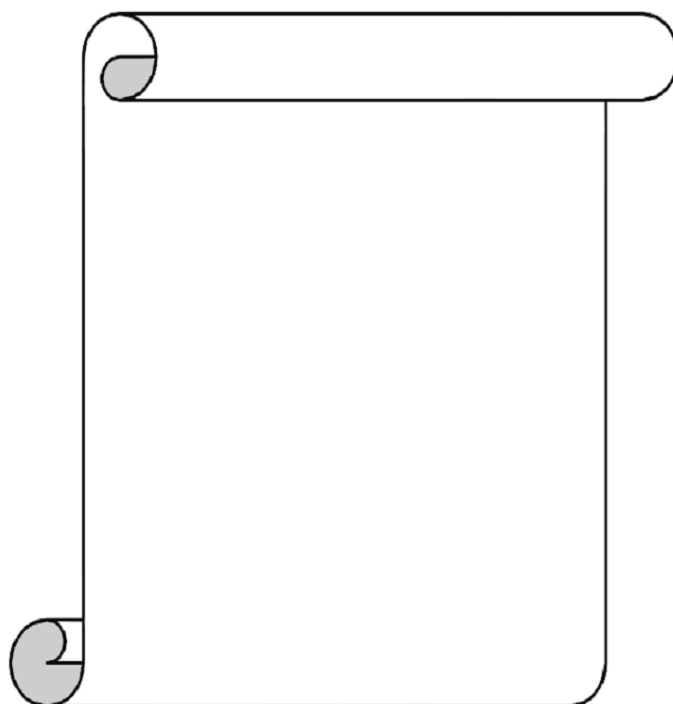
Sous l'encadrement de :

**Mme Raoudha KHCHERIF**

**Année Universitaire : 2012-2013**

# Appréciations de l'encadrant

Mme.Raoudha KHCHERIF



# Remerciements :

*Nous tenons à exprimer notre grande reconnaissance envers les personnes qui nous ont, de près ou de loin, apporté leur soutien. Qu'ils trouvent ici collectivement et individuellement l'expression de toute notre gratitude.*

*Nous adressons nos plus sincères remerciements à notre encadrante Mme. KHCHERIF Raoudha dont la courtoisie et la disponibilité ont été constantes à notre égard, pour ses conseils précieux et la qualité de l'encadrement dont elle nous a fait bénéficié tout le long de ce projet.*

*Que les membres du jury trouvent ici, l'expression de nos remerciements pour l'honneur qu'ils nous font en acceptant de juger ce travail.*

*Enfin, nous ne saurions laisser passer cette occasion sans saluer chaleureusement tous nos collègues à l'ENSI pour tout leur soutien moral dont ils nous ont fait bénéficier.*

# Table des matières

<b>Introduction Générale</b>	<b>1</b>
<b>1 Etude préalable :</b>	<b>3</b>
1.1 Etude de l'existant . . . . .	3
1.2 Problèmes dégagés . . . . .	4
1.3 Solutions promises . . . . .	7
<b>2 Analyse et Spécification</b>	<b>8</b>
2.1 Analyse des besoins . . . . .	8
2.1.1 Besoins fonctionnels . . . . .	8
2.1.1.1 Besoins du client . . . . .	9
2.1.1.2 Besoins de l'administrateur . . . . .	9
2.1.2 Besoins non fonctionnels . . . . .	9
2.2 Spécification des besoins . . . . .	10
2.2.1 Diagrammes de spécification cas d'utilisation . . . . .	10
2.2.1.1 Identification des acteurs . . . . .	10
2.2.1.2 Cas d'utilisation d'un administrateur . . . . .	11
2.2.1.3 Cas d'utilisation d'un client . . . . .	11
2.2.2 Scénario et diagrammes de séquence . . . . .	12
2.2.2.1 Partie Administrateur . . . . .	13
2.2.2.2 Partie Client . . . . .	16
<b>3 Conception</b>	<b>19</b>
3.1 Conception globale et architecture . . . . .	19
3.1.1 Architecture du site web . . . . .	19
3.1.1.1 Architecture générale MVC . . . . .	19

3.1.1.2	Java ServerFaces et MVC . . . . .	21
3.1.1.3	Serveur d'application . . . . .	22
3.1.2	Architecture de l'application Mobile . . . . .	23
3.1.2.1	REST . . . . .	24
3.1.2.2	JAX-RS . . . . .	25
3.2	Diagramme des packages du système . . . . .	26
3.3	Conception détaillée . . . . .	30
3.3.1	Conception détaillée côté site web . . . . .	30
3.3.2	Conception détaillée côté Android et web service . . . . .	31
3.3.3	Diagramme de la base des données . . . . .	32
3.4	Diagrammes d'activités . . . . .	33
3.4.1	Diagrammes d'activités de l'application Web . . . . .	33
3.4.2	Diagrammes d'activités de l'application Mobile . . . . .	34
<b>4</b>	<b>Réalisation</b>	<b>36</b>
4.1	Environnement de travail . . . . .	36
4.1.1	Environnement matériel . . . . .	36
4.1.2	Environnement logiciel . . . . .	37
4.2	Choix technologiques . . . . .	37
4.2.1	Choix technologiques communs . . . . .	38
4.2.1.1	Choix du langage Java . . . . .	38
4.2.1.2	Choix du serveur MYSQL . . . . .	38
4.2.2	Choix technologiques pour l'application Web . . . . .	39
4.2.2.1	Choix de J2EE . . . . .	39
4.2.2.2	Choix du framework pour les couches de présentation . . . . .	39
4.2.2.3	Choix de l'environnement de développement . . . . .	40
4.2.3	Choix technologiques pour l'application Mobile . . . . .	40
4.2.3.1	Choix de la plateforme Android . . . . .	40
4.2.3.2	Choix de l'environnement de développement . . . . .	41
4.2.3.3	Choix du JSON au lieu de XML . . . . .	41
4.3	Réalisation et présentation de l'application . . . . .	43
4.3.1	Présentation du site web . . . . .	43
4.3.2	Présentation de l'application mobile . . . . .	49

<b>Conclusion générale</b>	<b>54</b>
<b>Netographie</b>	<b>55</b>

# Table des figures

1.1	Résumé du marché offert par Amen Invest . . . . .	4
1.2	Résumé du marché offert par London Stock Exchange . . . . .	5
1.3	Exemple d'application Mobile de suivit de bourse . . . . .	6
2.1	Diagramme de cas d'utilisation concernant un Administrateur . . . . .	11
2.2	Cas d'utilisation pour le client . . . . .	12
2.3	Diagramme d'authentification de l'administrateur . . . . .	13
2.4	Diagramme d'ajout d'un nouveau utilisateur . . . . .	14
2.5	Diagrammes de modification des prix des actions . . . . .	15
2.6	Diagramme d'authentification du client . . . . .	16
2.7	Diagramme de Vente d'actions . . . . .	17
2.8	Diagramme d'achat d'actions . . . . .	18
3.1	Architecture MVC . . . . .	20
3.2	Architecture du framework JSF . . . . .	22
3.3	Cycle de vie d'une Activité sous Android . . . . .	23
3.4	Architecture REST . . . . .	24
3.5	Architecture REST . . . . .	25
3.6	Diagramme des packages du site web . . . . .	26
3.7	Diagramme des packages de l'application Android et du Web service . . . . .	28
3.8	Schéma de dépendances entre les packages du site web . . . . .	30
3.9	Schéma de dépendances entre les packages du l'application Android . . . . .	31
3.10	Base des données Bourse . . . . .	32
3.11	Diagramme d'activité du site web . . . . .	33
3.12	Diagramme d'activité de l'application Mobile . . . . .	34

4.1	Tableau des outils logiciels . . . . .	37
4.2	Architecture de la plateforme Android . . . . .	41
4.3	Structure de JSON . . . . .	42
4.4	Page d'accueil TuniBourse . . . . .	44
4.5	Espace Client après authentification . . . . .	45
4.6	Fenêtre de calcul de rentabilité . . . . .	46
4.7	Espace de l'action Telecom . . . . .	47
4.8	Mise à jour du prix . . . . .	48
4.9	Ajout d'un acteur . . . . .	49
4.10	Accueil . . . . .	50
4.11	Espace client . . . . .	51
4.12	Espace action Tunisiana . . . . .	52



# Introduction Générale

**L**E secteur boursier en Tunisie ne cesse de s'améliorer de plus en plus, mais reste comme même très déphasé en le comparant avec d'autres pays comme l'Angleterre, pays leader au monde avec le plus grands nombres d'intervenant et d'échange, car la bourse est considérée comme le moteur de l'économie numéro 1, c'est pourquoi la plupart des grandes entreprises ou celles qui veulent agrandir leur capital sont inscrites en bourse. Contrairement en Tunisie, on constate que les opérateurs de Télécommunication ne sont aucun d'eux, inscrit en bourse, malgré que ce type d'entreprise est le plus grand en terme d'investissement et de gain et « recherche à l'amélioration ».

Dans le cadre de Projet de Conception et de Développement PCD dans notre formation d'ingénieur en informatique en deuxième année à l'ENSI, notre travail consiste à la réalisation d'un site Web d'un intermédiaire en bourse avec une application Mobile .

Le but de ce projet est de développer un site Web en j2ee qui permet la gestion d'un portefeuille limité de «Télécommunication» côté administration et client et d'une application Mobile permettant aux clients le suivi de leurs portefeuilles.

Le présent rapport est organisé en quatre chapitres.

Le premier chapitre est consacré à l'étude de l'existant. En effet, nous allons d'une part étudier l'état de la bourse en Tunisie pour dégager ses différentes lacunes et d'autre part étudier l'état de la bourse à l'étranger pour s'inspirer des fonctionnalités qu'ont peut ajouter à notre projet. Dans un deuxième chapitre, nous présentons en premier lieu les besoins fonctionnels et non fonctionnels à respecter. Enfin, nous terminons par la présentation des diagrammes des cas d'utilisation et des diagrammes de séquences.

Le troisième chapitre est une étude conceptuelle dont la quelle nous présentons le modèle

conceptuel de notre application ainsi que la conception de notre base de données et des services offerts par notre projet.

Le dernier chapitre est consacré à la présentation des environnements matériel et logiciel ainsi que la réalisation de notre site Web et aussi notre application Mobile.

Ce rapport est clôturé par une conclusion rappelant notre problématique ainsi que les résultats obtenus et comment nous pouvons les améliorer.

# Chapitre 1

## *Etude préalable :*

### Introduction

Dans ce chapitre, nous allons présenter l'état de la bourse en Tunisie et en particulier les problèmes et les manques de services des intermédiaires en bourse puis nous présentons les objectifs de notre projet .

#### 1.1 Etude de l'existant

La Bourse en Tunisie a plus de quarante ans d'existence depuis sa création en 1969 en tant qu'établissement public. Au cours de cette période, la Bourse a connu des changements profonds qui ont fait d'elle un instrument de financement moderne et attractif, fonctionnant dans un cadre légal et technique adapté aux meilleures normes internationales, mais on ne peut jamais parler, dans notre cas, que la Bourse Tunisienne est entrain de jouer un rôle essentiel dans notre économie et ceci est perceptible à travers le nombre des personnes qui se sont inscrits en Bourse, à travers les intermédiaires, et surtout à travers les échanges quotidiennes et son activité en générale. De plus l'absence des opérateurs téléphoniques à la Bourse est aussi un indice énorme de faiblesse surtout qu'il est largement connu que ces types de sociétés demandent beaucoup de financement pour qu'ils puissent être à jour avec les nouvelles technologies de télécommunication et de l'informatique.

## 1.2 Problèmes dégagés

Il est clair que la procédure de participation en bourse est un peu compliquée et les services offerts par les intermédiaires en bourse ne sont pas très développés et n'incitent pas les gens à contribuer dans ce système. De plus on remarque une complexité au niveau de l'utilisation dans pratiquement tous les sites webs des intermédiaires de plus le chargement des courbes qu'ils ne sont compréhensibles que par de vrais experts. De même, la majorité des sociétés intégrées en bourse ne sont pas tellement connues comme les opérateurs de téléphonies : qui n'a pas un téléphone ? qui ne se connecte pas depuis son mobile ? qui n'a pas une bonne idée sur les nouveautés, les offres et les meilleurs services de ces opérateurs. Avoir la possibilité de consulter les prix instantanés des actions est aussi une fonctionnalité non encore existante pour se mettre à tous moments à l'abri des variations des prix sans réagir et intervenir .



FIGURE 1.1 – Résumé du marché offert par Amen Invest

La figure 1.1 nous présente l'intermédiaire en bourse AMEN INVEST qui est considéré comme le meilleur en Tunisie. Cependant, il présente le résumé du marché de façon générale et n'offre pas à ces clients des résumés de leurs transactions comme il n'a pas de service de calcul de rentabilité qui aident le client à bien choisir les futures transactions.



FIGURE 1.2 – Résumé du marché offert par London Stock Exchange

La figure 1.2 nous présente l'intermédiaire en bourse Anglais London Stock Exchange qui, au plus du résumé du marché, offre à ces clients le résumé de leurs choix et leurs gains avec l'aspect général du marché à savoir la variation du portefeuille du client vis à vis la variation des prix dans le marché.

De plus il n'ya aucun intéremédiaire en bourse qui offre une application Mobile de gestion de portefeuille pour ces clients en tunisie mais in en existe ailleurs.


Palmarès Euronext Gagnants (%)		Achat Cours Vente	% Var.
↑ <b>ACCB</b> Accentis	0.0100 <b>0.0200</b> 0.0200	+	100.0% 0.0100
↑ <b>ALPHX</b> Phenix Systems	6.37 <b>6.70</b> 6.70	+	55.8% 2.40
↑ <b>PUN</b> Punch International	5.18 <b>5.20</b> 5.22	+	36.8% 1.40
↑ <b>MLATP</b> Ati Petroleum	0.0500 <b>0.0400</b> 0.0600	+	33.3% 0.0100
↑ <b>TISN</b> Tiscali	0.0400 <b>0.0400</b> 0.0500	+	33.3% 0.0100
↑ <b>MLTRC</b> Trois Chenes	7.50 <b>7.50</b> 8.95	+	25.0% 1.50
↑ <b>ALDV</b> Adc Silc	0.1200 <b>0.1500</b> 0.1500	+	25.0% 0.0300
			
<b>Moniteur</b> Actualités   Messagerie   Forum   Configuration			

FIGURE 1.3 – Exemple d'application Mobile de suivit de bourse

Cette application Mobile, présenté par la figure 1.3 offre la consultation des prix de tous les titres figurants sur le marché avec ceux qui constituent le portefeuille du client en indiquant s'il sagit d'augmentation ou de diminution de prix dans une interface simple et claire.

Nous allons nous inspirer d'autres sites web d'intermédiaire en bourse pour avoir plus d'idées et de fonctionnalités qui pourrons nous être utiles pour nos applications Web et Mobile.

### **1.3 Solutions promises**

Notre application est principalement conçue pour faire face aux problèmes rencontrés par les particuliers pour s'introduire en Bourse depuis un site web ou sur Android. Elle permettra, en effet, de fournir toutes les données nécessaires pour le trading et le suivi des portefeuilles tout en garantissant un très bon niveau de sécurité à tous les clients d'une façon équitable dans le sens la rapidité de variation des prix et des actualités des sociétés en Bourse ainsi que sur l'aspect général de la réaction des clients face à n'importe quelle variation.

### **Conclusion**

**T**Out au long de ce chapitre, nous avons présenté une étude de l'existant dans le domaine boursier et nous avons tiré les problèmes essentiels de ce que existe actuellement. Comme il est précédemment décrit, notre projet (site web et application Mobile) est une nouvelle idée reposant sur un concept dynamique et développé et offrant de nouvelles fonctionnalités.

## Chapitre 2

# *Analyse et Spécification*

### Introduction

Notre objectif par le présent projet est de concevoir ensuite réaliser un site Web et une application Mobile d'un intermédiaire en bourse. Nous commencerons par une présentation des différents besoins fonctionnels et non fonctionnels relatifs au projet. Ensuite, nous procéderons à une spécification détaillée relayant les différents intervenants administrateurs et clients.

### 2.1 Analyse des besoins

Dans cette section, nous exposons d'une manière textuelle les services que le système est censé fournir. Nous commençons en premier lieu par définir les besoins fonctionnels. Nous passons ensuite à décrire les besoins non fonctionnels.

#### 2.1.1 Besoins fonctionnels

Dans cette partie, nous allons exposer l'ensemble des besoins fonctionnels auxquels devraient répondre impérativement notre Site Web et l'application Mobile. En effet, selon une vue ensembliste, le besoin fonctionnel partagé entre tous les clients est d'avoir une participation à la bourse plus claire, simple et beaucoup plus informative, et pour les administrateurs, de mieux mettre à jour et d'améliorer les applications.



### **2.1.1.1 Besoins du client**

Le site Web doit mettre à la disposition du client l'ensemble des actions qui constituent son portefeuille avec leurs proportions. Elle doit également lui permettre de savoir les prix actuels, les variations des actions et leur disponibilité sur le marché pour qu'il puisse réagir en achetant ou en vendant des actions et par conséquent ces transactions seront enregistrées. De plus, chaque client exige l'équité d'avoir l'information dans le sens de temps de réponse de l'application sans privilège attribué.

Quant à l'application Mobile, elle doit fournir rapidement au client les données nécessaires pour faire des achats ou des ventes rapides suite à une variation des prix des action.

### **2.1.1.2 Besoins de l'administrateur**

L'administrateur qui dispose des droits exclusifs, dans le site web, aux ajouts et aux contacts des clients, devrait être capable d'activer l'application pendant les périodes d'échanges, de la désactiver ailleurs et d'élargir le nombre des actions totales dans le portefeuille de l'intémédiaire pour que ses clients les partages en cas de nécessité.

## **2.1.2 Besoins non fonctionnels**

Les besoins non fonctionnels présentent les exigences internes suivantes :

- Contrainte ergonomique : une interface simple et conviviale pour que l'administrateur ait une manipulation aisée et rapide. Le client doit jouir d'un espace sécurisé en lui offrant des courbes dynamiques de variation des prix.
- Invoquer une certaine souplesse au niveau du traitement interne de l'application contre les problèmes intolérables aux erreurs de pannes de serveur ou d'impossibilité de rendre service aux clients.
- Contrainte sur la fiabilité du site web et de l'application mobile : les serveurs doivent être capables de gérer un grand nombre d'accès et de transactions simultanées. D'autre part, en termes de vitesse ou de temps de réponse, les opérations de calculs ainsi que les recherches des données de la base doivent être fournies au bout d'un temps réduit.
- Contrainte d'évolution : l'application Mobile et le site doivent permettre une maintenance facile et être susceptibles à évoluer. Ainsi, nous visons à concevoir un ensemble de classes java cohérentes et facilement réutilisables.

## **2.2 Spécification des besoins**

Au niveau de cette étape, nous allons modéliser notre application moyennant les cas d'utilisation qui permettent de structurer les interactions entre les utilisateurs, appelés acteurs (administrateur et client) et le système. Ces cas d'utilisation permettent de relier les actions faites par un utilisateur avec les réactions attendues du système.

### **2.2.1 Diagrammes de spécification cas d'utilisation**

Nous traitons à ce niveau les diagrammes de cas d'utilisation qui permettent de décrire l'interaction entre les acteurs et l'application. C'est pourquoi nous abordons cette section par identifier les acteurs. Nous passons ensuite à détailler les cas d'utilisation de chaque acteur.

#### **2.2.1.1 Identification des acteurs**

Un acteur est une entité externe qui agit sur le système qui, en réponse à l'action de tout acteur fournit un service correspondant à son besoin. Notre application (site Web et application Mobile) met en oeuvre deux acteurs :

- L'administrateur : Cet acteur ne peut intervenir qu'au site Web. En effet, cet acteur est chargé de gérer les comptes des clients et d'apporter les mise à jours nécessaires à notre application.
- Le client : Cet acteur peut jouir des services du site Web ainsi ceux de l'application Mobile.

### 2.2.1.2 Cas d'utilisation d'un administrateur

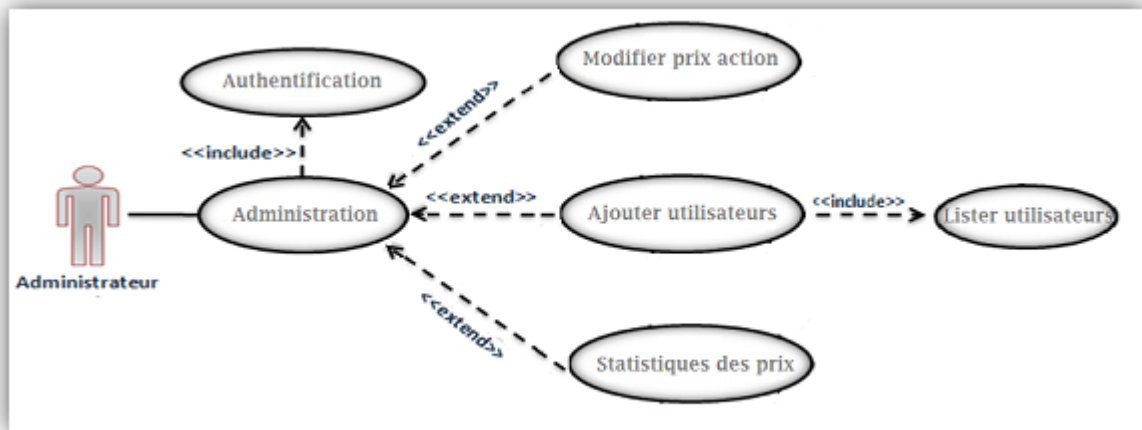


FIGURE 2.1 – Diagramme de cas d'utilisation concernant un Administrateur

Le diagramme des cas d'utilisation de la figure 2.1 présente les diverses fonctionnalités dont jouit l'administrateur de l'application Web. Toutes les actions et les fonctionnalités qui participent à la configuration et à l'administration du système et qui consiste à gérer la base de données à travers la mise à jour des prix des différentes actions et aussi l'ajout des acteurs sont confiées à l'administrateur. Aussi, afin de faciliter la tâche à l'administrateur, le système lui offre l'opportunité de visualiser les statistiques des prix et de lister les différents utilisateur du système via une interface conviviale.

### 2.2.1.3 Cas d'utilisation d'un client

Le diagramme de cas d'utilisation de la figure 2.2 présente les diverses fonctionnalités dont jouit un client de l'application. En effet, la place du client est prépondérante. Le système est avant tout conçu pour des clients auxquels il est nécessaire de s'intéresser en vue d'anticiper leur attente à l'égard du système.

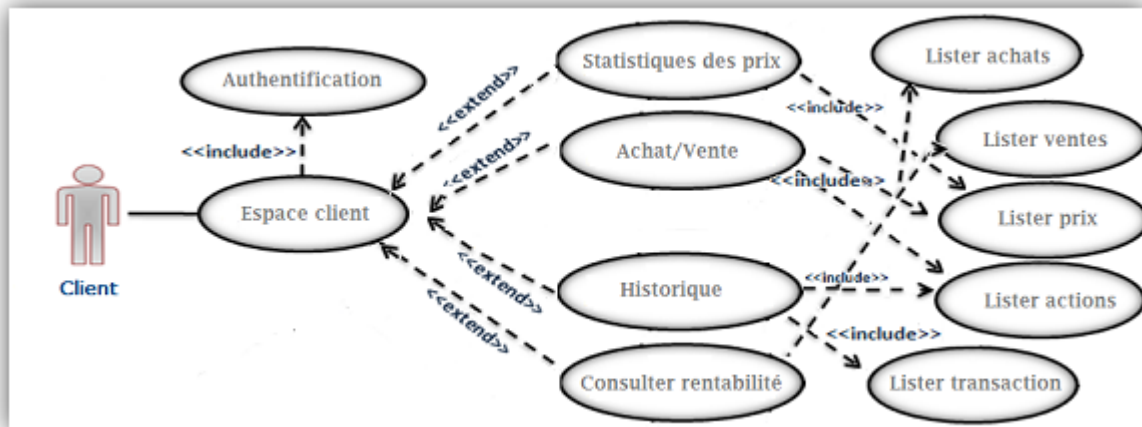


FIGURE 2.2 – Cas d'utilisation pour le client

Tout d'abord, le client doit avoir un compte chez l'intermédiaire en bourse. Il s'agit classiquement d'attributs pour authentification (login, mot de passe).

Ce compte permet donc au à l'utilisateur de s'authentifier chaque fois qu'il veut accéder au service de nos applications Web et Mobile. Cette obligation de contrôler les accès aux ressources consiste essentiellement à stocker les attributs d'authentification(login, mot de passe) pour décider d'une autorisation d'accès et afin d'enregistrer la trace de chaque client. Après l'attribution des droits d'accès, le système fournit au client le contenu de son portefeuille, lui permet de visualiser la rentabilités de son portefeuille et lui fournit les statistiques des prix des différentes actions. Le client peut choisir une des actions présentes pour en vendre ou en acheter. Il peut jeter un coup d'oeil sur la liste de ces transactions spécifiques au type d'action choisit auparavant.

### 2.2.2 Scénario et diagrammes de séquence

Le diagramme de séquence permet de représenter les vues dynamiques du système. En effet, il montre les collaborations entre les objets selon un point de vue temporel en mettant l'accent sur la chronologie des envois des messages. Dans ce qui suit, nous allons représenter les diagrammes de séquences les plus importants de notre système et les éventuels scénarios.

## 2.2.2.1 Partie Administrateur

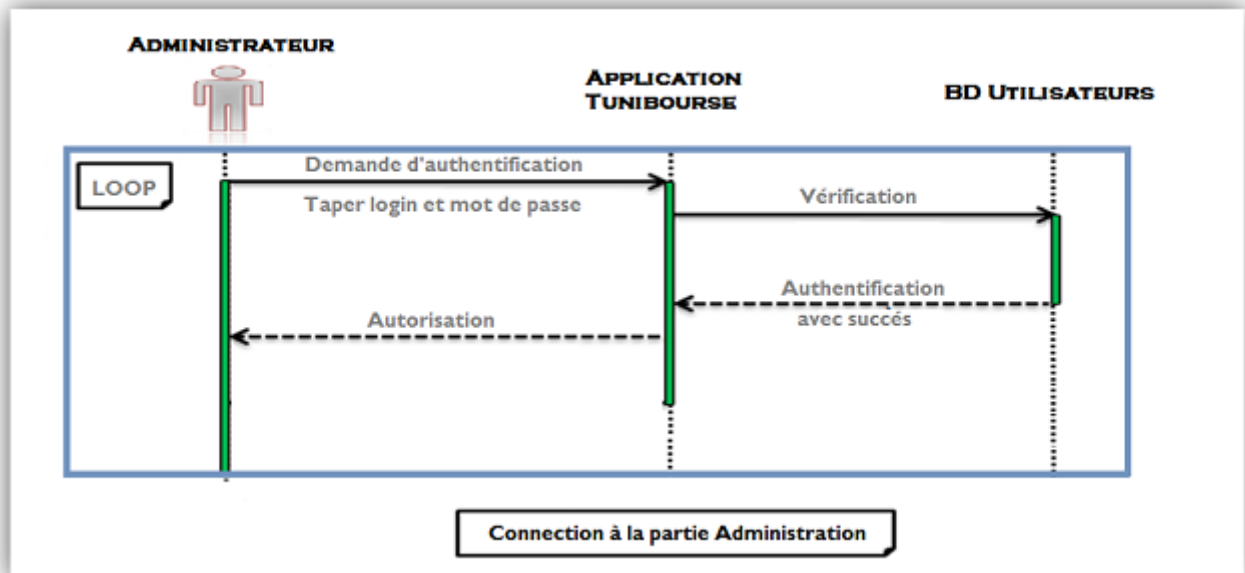


FIGURE 2.3 – Diagramme d’authentification de l’administrateur

Cette figure 2.3 décrit le déroulement de l’authentification de l’administrateur, à savoir la vérification du login et du mot de passe insérés par l’administrateur.

– **scénario :**

L’administrateur saisie le login et le mot de passe qui vont être vérifiés par le système depuis la table Utilisateurs de notre base de données. Dans le cas de la validité du login et du mot de passe, le serveur retourne à l’application le type de l’acteur connecté(administrateur dans notre cas) pour générer ensuite l’espace de l’administrateur. Sinon, un message d’erreur est affiché à l’administrateur qui sera retransmis à la page d’accueil pour saisir les données d’authentification de nouveau.

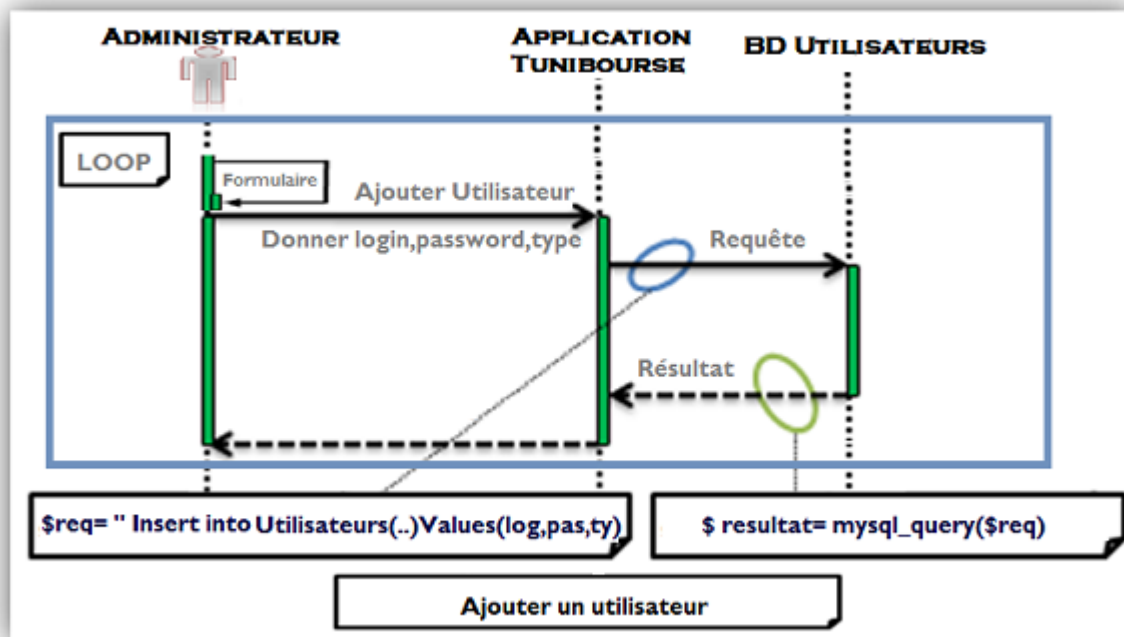


FIGURE 2.4 – Diagramme d'ajout d'un nouveau utilisateur

La figure 2.4 décrit l'ajout d'un acteur par l'administrateur qui doit fournir le login, le mot de passe et le type de l'acteur à ajouter.

– **scénario :**

Tout d'abord, l'administrateur doit s'authentifier. Après, il trouve la possibilité d'ajouter un nouveau acteur tout en remplissant le formulaire approprié. Une fois l'administrateur tape le login, le mot de passe et le type(administrateur ou client) du nouveau acteur, Le système verifie l'unicité du login tapé et en cas d'affirmative, il ajoute le nouveau acteur dans la table Utilisateurs de notre base de données et l'administrateur peut consulter la nouvelle liste des acteurs affichée dans son espace, sinon l'administrateur est redirigé vers cette page de nouveau avec un message d'erreur afin de remplir de nouveau le formulaire.

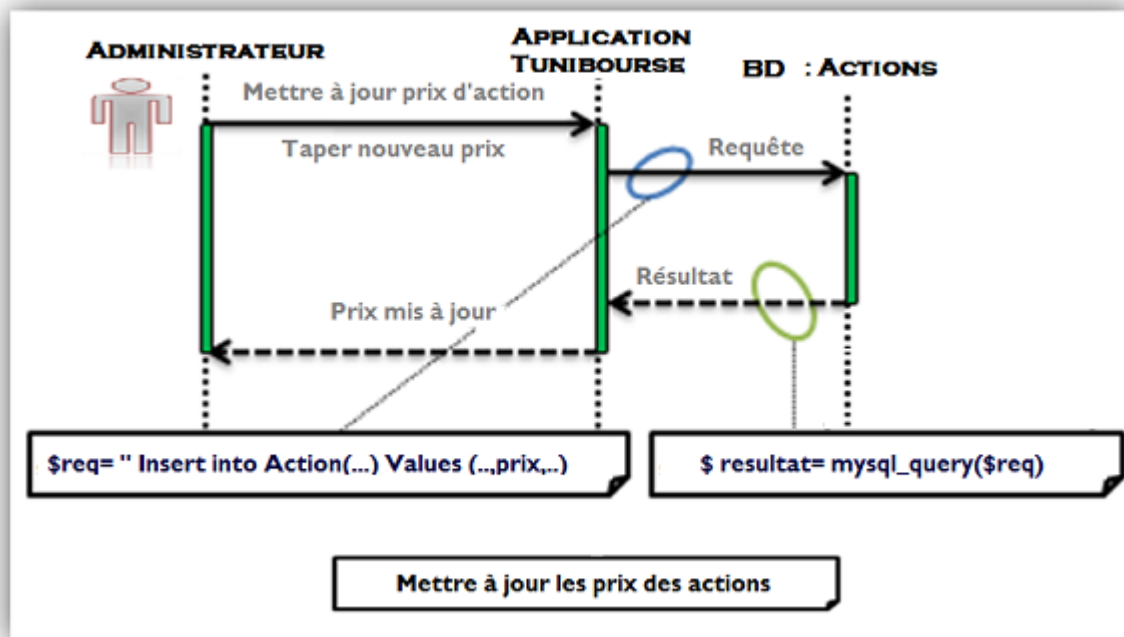


FIGURE 2.5 – Diagrammes de modification des prix des actions

La figure 2.5 décrit le scénario de la modification du prix des actions par l'administrateur qui n'a qu'à taper le nouveau prix pour qu'il soit mis à jour.

– **scénario :**

Tout d'abord, l'administrateur doit s'authentifier. Après, il peut consulter, dans son espace, la courbe de variation des prix de chaque action et en cas d'augmentation ou de diminution d'un prix, l'administrateur doit remplir le formulaire approprié. Le système effectue la modification du prix tout en insérant le nouveau prix de l'action dans la table Actions de notre base de données. Ainsi, l'administrateur peut visualiser le changement de prix dans la liste relative aux modification des prix des actions réalisées par tous les administrateurs du système.

## 2.2.2.2 Partie Client

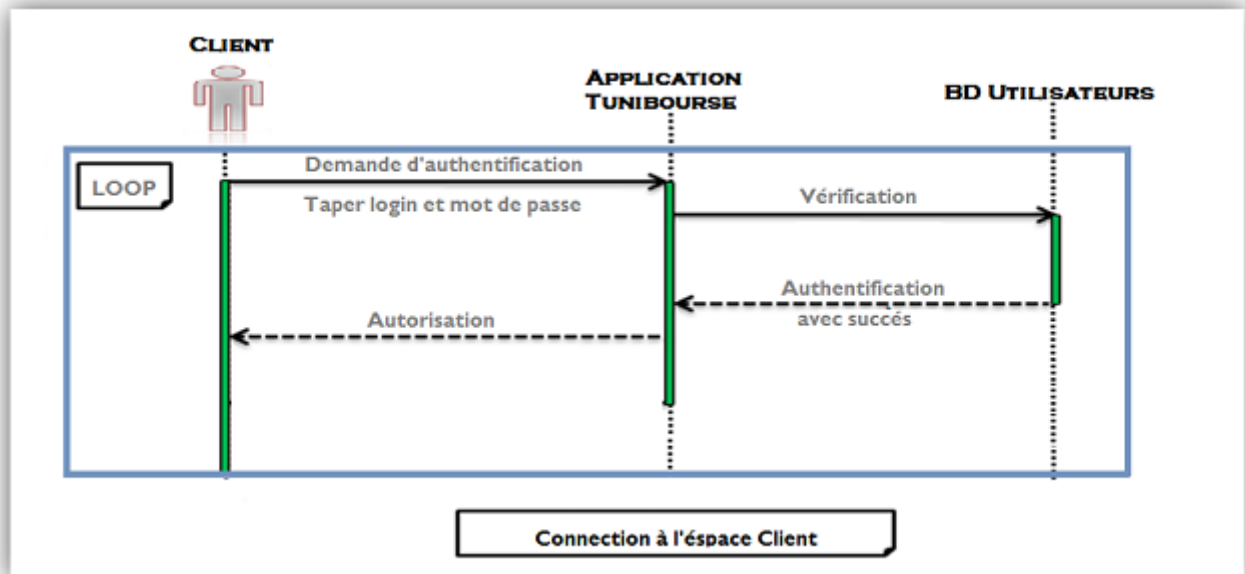


FIGURE 2.6 – Diagramme d'authentification du client

La figure 2.6 décrit le déroulement de l'authentification du client, à savoir la vérification du login et du mot de passe.

– **scénario :**

Le client saisie le login et le mot de passe qui vont être vérifiés par le système depuis la table Utilisateurs de notre base de données. Dans le cas de la validité du login et du mot de passe. Le serveur retourne à l'application le type de l'acteur connecté (client dans notre cas) pour générer ensuite l'espace du client. Sinon, un message d'erreur est affiché au client qui sera invité à saisir les données d'authentification de nouveau.



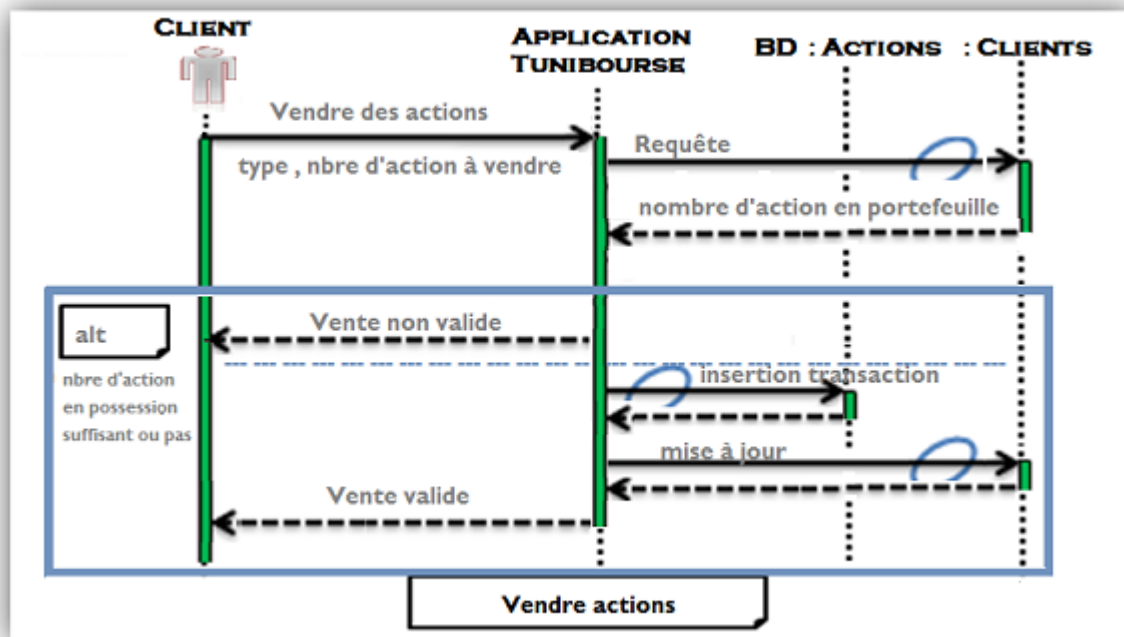


FIGURE 2.7 – Diagramme de Vente d'actions

Dans la figure 2.7 ci dessus, on décrit le scenario de la vente des actions. En effet, le client doit préciser le nombre d'actions à vendre.

– **scénario :**

Tout d'abord le client doit s'authentifier puis choisir l'une des actions pour être dirigé à l'espace relatif à l'action choisie. C'est ici que le client trouve un formulaire où il doit saisir le nombre d'action à vendre. Le système vérifie si le nombre d'actions à vendre est inférieur au nombre d'action du portefeuille du client depuis la table Clients. Dans le cas de la validité de la vente, le serveur retourne à l'application un message de confirmation et le système modifie la table Clients tout en diminuant le nombre d'actions restants dans le portefeuille du client comme il modifie la table Actions qui subit elle même une augmentation au niveau du nombre d'actions restants pour l'intermédiaire en bourse. Sinon, un message d'erreur est affiché au client qui devra resaisir un autre nombre d'actions à vendre inférieur au premier.

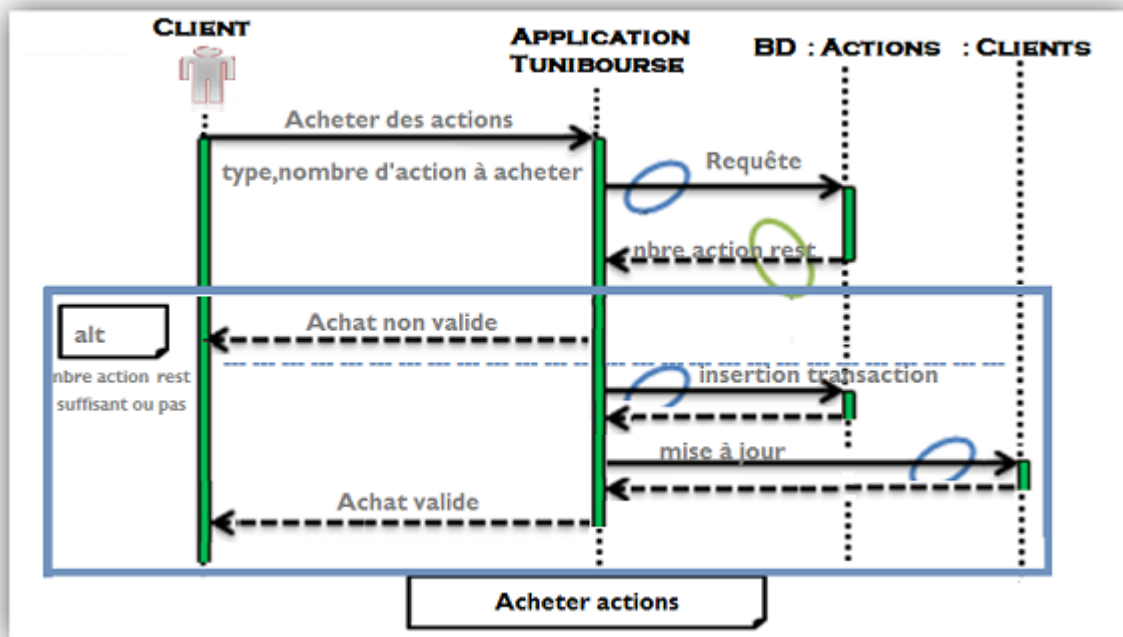


FIGURE 2.8 – Diagramme d'achat d'actions

La figure 2.8 nous décrit le scénario de l'achat des actions.

– scénario :

Tout d'abord le client doit s'authentifier puis choisir l'une des actions pour être dirigé à l'espace relatif à l'action choisie. C'est ici que le client trouve un formulaire où il doit saisir le nombre d'actions à acheter. Le système vérifie si le nombre d'actions à acheter est inférieur au nombre d'action du portefeuille du client depuis la table Clients. Dans le cas de la validité de l'achat, le serveur retourne à l'application un message de confirmation et le système modifie la table Clients tout en augmentant le nombre d'actions restants dans le portefeuille du client comme il modifie la table Actions qui subit elle même une diminution au niveau du nombre d'actions restants pour l'intermédiaire en bourse. Sinon, un message d'erreur est affiché au client qui devra resaisir un autre nombre d'actions à acheter inférieur au premier.

## Conclusion

Après avoir présenté les besoins de l'application ainsi que l'objectif, nous aurons intérêt à exposer la conception de l'application, passant donc au chapitre suivant.

## Chapitre 3

# Conception

### Introduction

**A**près avoir décortiqué les besoins des utilisateurs, nous entamons la phase conception de notre application. Pour cela, nous allons commencer par la spécification de l'architecture adoptée. Ensuite, nous passerons à la modélisation des services et de la base de données à travers les modèles UML.

### 3.1 Conception globale et architecture

Nous aurons besoin à cette étape de spécifier l'architecture de notre application. Nous allons étudier quelques architectures et voir si elles permettent de répondre à notre besoin ou pas.

#### 3.1.1 Architecture du site web

Dans cette sous section, nous allons étudier les différentes architectures utilisées dans l'application Web.

##### 3.1.1.1 Architecture générale MVC

La conception de notre application porte essentiellement sur l'architecture de l'application et la conception des données requises par l'application de gestion des projets de deux modules. Le motif de conception suivi est le motif MVC (Modèle Vue Contrôleur) .

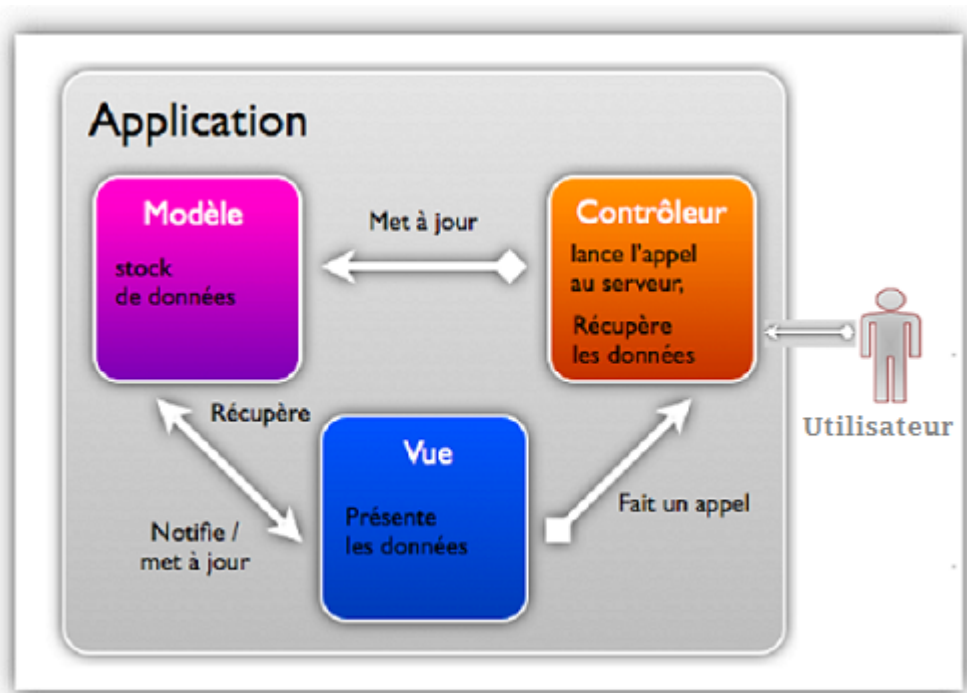


FIGURE 3.1 – Architecture MVC

MVC est une architecture à trois couches utilisée pour la programmation client/serveur et d'interface graphique. C'est un modèle architectural très puissant qui intervient dans la réalisation d'une application. Il tire sa puissance de son concept de base qui est la séparation de modèle, vue et contrôleur.

C'est trois couches sont décrites comme suit :

- **Modèle** : Le modèle représente le coeur de l'application : traitements des données, interactions avec la base de données. Il décrit les données manipulées par l'application. Il regroupe la gestion de ces données et est responsable de leur intégrité. La base de données sera l'un de ses composants. Le modèle comporte des méthodes standards pour mettre à jour ces données (insertion, suppression, mise à jour). Il offre aussi des méthodes pour récupérer ces données. Les résultats renvoyés par le modèle ne s'occupent pas de la présentation. Le modèle ne contient aucun lien direct vers le contrôleur ou la vue. Sa communication avec la vue s'effectue au travers du patron Observateur.

- **Vue** : Ce avec quoi l'utilisateur interagit se nomme précisément la vue. Sa première tâche est de présenter les résultats renvoyés par le modèle. Sa seconde tâche est de recevoir toute

action de l'utilisateur (clic de souris, sélection d'un bouton radio, entrée de texte, etc...). Ces différents événements sont envoyés au contrôleur. La vue n'effectue pas de traitement, elle se contente d'afficher les résultats des traitements effectués par le modèle et d'interagir avec l'utilisateur.

Par exemple, un utilisateur qui ôse faire une transaction doit consulter la vue appropriée qui lui permet de remplir les données nécessaires du formulaire.etc...

- **Contrôleur** : Le contrôleur prend en charge la gestion des événements de synchronisation pour mettre à jour la vue ou le modèle et les synchroniser. Il reçoit tous les événements de l'utilisateur et déclenche les actions à effectuer. Si une action nécessite un changement des données, le contrôleur demande la modification des données au modèle, et ce dernier notifie la vue que les données ont changé pour qu'elle se mette à jour. D'après le patron de conception observateur/observable, la vue est un « observateur » du modèle qui est lui « observable. » Certains événements de l'utilisateur ne concernent pas les données mais la vue. Dans ce cas, le contrôleur demande à la vue de se modifier. Le contrôleur n'effectue aucun traitement, ne modifie aucune donnée. Il analyse la requête du client et se contente d'appeler le modèle adéquat et de renvoyer la vue correspondant à la demande.

Par exemple, dans notre cas, une action de l'utilisateur peut être l'entrée d'un nouveau acteur. Le contrôleur ajoute cet acteur au modèle et demande sa prise en compte par la vue. Une action de l'utilisateur peut aussi être de sélectionner une nouvelle action pour visualiser certaines statistiques de variation de prix. Ceci ne modifie pas la base des cours mais nécessite simplement que la vue s'adapte et offre à l'utilisateur une vision des statistiques de variation de prix de cette action. Quand un même objet contrôleur reçoit les événements de tous les composants, il lui faut déterminer quelle est l'origine de chaque événement. Ce tri des événements peut s'avérer fastidieux et peut conduire à un code peu élégant (un énorme switch). C'est pourquoi le contrôleur est souvent scindé en plusieurs parties dont chacune reçoit les événements d'une partie des composants.

### 3.1.1.2 Java ServerFaces et MVC

Java ServerFaces (JSF) est un framework de développement d'applications Web basé sur plusieurs composants , qui permet de développer des interfaces client riches tout en respectant le paradigme MVC. Concrètement, JSF est une spécification (actuellement en version 2.2) pour

fournir un ensemble d'API et de jeux de composants pour le développement d'applications web similairement au développement d'application (( lourdes )) en Swing par exemple.

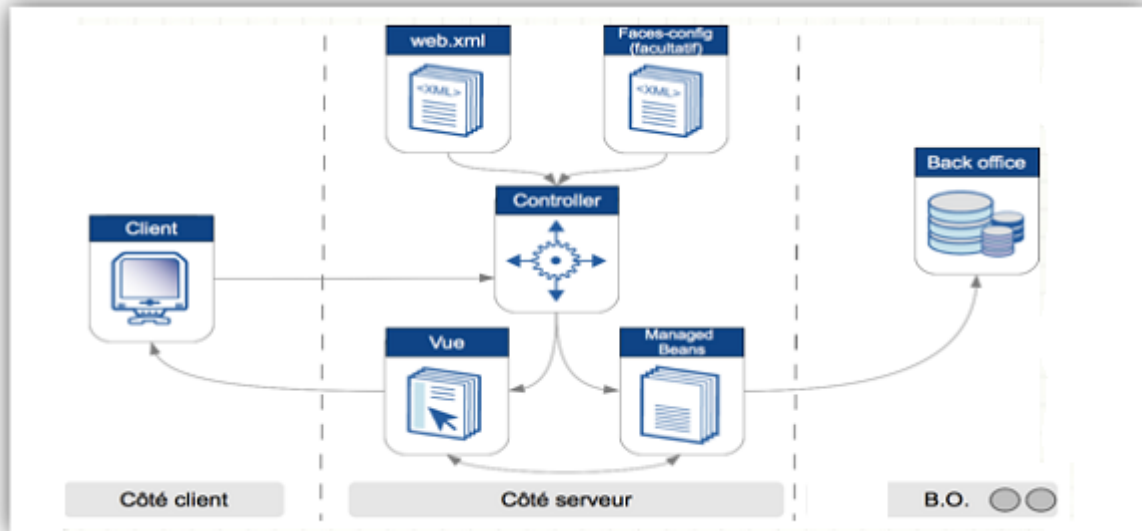


FIGURE 3.2 – Architecture du framework JSF

### 3.1.1.3 Serveur d'application

Le serveur d'application est l'environnement d'exécution des applications côté serveur. Il prend en charge l'ensemble des fonctionnalités qui permettent à N clients d'utiliser une même application :

- Gestion de la session utilisateur : N clients utilisant une même instance d'application sur le serveur, il est nécessaire que le serveur d'application puisse conserver des contextes.
- Un identifiant propre à chaque utilisateur : La plupart des serveurs d'application génèrent un identifiant unique pour chaque nouveau client et transmettent cet identifiant lors de chaque échange http par URL longs, variables cachées ou cookies.
- Ouverture sur de multiple sources de données : C'est le serveur d'application qui rend accessible les données des applications du système d'information. Il doit donc pouvoir accéder à de nombreuses sources de données.

### 3.1.2 Architecture de l'application Mobile

Sous Android, une application est composée d'une ou plusieurs activités. Une activité est la base d'un composant pour la création d'interfaces utilisateur.

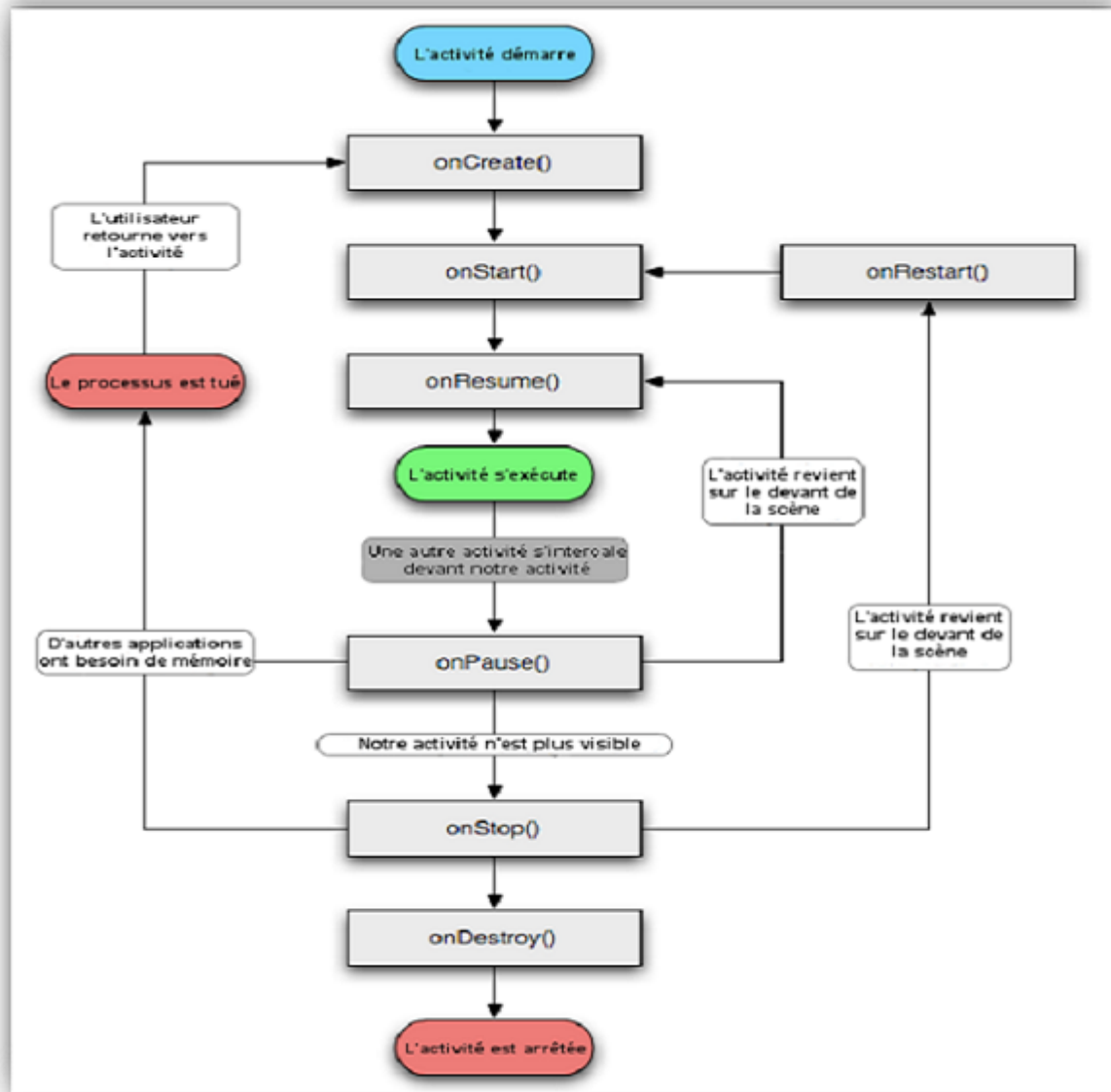


FIGURE 3.3 – Cycle de vie d'une Activité sous Android

- L'activité démarre : la méthode onCreate est appelée.
- Pendant l'utilisation d'une activité, l'utilisateur presse la touche Accueil, ou bien l'application téléphone, qualifiée comme prioritaire et qui interrompt son fonctionnement par un appel téléphonique entrant. L'activité est arrêtée (appel de onStop), le développeur détermine l'impact

sur l'interface utilisateur, par exemple la mise en pause d'une animation puisque l'activité n'est plus visible.

- Une fois l'appel téléphonique terminé, le système réveille l'activité précédemment mise en pause (appel de `onRestart`, `onStart`).
- L'activité reste trop longtemps en pause, le système a besoin de mémoire, il détruit l'activité (appel de `onDestroy`).
- `onPause` et `onResume` rajoutent un état à l'activité, puisqu'ils interviennent dans le cas d'activités partiellement visibles, mais qui n'ont pas le focus. La méthode `onPause` implique également que la vie de cette application n'est plus une priorité pour le système. Donc si celui-ci a besoin de mémoire, l'Activity peut être fermée. Ainsi, il est préférable, lorsque l'on utilise cette méthode, de sauvegarder l'état de l'activité dans le cas où l'utilisateur souhaiterait y revenir avec la touche Accueil.

### 3.1.2.1 REST

La conception de l'application Android porte essentiellement sur la consommation du web Service Jax-Rs qui est basé sur REST.

REST est un style d'architecture réseau pour Web Services qui met l'accent sur la définition de ressources identifiées par des URI, et utilise les messages du protocole HTTP pour définir la sémantique de la communication client/serveur : GET pour le rapatriement d'une ressource, POST pour une création, PUT pour une modification/création, DELETE pour un effacement.

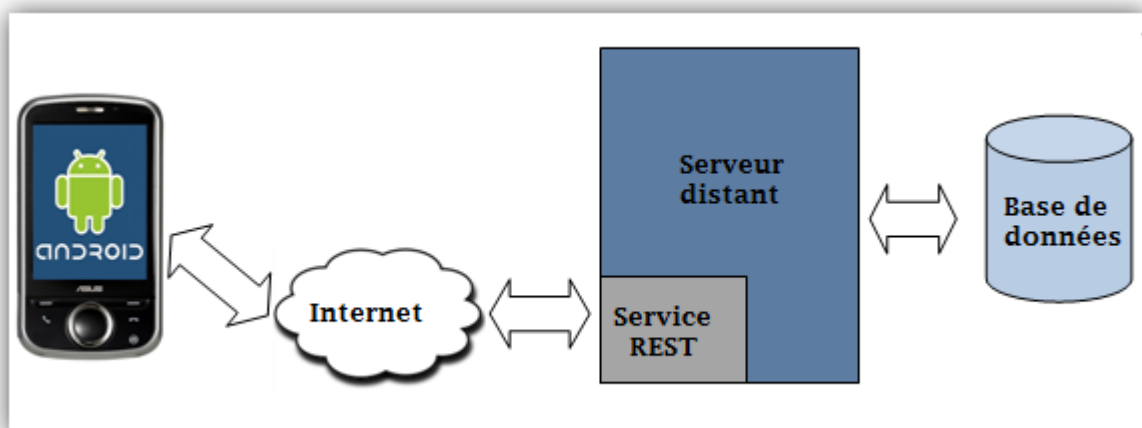


FIGURE 3.4 – Architecture REST



### 3.1.2.2 JAX-RS

JAX-RS : Java API for RESTful Web Services est une interface de programmation Java permettant de créer des services Web avec une architecture REST.[1] JAX-RS est la standardisation des services REST en Java. Cette spécification définit un ensemble d'annotations permettant d'exposer des classes java comme service REST. L'intérêt de la standardisation est de ne pas se lier à une implémentation spécifique.

#### Annotations

Pour implémenter les services REST, JAX-RS définit un certain nombre d'annotation. On retrouvera principalement :

- @Path : définit le chemin de la ressource. Cette annotation se place sur la classe et/ou sur la méthode implémentant le service.
- @GET, @PUT, @POST, @DELETE : définit le verbe implémenté par le service.
- @Produces spécifie le ou les Types MIME de la réponse du service.
- @Consumes : spécifie le ou les Types MIME acceptés en entrée du service.

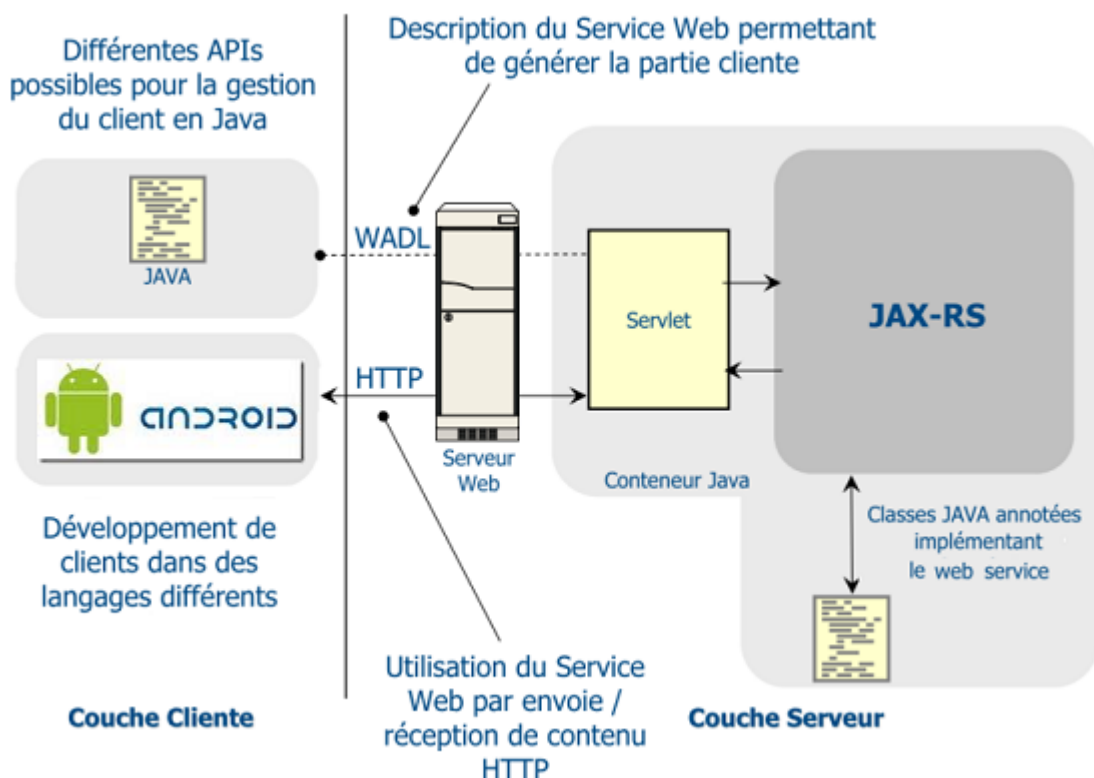


FIGURE 3.5 – Architecture REST

### 3.2 Diagramme des packages du système

Un élément important de la conception logicielle est le découpage en modules ou paquets. Un module ou un paquetage permet de regrouper des classes logiquement liées. L'organisation de l'ensemble des modules composant une application ou un système constitue son architecture logicielle.

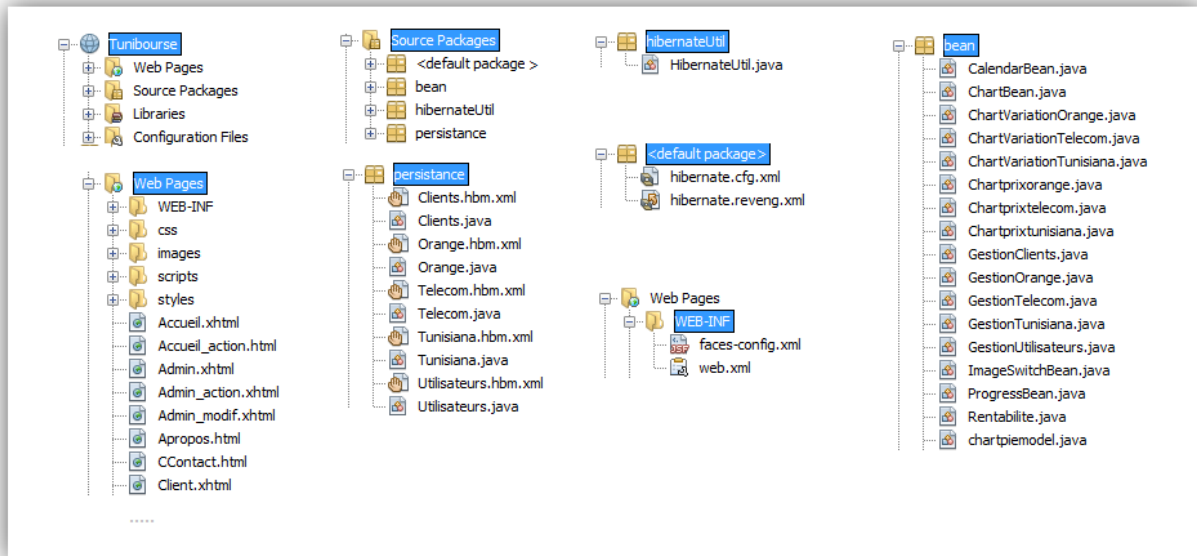


FIGURE 3.6 – Diagramme des packages du site web

- **Web Pages** : contient les vues qui constituent l'interface graphique du projet et ses dépendances en style css et code java script et images et d'autres packages qui seront décrits dans la suite.

- **persistence** : regroupe les classes qui modélisent notre base de données.

Toutes les instances d'une classe persistante ne sont pas forcément dans l'état persistant - au lieu de cela, une instance peut être éphémère ou détachée.

Hibernate fonctionne de manière optimale lorsque ces classes suivent quelques règles simples, aussi connues comme le modèle de programmation Plain Old Java Object (POJO). Cependant, aucune de ces règles ne sont des besoins absolus. En effet, Hibernate3 suppose très peu de choses à propos de la nature des objets persistants.

- **bean** : Les beans sont largement utilisées dans une application utilisant JSF notamment pour permettre l'échange de données entre les différentes entités et le traitement des événements.

Les beans managés sont des javabeans dont le cycle de vie va être géré par le framework JSF en fonction des besoins et du paramétrage fourni dans le fichier de configuration faces-config.xml.

- **hibernateUtil** : s'occupe du démarrage et rend la gestion des Sessions plus facile.

Pour gérer les accès à la base de données, Hibernate utilise des sessions. Ce sont des instances de la classe org.hibernate.session qui permettent d'encapsuler une connexion JDBC. en effet la classe HibernateUtil permet de gérer les sessions Hibernate et de remplir la base de données au lancement du programme pour effectuer les tests.

- **Web Inf** : Toute application utilisant JSF doit posséder deux fichiers de configuration au format XML ,contenu dans le répertoire WEB-INF, qui vont contenir les informations nécessaires à la bonne configuration et exécution de l'application. Le premier fichier web.xml est le descripteur de toute application web J2EE. Le second fichier faces-config.xml est un fichier de configuration particulier au paramétrage de JSF.

- web.xml : Les applications Web Java l'utilisent comme un fichier descripteur de déploiement pour définir la méthode de mappage des URL , déterminer les URL qui nécessitent une authentification ainsi que d'autres informations.

Le fichier web.xml doit contenir au minimum certaines informations notamment, la servlet ou classe faisant office de contrôleur, la première vue lancée après le démarrage et d'autres paramètres.

- faces-config.xml : Le fichier gérant la logique de l'application web.

Ce fichier au format XML permet de définir et de fournir des valeurs d'initialisation pour des ressources nécessaires à l'application utilisant JSF. Le tag racine de ce document XML est le tag <face-config>. Ce tag peut avoir plusieurs tags fils : Le tag <application> permet de préciser des informations sur les entités utilisées par l'internationalisation et/ou de remplacer des éléments de l'application.

Chacun des beans managés doit être déclaré avec un tag <managed-bean> qui possède trois tag fils :

- <managed-bean-name> : le nom attribué au bean (celui qui sera utilisé lors de son utilisation).
- <managed-bean-class> : le type pleinement qualifié de la classe du bean.
- <managed-bean-scope> : précise la portée dans laquelle le bean sera stockée et donc utilisable.

Ces informations seront utilisées par JSF pour automatiser la création ou la récupération d'un bean lorsque celui-ci sera utilisé dans l'application. Le grand intérêt de ce mécanisme est de ne pas avoir à se soucier de l'instanciation du bean ou de sa recherche dans la portée puisque c'est le framework qui va s'en occuper de façon transparente.

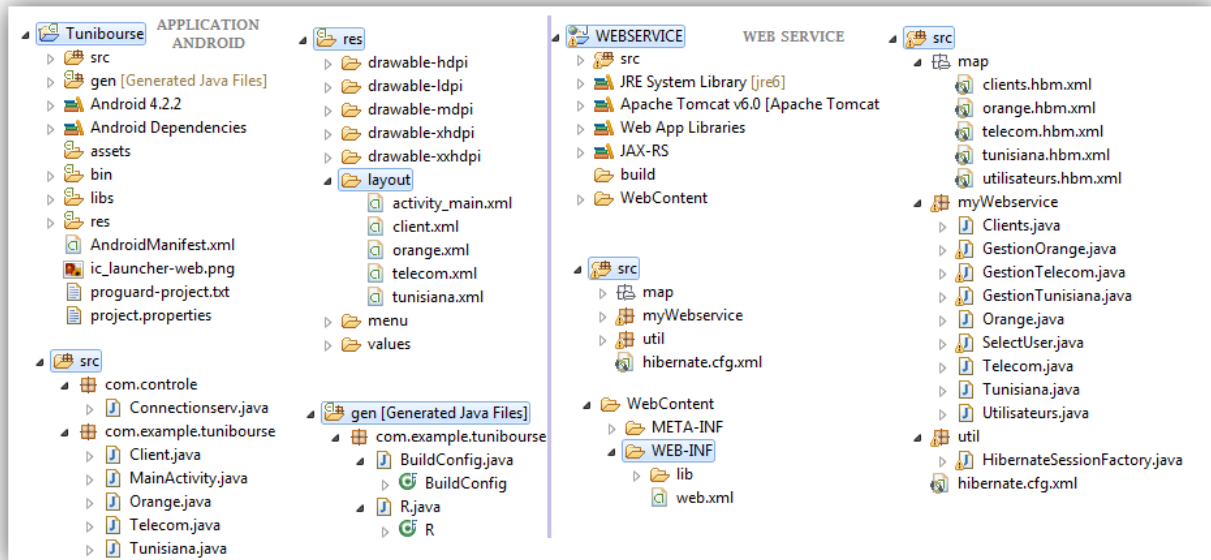


FIGURE 3.7 – Diagramme des packages de l'application Android et du Web service

La figure ci dessus décrit la répartition des différentes classes utilisées dans l'application Android et le Web Service. En effet, pour l'application Android, les classes .java qui nous génèrent l'interface graphique sont regroupées sous le chemin suivant : src/com.example.tunibourse et la classe qui nous garantie la connection au Web Service et sous mise sous le chemin suivant : src/com.controle. Quant aux pages .xml, elles doivent être localisées dans le fichier /res/layout.

#### • Application Android TuniBourse :

- AndroidManifest.xml : c'est dans ce fichier que l'on déclare ce que contiendra l'application (les activités, les services, etc). On y indique également la façon dont ces composants seront reliés au système Android lui-même en précisant, par exemple, l'activité (ou les activités) qui doivent apparaître dans le menu principal du terminal.
- bin/ : contient l'application compilée.
- gen/ : contient le code source produit par les outils de compilation d'Android.
- libs/ : contient les fichiers JAR extérieurs nécessaires à l'application.

- src/ : contient le code source Java de l'application.
- assets/ : contient les autres fichiers statiques fournis avec l'application pour son déploiement sur le terminal.
- res/ : contient les ressources icônes, descriptions des éléments de l'interface graphique (layouts), etc. empaquetées avec le code Java compilé. Parmi les sous-répertoires de /res, citons :

- res/drawable/ pour les images (PNG, JPEG, etc.).
- res/layout/ pour les descriptions XML de la composition de l'interface graphique.
- res/menu/ pour les descriptions XML des menus.
- res/raw/ pour les fichiers généraux (un fichier CSV contenant les informations d'un compte, par exemple).
- res/values/ pour les messages, les dimensions, etc.
- res/xml/ pour les autres fichiers XML généraux que vous souhaitez fournir.

● **Web Service :**

- src/ : contient 3 packages (map, myWebservice et util)
  - map : contient les fichiers (.hbm.xml) qui représentent les différentes tables de la base de données.
  - myWebservice : contient les fichiers (.java) qui contiennent les setters et les getters des différentes classes représentant des tables de la bases et aussi les classes dans lesquelles on a définit les différentes requêtes.
  - util : contient la classe HibernateSessionFactory qui s'occupe de l'ouverture des sessions.

### 3.3 Conception détaillée

Nous présentons les dépendances entre les packages ainsi que les diagrammes de base de données.

#### 3.3.1 Conception détaillée côté site web

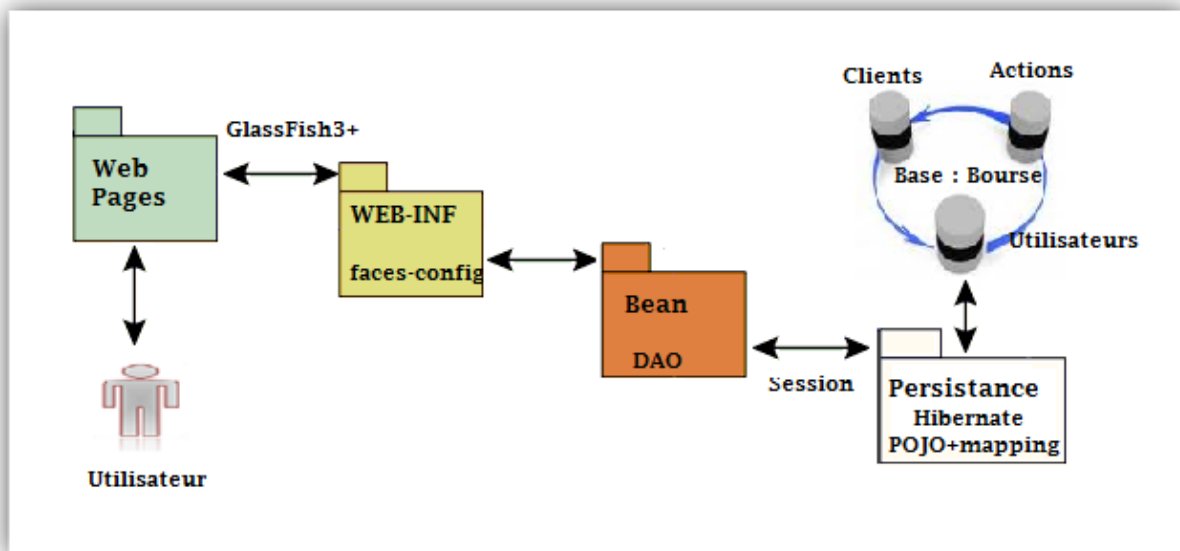


FIGURE 3.8 – Schéma de dépendances entre les packages du site web

La figure ci dessus décrit les différentes dépendances entre les packages. En fait, l'utilisateur réagit avec l'interface graphique gérée par le Web Pages qui se contente de recevoir les données et les afficher. Ceci nécessite une configuration du Web-Inf qui relie entre le Web pages et le Bean. Ce dernier se connecte à la base après avoir ouvrir une session hibernate en utilisant le package Persistance qui est le représentant des tables de la base de données dans notre projet. Puis on aura la véhiculation de l'information au sens inverse selon la valeur du traitement fait.

### 3.3.2 Conception détaillée côté Android et web service

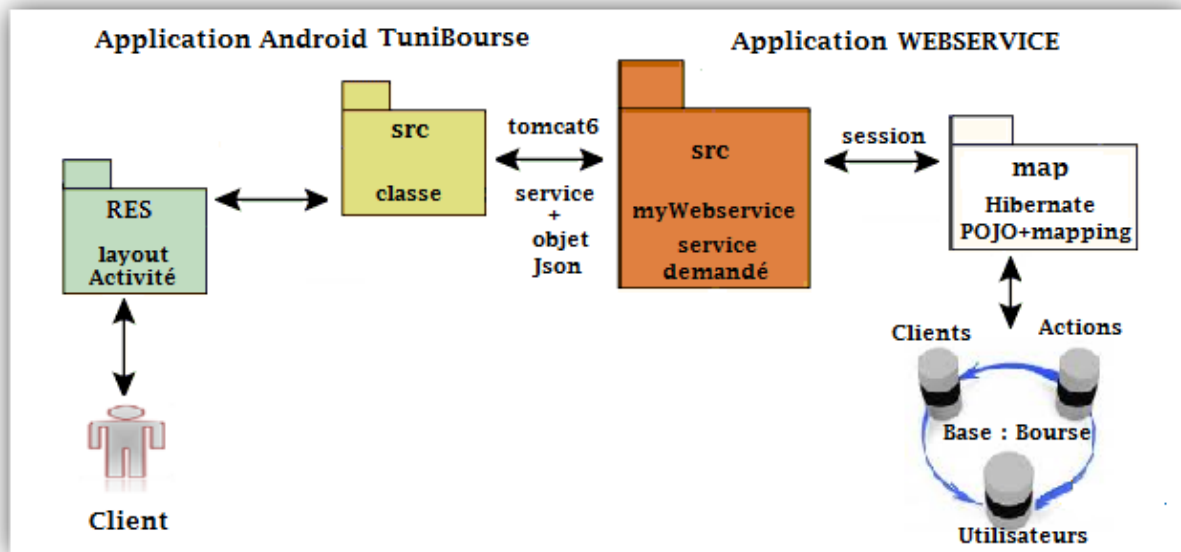


FIGURE 3.9 – Schéma de dépendances entre les packages de l'application Android

On dispose de deux parties qui communiquent à travers un serveur apache tomcat6 : une application Android TuniBourse et une application WEBSERVICE. L'application Android TuniBourse contient essentiellement les classes .xml et les classes .java qui collaborent entre eux pour fournir au client une interface graphique conviviale. Une fois cette application a besoin de récupérer des données de la base ou même d'envoyer des données à la base, un objet JSON doit être construit et envoyé à travers le tomcat6 vers l'application WEBSERVICE qui se connecte à la base pour récupérer ou insérer les données nécessaires puis retourne un objet json ou un simple message pour indiquer l'état de la fonction vérifiée ou pas.

## 3.3.3 Diagramme de la base des données

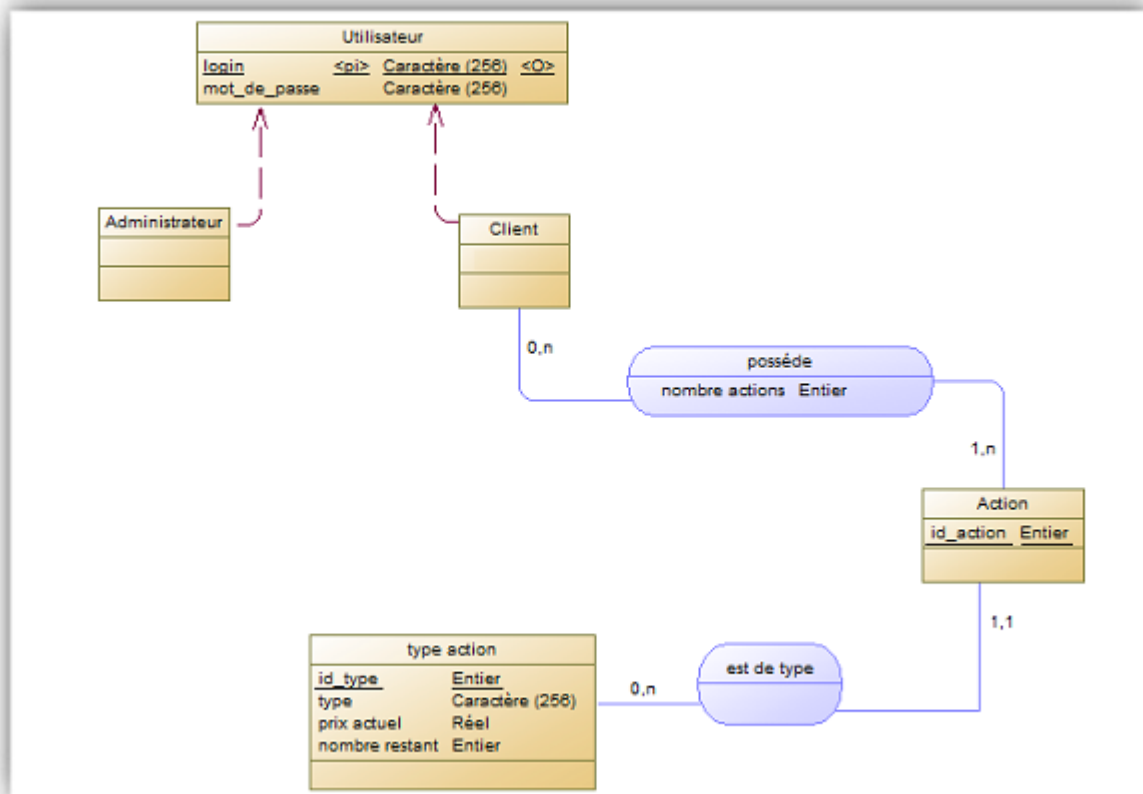


FIGURE 3.10 – Base des données Bourse

A partir des besoins dégagés, nous avons pu distinguer les entités nécessaires pour le bon fonctionnement de cette application :

- **Utilisateur** (login, mot de passe)
- **Client** ()
- **Administrateur** ()
- **Action** (id-action)
- **Type action** (id-type, type, prix-actuel, nombre-restant)
- **possède** (#login, #id-action, nombre-actions)



### 3.4 Diagrammes d'activités

Les diagrammes d'activités sont utilisés pour illustrer les flux de travail et les événements dans un système, du niveau métier jusqu'au niveau opérationnel.

#### 3.4.1 Diagrammes d'activités de l'application Web

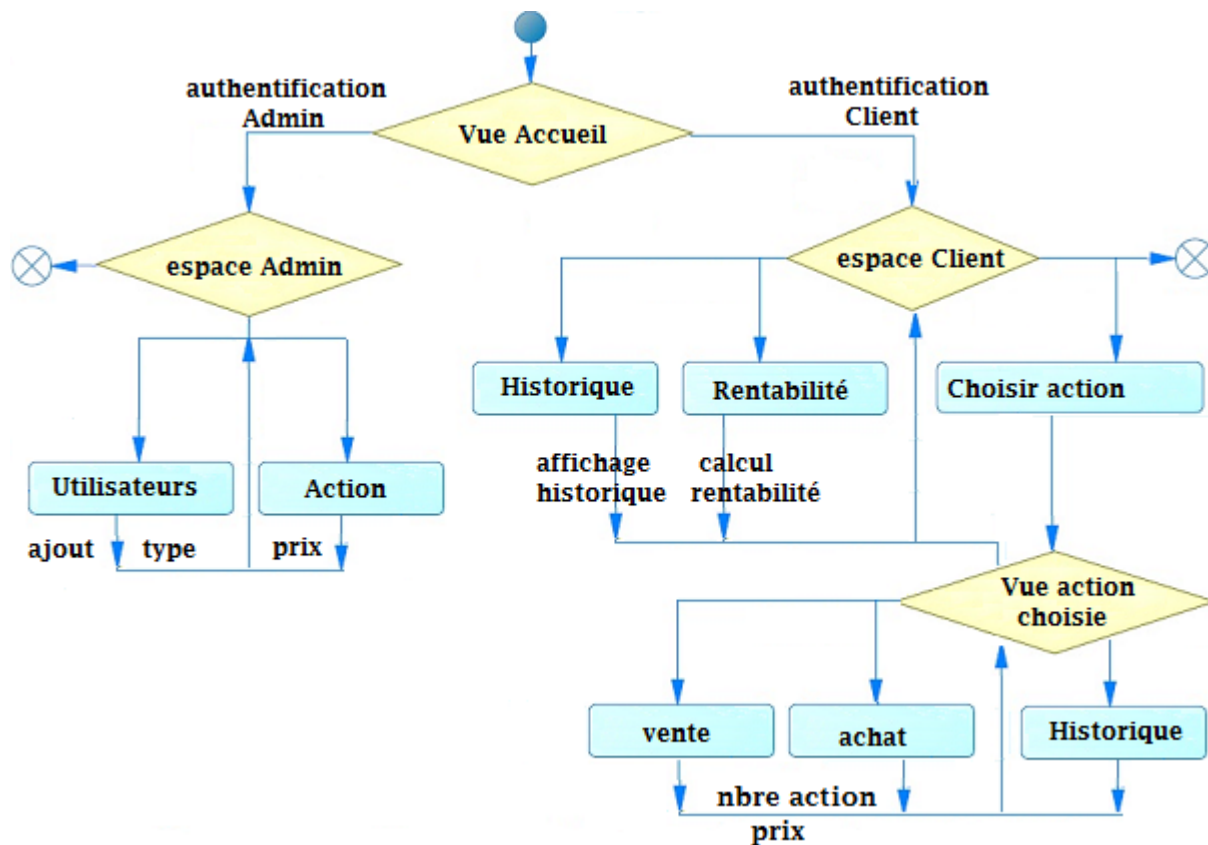


FIGURE 3.11 – Diagramme d'activité du site web

Le diagramme d'activité ci-dessus décrit l'aspect dynamique de notre site web. En effet, selon le type de l'acteur qui s'est authentifié, on aura un espace réservé au client et un autre réservé à l'administrateur.

- Si l'acteur authentifié est un client alors une page s'ouvre contenant le portefeuille du client bien détaillé et offrant au client de visualiser la rentabilité de ces actions. Le client peut à partir du menu de cette page soit se déconnecter, soit aller aux pages réservées aux ventes et achats des différentes actions. Une fois le client clique sur le bouton de l'une des actions, il sera transmis vers une autre page spécifique à cette action là. Toutes les

informations possibles sur ce type d'action sera fournis au client comme un diagramme modélisant les ventes, les achats et le prix actuel de l'action, ainsi qu'une table d'historique. Et évidemment, le client peut dans cette page acheter ou vendre des actions.

- Si l'acteur authentifié est un administrateur, il aura la possibilité d'ajouter de nouveaux acteurs comme il aura aussi la possibilité de mettre à jour les prix des actions.

### 3.4.2 Diagrammes d'activités de l'application Mobile

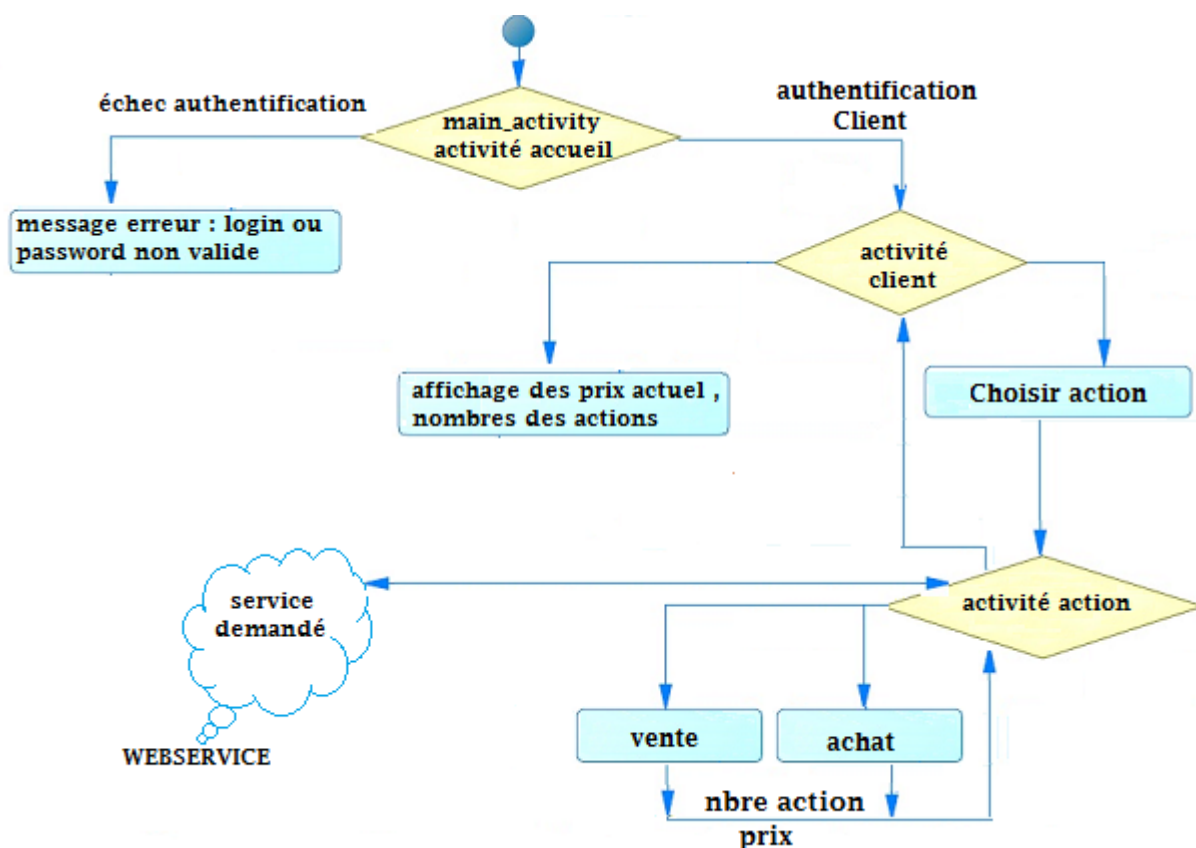


FIGURE 3.12 – Diagramme d'activité de l'application Mobile

Le diagramme d'activité ci-dessus décrit l'aspect dynamique de l'application Mobile. En effet, cette application est dédiée uniquement pour le client qui, suite à son authentification par l'insertion de son login et son mot de passe, sera transmis dans une autre page où il pourra visualiser les prix des différentes actions et le nombre d'action qu'il possède.

Au cas ou il décide d'acheter ou de vendre des actions il n'a qu'à cliquer sur un bouton pour être transmis à une page dans laquelle il sera invité à remplir un formulaire pour finaliser l'achat ou la vente d'actions.

## **Conclusion**

**T**Out au long de ce chapitre, nous avons détaillé la conception de notre base de données et de nos applications Web et Mobile à travers leur différents diagrammes de navigation et d'activité afin que ça soit le plus souple et le plus aisé pour l'étape suivante. Le chapitre suivant met en évidence, le fruit de ce passage et les différents résultats du développement des applications demandées.

# Chapitre 4

## *Réalisation*

### Introduction

**A**près avoir réalisé la phase de spécification et de conception de notre projet, nous consacrons ce chapitre pour : la présentation de l'environnement matériel et logiciel dans la première section puis nous présentons dans une deuxième section les choix technologiques et nous finirons par présenter les résultats de l'implémentation de nos applications Web et Mobile dans la troisième section.

### 4.1 Environnement de travail

Nous présenterons dans cette section l'environnements matériel et logiciel qui nous ont permis de mettre en oeuvre notre application. En effet, nous réservons la première sous section pour la description des matériels utilisés alors que dans la section qui suit nous parlons des logiciels adoptés pour la réalisation de notre projet.

#### 4.1.1 Environnement materiel

Pour la réalisation de ce projet, nous avons disposé de deux micro-ordinateurs dont la configuration de chacun est la suivante :

- Première machine :
  - Processeur : Intel(R) Core(TM) i5 CPU M480 @ 2.67GHz 2.67 GHz .

- Système d’exploitation : Windows 7 Professionel.
  - RAM : 4.00 Go.
  - Disque dur : 386 Go.
- Deuxième machine :
    - Processeur : Intel(R) Core(TM)2 Duo CPU P7350 @ 2.00GHz 2.00 GHz .
    - Système d’exploitation : Windows 7 Edition Intégrale.
    - RAM : 4.00 Go.
    - Disque dur : 300 Go.

#### 4.1.2 Environnement logiciel

Pendant la phase de développement de notre application nous avons employé les outils cités dans le tableau si dessous :

Logiciel	Version	Fonctionnalité
Eclipse	Helios	Serveur d’application
Apache Tomcat	6.0	Serveur web
MySQL	5.1	Serveur de base de données
Androïd	2.2 API Level 8	Plateforme mobile
NetBeans	IDE 7.2.1	Serveur d’application

FIGURE 4.1 – Tableau des outils logiciels

## 4.2 Choix technologiques

Cette partie traitera les choix effectués pour l’élaboration de notre projet à savoir les langages de développement, les environnements de travail, la plateforme et le serveur de la base de données.

## 4.2.1 Choix technologiques communs

### 4.2.1.1 Choix du langage Java

Java est l'un des langages les plus répandus et ceci grâce à ces caractéristiques :

- **Fiable** : La conception de java incite le développeur à traquer les erreurs, lancer des vérifications dynamiques, éliminer les situations qui causent les erreurs...Ce qui le différencie par rapport à C++ est le fait que java dispose d'un modèle de pointeur qui évite les risques d'écrasement de la mémoire et d'endommagement des données.
- **Orienté objet** : Implémente les concepts de base de la programmation orientée objet à savoir l'utilisation des classes, l'encapsulation, l'héritage...
- **Simple** : Java a été conçu de manière analogue au langage C mais possédant une syntaxe plus simple. Les pointeurs en Java ont été écartés et donc il n'y a pas de manipulation directe de la mémoire. En plus, les API (Application Programming Interface) sont riches offrant de nombreuses fonctionnalités accélérant ainsi le développement.
- **Sécurisé** : Puisque Java peut être exploité pour les environnements serveurs et distribués il permet la construction des systèmes inaltérables et sans virus. D'un autre côté, le compilateur interdit l'accès direct à la mémoire et l'accès au disque dur est aussi réglementé.
- **Portable** : Après compilation, le code Java se transforme en Byte Code (très proche du langage machine) qui sera interprété et exécuté par la suite par la JVM (Java Virtual Machine). Il est donc possible d'exécuter les programmes Java sur toute machine possédant une JVM.

### 4.2.1.2 Choix du serveur MYSQL

Bien que le serveur MySQL admette beaucoup de concurrents, il possède sur eux plusieurs avantages :

- **Performances** : MySQL est un système très rapide. En 2002, eWeek a publié un banc d'essais qui comparait cinq bases de données alimentant une application web. Le meilleur résultat a placé MySQL et Oracle à la même position, sachant qu'Oracle est bien plus cher que MySQL.
- **Coût réduit** : MySQL est disponible gratuitement sous une licence open-source ou, pour un prix abordable sous licence commerciale. L'utilisateur peut selon le besoin acheter une licence et donc s'il ne souhaite pas distribuer son application l'achat d'une licence n'est pas nécessaire.
- **Simplicité d'emploi** : MySQL est bien plus simple à installer que les autres produits concur-

rents. L'apprentissage requis à l'acquisition des compétences d'un DBA (DataBase Administrator) est courte comparé à celle des autres bases de données. Ceci est dû au fait que la base de données de MySQL utilise SQL.

- **Potabilité** : MySQL a la possibilité de fonctionner convenablement sur plusieurs systèmes d'exploitation plus précisément Windows et les systèmes Unix.

## 4.2.2 Choix technologiques pour l'application Web

### 4.2.2.1 Choix de J2EE

Avant d'aborder la phase d'implémentation du site il était primordial de choisir la plate-forme adéquate qui nous permettra d'implémenter d'une manière cohérente et de façon intégrée toutes les couches prévues par notre architecture dans la phase conception. Le choix était limité entre la plate-forme .NET et J2EE.

- JEE :une plate-forme d'exécution par excellence J2EE signifie java 2 Entreprise Edition et repose donc sur java, langage objet conçu pour être " portable ". Une caractéristique qui passe donc notamment par l'utilisation d'un module client (nommé machine virtuelle) pour assurer l'exécution des programmes sur n'importe quel type de serveur et de system d'exploitation.
- .NET : une plate-forme d'intégration, à la différence de J2EE dont le socle est utilisable par d'autres fournisseurs de serveurs d'applications, la plate-forme .net reste la propriété exclusive de son créateur : Microsoft. Elle se compose de plusieurs produits : le system d'exploitation Windows et ses dérivés dédiés au terminaux portables (Windows CE.NET) et aux applications embarquées.

### 4.2.2.2 Choix du framework pour les couches de présentation

#### Choix entre JSF et Struts

- JSF :[2][3]
- JSF supprime la notion de servlet grâce aux managed beans et propose l'aide à la réalisation de contrôle de formulaire.
- JSF offre une mise en place beaucoup plus simple et rapide
- Faciliter l'écriture d'interface à partir d'une bibliothèque de contrôles

- Gérer automatiquement l'état HTTP entre client et serveur (en cas de Postback par exemple)
- Fournir un modèle simple pour la gestion des événements côté client et côté serveur
- Autoriser les créations ou l'enrichissement de composants graphiques utilisateur
- Utilise les managed bean
- Configuration de la navigation entre les pages
- Nombreux composants graphiques réutilisables (icefaces ,richfaces , primefaces )

- Struts :

- Installation et paramétrage difficile
- Erreur difficile à trouver (car la séparation des fichiers et des actions est à effectuer)
- ancienne technologie
- Fort couplage des actions avec le framework (les actions doivent hériter de la classe Action)
- Fort couplage avec l'API servlet

### **Choix entre PrimeFaces , RichFaces et IceFaces**

Ils sont les 3 des bibliothèques de composants. RichFaces et IceFaces ne sont utilisables qu'avec jsf 1.2 et PrimeFaces n'est utilisable qu'à partir de JSF 2.0 PrimeFaces, étant comme un jeu de composants open-source supportant Ajax, JSF 2.0, Push, est le plus développé et offre beaucoup plus de choix.[4]

#### **4.2.2.3 Choix de l'environnement de développement**

Eclipse Helios est connu comme étant l'un des environnements de développement les plus conviviaux ainsi qu'intuitifs. Il est également open source et donc accessible à tous les développeurs. Un autre avantage est qu'il est paramétrable et l'intégration des nouveaux composants est relativement simple à travers les plugins. La documentation et les tutoriels sont nombreux rendant son apprentissage encore plus facile.

### **4.2.3 Choix technologiques pour l'application Mobile**

#### **4.2.3.1 Choix de la plateforme Android**

La plateforme Android est une plateforme basée sur l'architecture Linux conçue par cinq couches comme la montre la figure :[5]



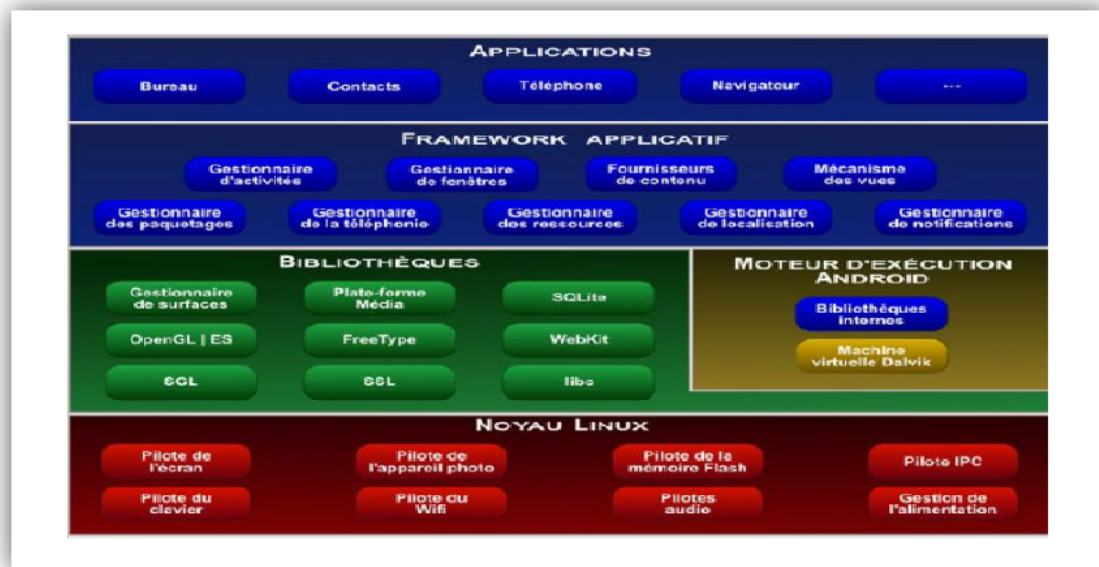


FIGURE 4.2 – Architecture de la plateforme Android

- Le noyau Linux responsable du caractère multitâches de la plateforme.
- Les bibliothèques graphiques et multimédia.
- Une machine virtuelle adaptée Java.
- Le Framework applicatif assurant la gestion des diverses composantes et fonctionnalités : fenêtres, activités, téléphonie, localisation...
- Les différentes applications : navigateur, calendrier...

#### 4.2.3.2 Choix de l'environnement de développement

NetBeans est connu aussi comme étant l'un des environnements de développement les plus conviviales ainsi qu'intuitifs. Il est également open source et donc accessible à tous les développeurs. Netbeans est un EDI multilangage et multiplateforme très performant : gestion de projets, coloration syntaxique, auto complétion,...

#### 4.2.3.3 Choix du JSON au lieu de XML

Extensible Markup Language (XML) a longtemps été une façon populaire de structurer les données en utilisant un langage de balisage familier. Il est à la fois humain et lisible par

machine et est très similaire en apparence à HTML. Il en résulte un ensemble de normes pour la communication de données sur des réseaux entre les périphériques et est itérative analysable. Un avantage de consommer des données dans ce format réside dans sa structure. Les données peuvent être ajoutées ou retirées du jeu de résultats, mais il existera souvent dans un format prévisible de sorte que vous n'avez pas à vous soucier de votre analyse de routine rupture lorsque les données changent au fil du temps. Malgré ses avantages, un inconvénient de XML est sa taille, car il contient beaucoup de caractères strictement liés à la mise en forme. Lors du téléchargement de données vers un appareil mobile, l'idéal serait de simplement obtenir le contenu pertinent à la place des données relatives à la mise en forme du contenu. C'est là qu'intervient JSON.

**JavaScript Object Notation (JSON)** est un format d'échange de données basé sur le texte issu du langage de script JavaScript. Il est formaté comme paires clé-valeur et est souvent salué comme étant une faible surcharge au format XML parce qu'il se concentre davantage sur le contenu et moins sur la forme. Cela fonctionne à notre avantage quand on veut garder les paquets d'échange de données aussi compact que possible de manière viable. Maintenant, pour être juste envers XML, il est possible de le formater de manière spécifique ou utiliser la compression pour le rendre comparable en taille à JSON, mais en général, nous trouvons JSON d'être beaucoup plus petit en taille et donc préférable de XML.[6]

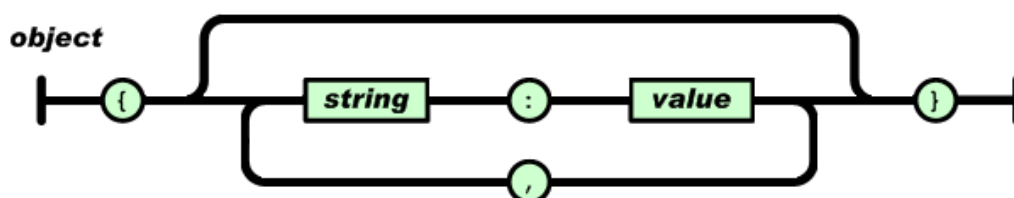


FIGURE 4.3 – Structure de JSON

JSON diffère de XML dans la notation en général. Une paire clé / valeur au format JSON ressemblerait à quelque chose comme ceci :

"clé" : "valeur"

Lorsque la notation XML équivalente serait notée comme suit :

<node> <key> valeur </ key> </ node>

Nous pouvons remarquer que dans XML, nous devons nommer le noeud pour chaque instance ainsi que chaque K / V peut être définie comme un attribut ou un objet. C'est parfois lourd et

pas nécessairement en vaut la notation supplémentaire. Cela va coûter la bande passante et le temps de traitement.

Aussi, pour représenter avec précision XML et analyser correctement, nous devrions également avoir un schéma fourni qui donnera des instructions quant à savoir si les valeurs sont des objets ou des attributs. Par rapport à JSON, ce n'est inutile en raison de la structure de sérialisation de JSON.

Après toutes les comparaisons et les mesures, cela dépend vraiment de l'ampleur de la demande et la quantité de données / objets qui sont passés entre les processus sur le Web. JSON est une solution légère pour un problème léger, son analyse est rapide, efficace et a beaucoup de différents types d'objets qui peuvent être représentés.

## **4.3 Réalisation et présentation de l'application**

Nous consacrons cette section pour la présentation des résultats obtenus suite à la phase d'implémentation. En effet, la première sous section porte sur la phase de réalisation de l'application Web alors que la deuxième sous section porte sur la phase de réalisation de l'application Mobile.

### **4.3.1 Présentation du site web**

Dans cette sous section nous présentons le travail réalisé à travers quelques prises d'écrans relatives à l'application Web qui est dédiée à nos deux acteurs : l'administrateur et le client.

- Page d'accueil :

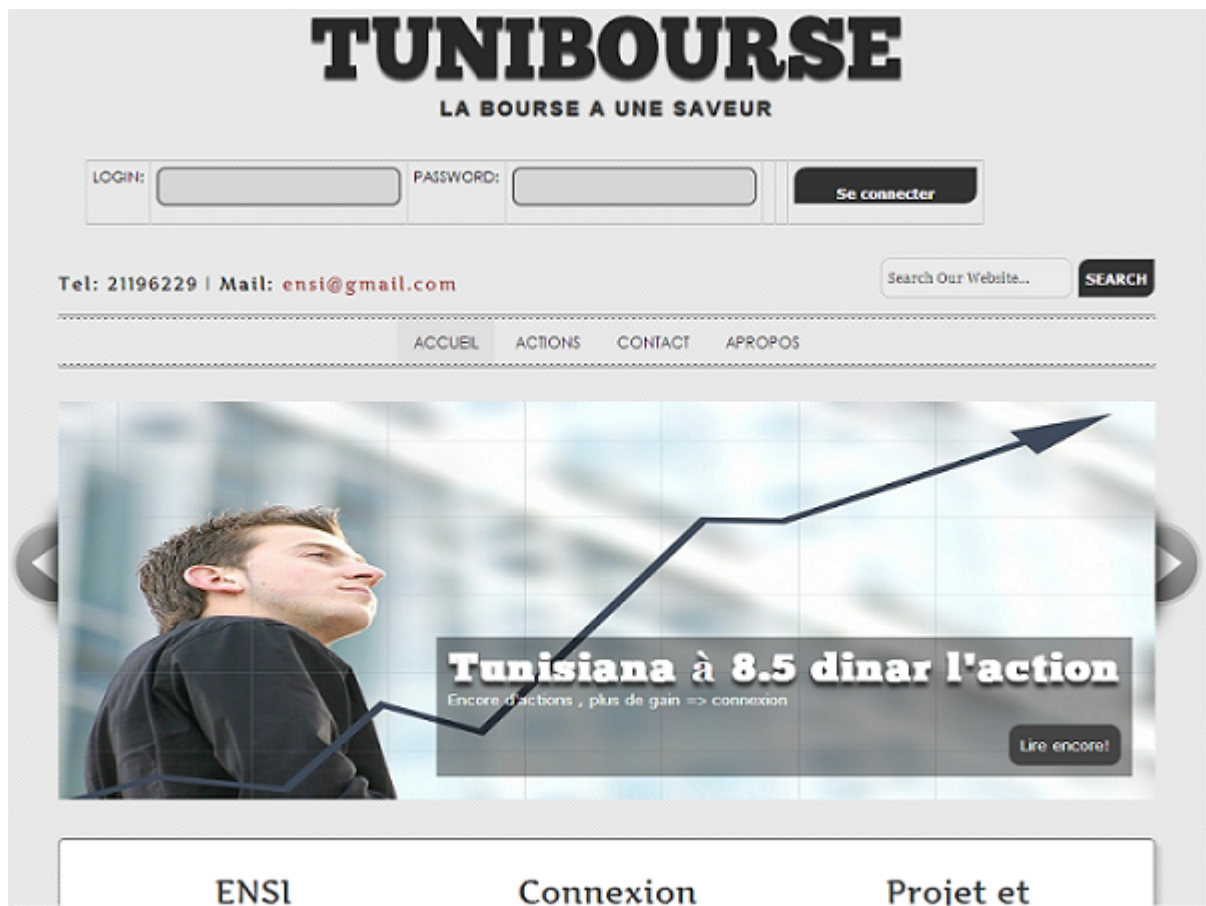


FIGURE 4.4 – Page d'accueil TuniBourse

La figure 4.4 nous présente l'espace d'accueil de notre site Web. L'utilisateur peut jouir d'une publicité concernant nos différents actions. Aussi, l'utilisateur peut cliquer sur l'un des boutons dans la barre en haut de la page pour aller à une autre page. Enfin, l'utilisateur, ayant un compte, peut se connecter en tapant son login et son mot de passe. Une fois authentifié, l'utilisateur sera redirigé selon les droits d'accès (client ou administrateur) vers son espace

- Page de l'espace client :

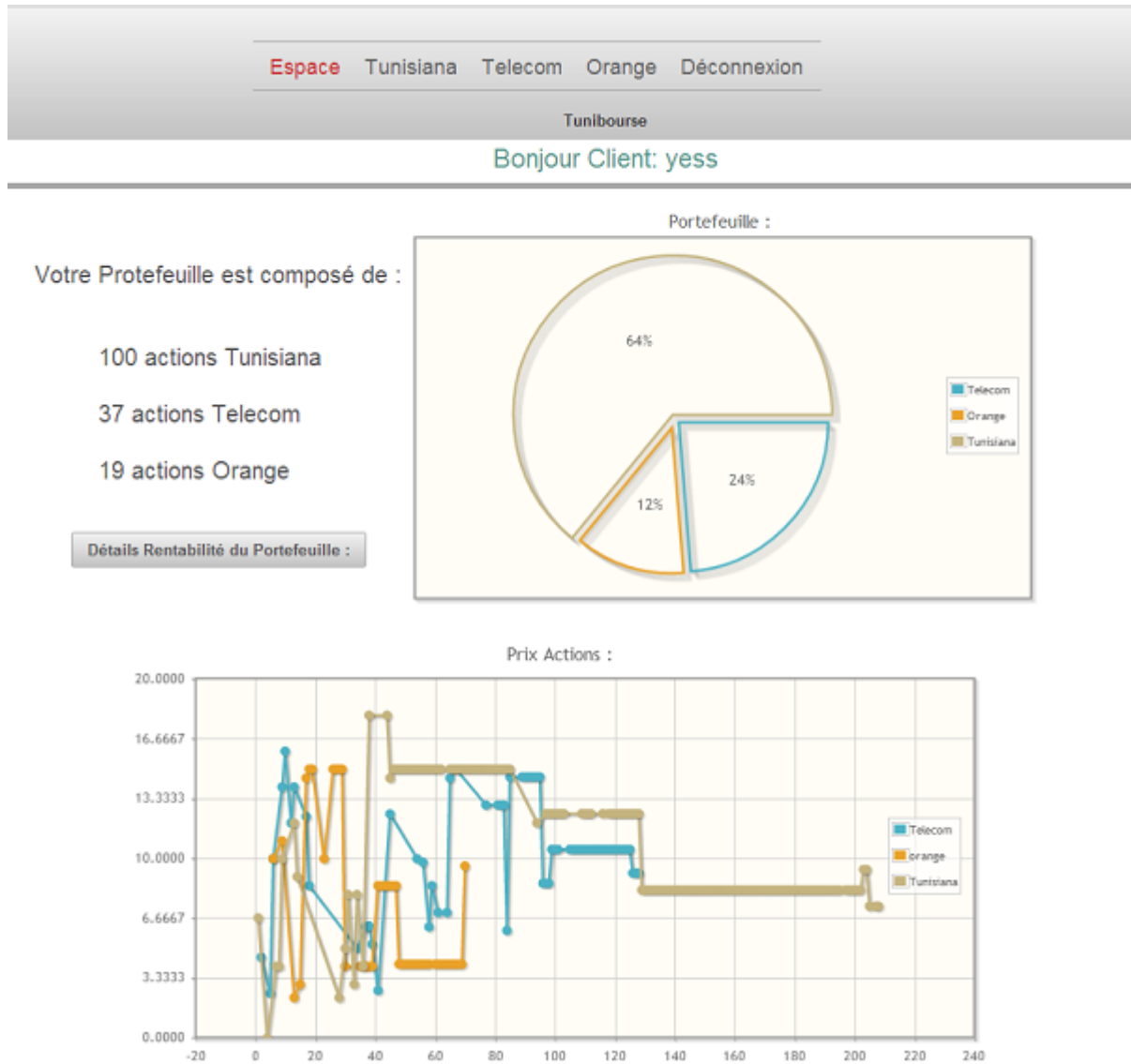


FIGURE 4.5 – Espace Client après authentification

La figure 4.5 nous présente l'espace du client déjà authentifié. Le client peut visualiser dans cette page le contenu et la rentabilité de son portefeuille comme décrit par la figure 4.6, ainsi qu'une courbe représentant la variation des prix de chaque action.

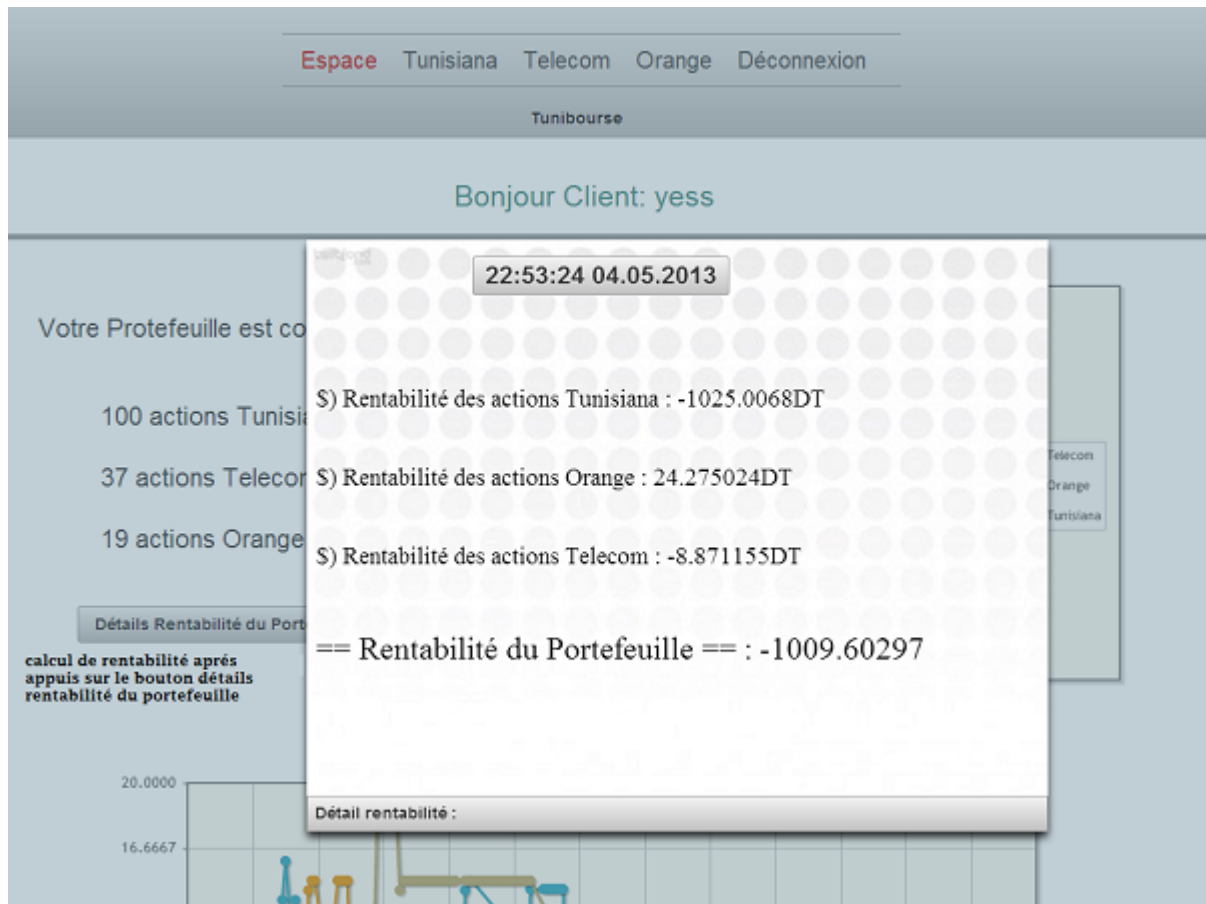


FIGURE 4.6 – Fenêtre de calcul de rentabilité

Le calcul de rentabilité se fait dans un premier temps par chaque type d'action constituant le portefeuille selon tous les achats et les ventes et ainsi nous pouvons avoir ce que le client a gagné de cette action dans le passé puis sachant le nombre restant qu'il dispose actuellement et selon le prix actuel nous calculons la valeur actuelle de son portefeuille.

Le client, à travers les boutons qui existent dans le menu de cette page, peut soit se déconnecter soit aller à une autre page relative à l'une des actions.

• Page de l'action Telecom :

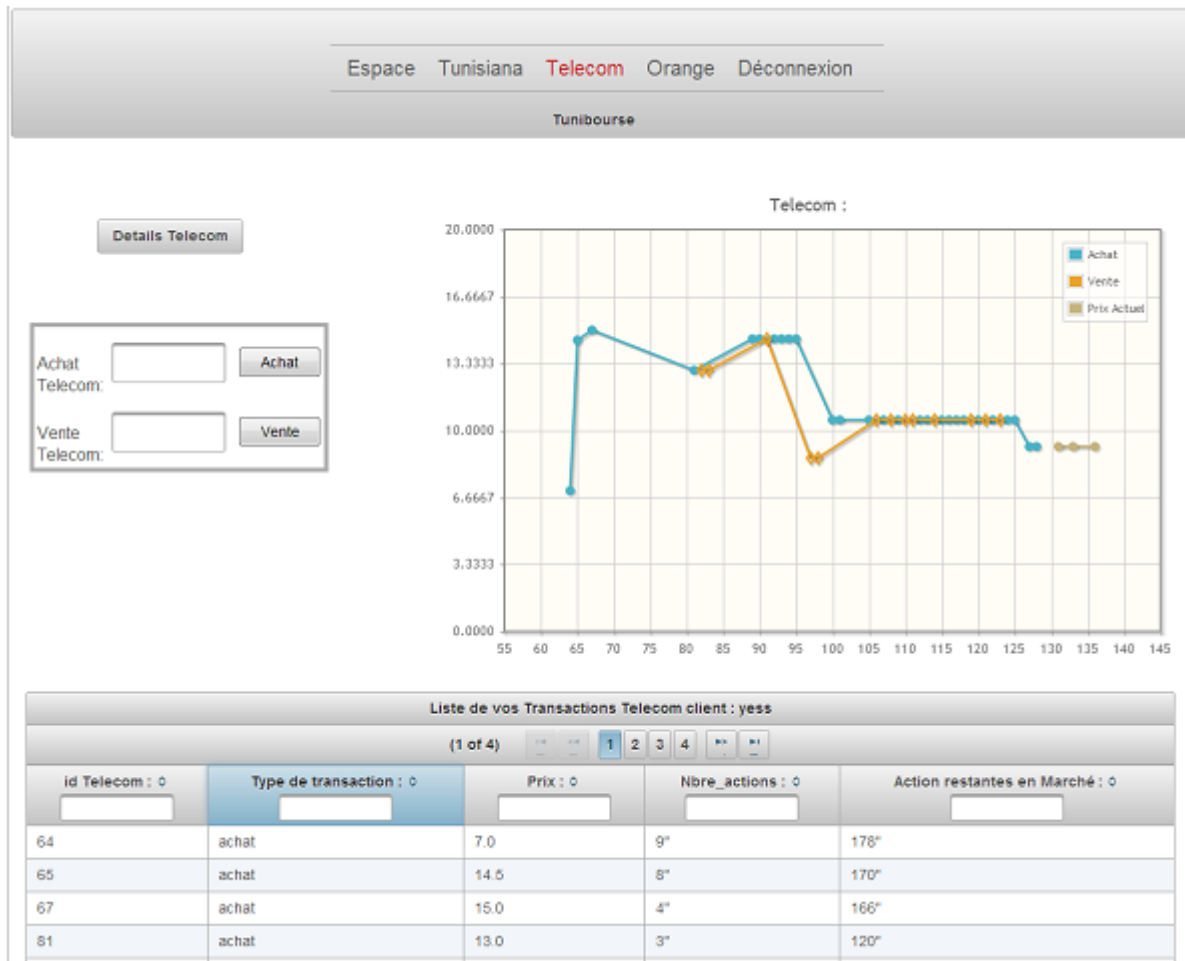


FIGURE 4.7 – Espace de l'action Telecom

La figure 4.7 nous présente l'espace de l'action Telecom. En effet, le client peut visualiser les différentes transactions d'actions Telecom qu'il a effectué soit par une courbe soit dans un tableau. Le client peut acheter ou vendre des actions Telecom juste en remplissant le formulaire existant.

• Page de la mise à jour des prix :



FIGURE 4.8 – Mise à jour du prix

Etant identifié, l'administrateur sera dirigé vers la page présentée par la figure 4.8. C'est ici que l'administrateur peut changer les prix des différentes actions en saisissons les nouveaux prix dans le tableau en bas. L'administrateur peut visualiser les changements de prix grâce à une courbe comme il peut aller à la page Utilisateurs ou se déconnecter à travers les boutons du menu en haut de la page.



- Page de l'ajout d'utilisateurs :

The screenshot shows a web interface for managing users. At the top, there is a navigation bar with three buttons: 'Actions', 'Utilisateurs' (highlighted in red), and 'Déconnexion'. Below this bar, a subtitle reads 'Ajout / Suppression Utilisateurs'.

On the left side, there is a form for adding a new user. It contains three input fields: 'Login', 'Password', and 'Type'. The 'Type' field is a dropdown menu currently set to 'client'. Below these fields is a 'Valider' button.

On the right side, there is a table titled 'Liste des Utilisateurs et Admins :'. The table has three columns: 'Login', 'Password', and 'Type'. It displays a list of users, with the first page of results shown. The table includes pagination controls at the top and bottom, indicating '(1 of 5)' records.

Login	Password	Type
abc	abc	client
admin	admin	admin
aliiiii	akjhh	client
amin	amine	client
amino	amineo	client
arbi	fgf	client
Ayedi	Ahmad	client

FIGURE 4.9 – Ajout d'un acteur

la figure 4.9 présente la page Utilisateurs. C'est ici que l'administrateur peut ajouter d'autre utilisateurs en saisissons les différents champs du formulaire. L'administrateur peut visualiser la liste des différents utilisateurs comme il peut retourner à la page Actions ou se déconnecter à travers les boutons du menu en haut de la page.

#### 4.3.2 Présentation de l'application mobile

Dans cette sous section nous présentons le travail réalisé à travers quelques prises d'écrans relatives à l'application Mobile qui n'est dédiée qu'au client.

- Page d'accueil :

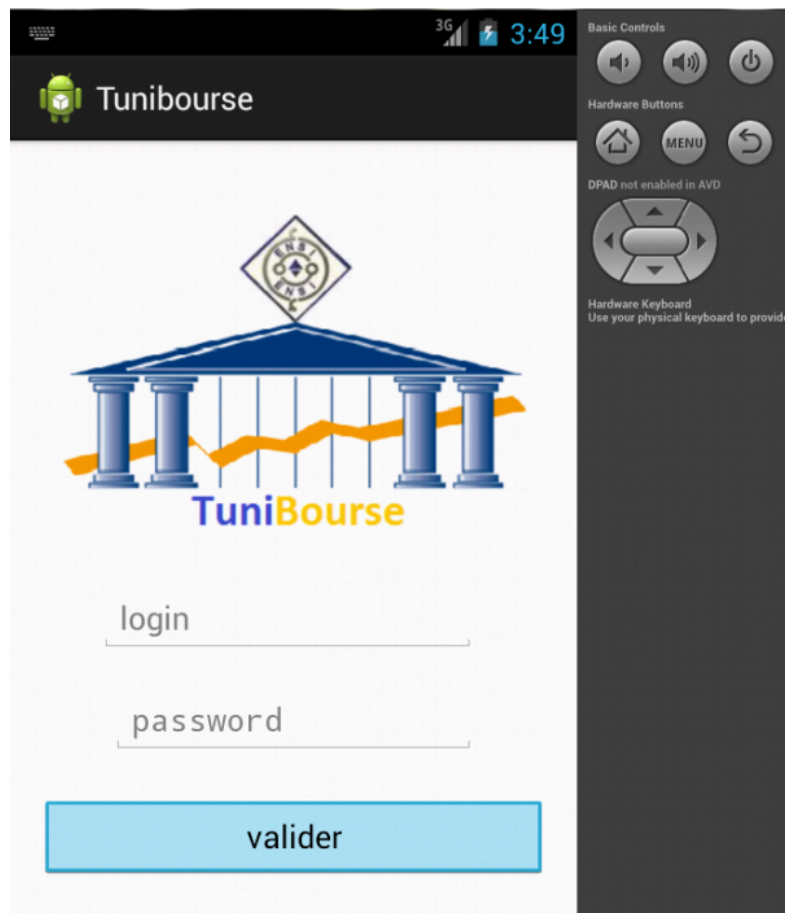


FIGURE 4.10 – Accueil

La figure 4.10 nous présente la page d'accueil de notre application Mobile. Le client doit forcément saisir le login et le mot de passe pour pouvoir accéder aux services de cette application. Une fois authentifié, le client sera dirigé vers un page spécifique à lui par un simple clique sur le bouton valider.

- Page de l'espace client :

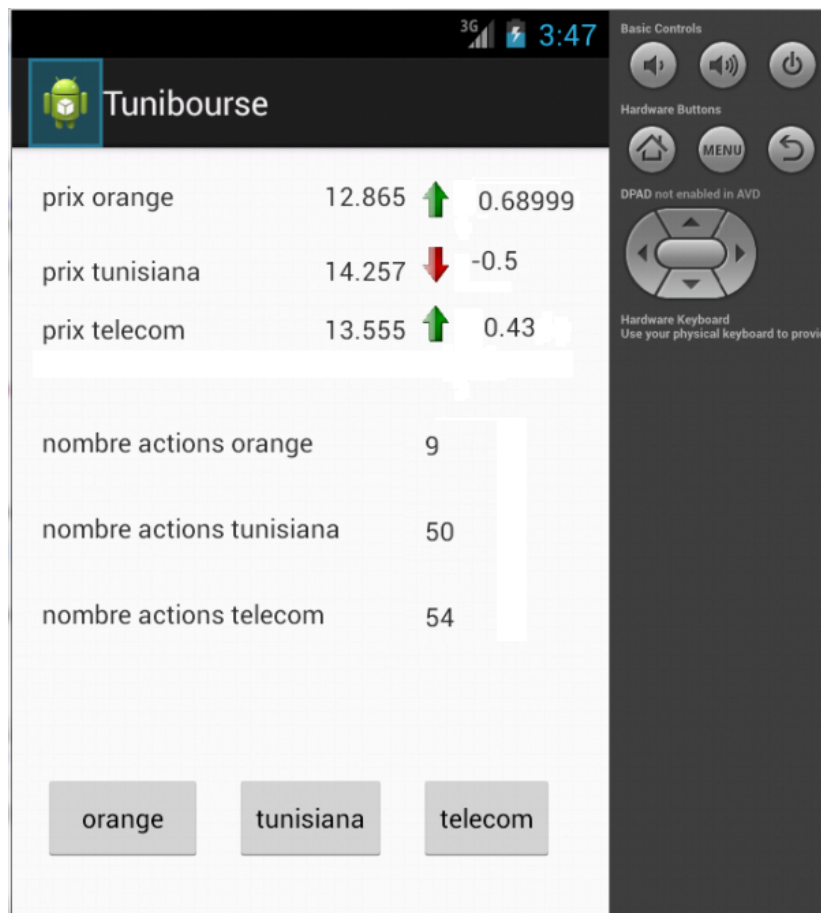


FIGURE 4.11 – Espace client

La figure 4.11 nous présente la page de l'espace du client déjà authentifié. En effet, cette page informe le client du contenu de son portefeuille comme elle lui donne la possibilité de visualiser les prix actuels et les états des différentes actions. Le client peut aller aux pages relatives aux action grâce aux boutons existants en bas.

- Page de l'achat ou vente de Tunisiana :

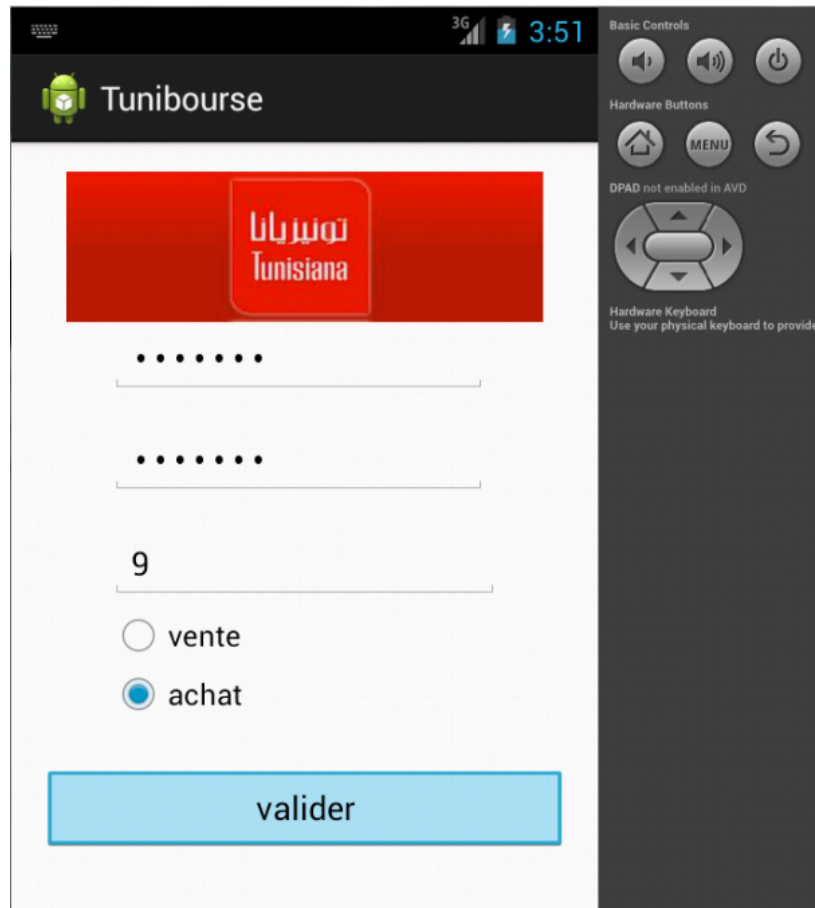


FIGURE 4.12 – Espace action Tunisiana

La figure 4.12 nous présente la page de l'action Tunisiana. Le client peut soit vendre soit acheter des actions Tunisiana et ceci en remplissant le formulaire et en validant l'achat par le bouton valider.

## Conclusion

Dans ce chapitre, nous avons expliqué, dans un premier lieu, les différents choix technologiques.

Nous avons ensuite présenté les différentes fonctionnalités de notre application détaillées par des imprimes écrans.

# Conclusion Générale

L'objectif de notre projet est de mettre en place, d'une part une application Web permettant la gestion d'un portefeuille de 'Télécommunication' interagissant avec deux acteurs : client et administrateur et d'autre part une application Mobile permettant à un client essentiellement la rapidité de sa réaction face aux changements instantanés des prix en bourse.

Afin de réaliser ces objectifs, nous avons utilisés dans nos applications une base de données dont laquelle sont stockées toutes les informations nécessaires qui nous ont permises de fournir à l'utilisateur par différents services à savoir l'achat et la vente des actions, la consultation de l'historique des transactions effectuées, la visualisation de la rentabilité du portefeuille et d'autres services qui ont rendus notre projet innovant et moderne dans le secteur boursier en Tunisie.

Durant la réalisation du projet, nous avons été confrontés à divers problèmes liés à la manipulation de nouvelles technologies et essentiellement le choix de ces derniers et aussi leur intégration dans un même environnement.

Le point fort de nos applications est l'intégration de nouveaux services dans le secteur boursier Tunisien. En effet, l'utilisateur peut facilement interagir avec son portefeuille et l'intermédiaire en bourse que ce soit avec une application Web ou aussi avec une application Mobile. Un autre point fort de notre projet est l'extensibilité du code qui peut être simplement modifié et réutilisé selon les différents besoins des intermédiaires en bourse.

Enfin, notre application peut évoluer surtout dans la partie Mobile qui pourra être plus enrichies vis à vis les services existants dédiés aux clients ainsi qu'elle pourra évoluer par l'ajout des services relatifs à l'administrateur.

# Netographie

- [1] <http://www.mkyong.com/tutorials/jax-rs-tutorials/>, 'définition de JAX-RS', 01/05/2013
- [2] <http://www.coreservlets.com/JSF-Tutorial/jsf2/>, 'tutoriel sur JSF', 07/05/2013
- [3] <http://www.mkyong.com/tutorials/jsf-2-0-tutorials/>, 'tutoriel sur JSF', 07/05/2013
- [4] <http://primefaces.org/>, 'primefaces', 08/05/2013
- [5] <http://androidmentor.com/system/24/android-platform-architecture.html>, 'l'architecture de Android', 06/05/2013
- [6] <http://www.seguetech.com/blog/2013/04/01/xml-vs-json-web-services-best-choice>, 'choix de JSON', 06/05/2013