Ministère de l'Enseignement Supérieur
et de la Recherche Scientifique
Université Virtuelle de Tunis

UVT
جامعة تونس الافتراضية
UNIVERSITÉ VIRTUELLE DE TUNIS

MEMOIRE
DE STAGE DE FIN D'ETUDE

Pour l'obtention du

MASTERE PROFESSIONNEL

« Nouvelles Technologies des Télécommunications et Réseaux »

Présenté par :

Walid Trabelsi

Titre

Création d'une application web
« KANBAN » pour les restaurateurs

Soutenu le : 08/02/2014

Devant le jury :

Mr. Ezeddine Ben Braiek Président

Mr. Hassen Seddik Membre

Mr. Jalel Khedhiri Membre

www.uvt.rnu.tn
14, Rue Yahia Ibn Omar - 1082 Mutuelleville - Tunis
Tél. : +216 71 28 99 81 / +216 71 89 17 31 - Fax : +216 71 89 26 25

Dédicaces

Je tiens à remercier monsieur Jameleddine et monsieur Riadh pour leurs précieuses assistances et leurs orientations.

Je tiens à présenter mes expressions de reconnaissance envers tous mes enseignants qui ont contribué à ma formation en Mastère N2TR et qui ont participé à l'enrichissement de ma carrière universitaire et aux membres du jury pour l'honneur qu'ils me feront en acceptant de juger ce modeste travail.

Que tous ceux qui, tant aimablement ont participé de près ou de loin à l'élaboration de ce mémoire, trouvent en ces quelques lignes un modeste témoignage d'une sincère gratitude.

TABLE DES MATIERES

LISTE DES FIGURES

INTRODUCTION GENERAL

Durant ces dernières années l'informatique s'est imposée d'une manière très impressionnante dans les entreprises, cela est dû à son apport extraordinaire dans la gestion de leurs systèmes d'informations et la mise en application de certaines méthodes de fabrication ou de contrôle citons à titre d'exemple la fameuse méthode japonaise « kanban ».

« Kanban » (カンバン ou 看板) est un terme japonais signifiant « regarder le tableau » cette méthode, déployée à la fin des années 50 dans les usines Toyota, est mise en place entre deux postes de travail et limite la production du poste amont aux besoins exacts du poste aval. Cette méthode est surtout adaptée aux entreprises ayant une production répétitive et relativement régulière.

Le principe de kanban est basé sur son nombre en circulation qui doit être limité pour éviter la constitution d'encours trop importants. La méthode kanban ne dispense pas cependant d'établir des prévisions de vente et un programme de production détaillé à moyen terme. C'est en effet une technique de gestion de la production à court terme et elle peut s'intégrer parfaitement dans la gestion non pas seulement industrielle mais aussi la gestion de données professionnelles y est compris celui de la gestion de données restaurateurs auquel nous rattacherons d'ailleurs notre étude, et cela pour une meilleure gestion des différents traitements exigés selon cette méthode des cartes.

Nous avons constaté, en effet, pendant nos recherches que l'ensemble des traitements au sein du restaurant se fait manuellement, ce qui engendre un certain nombre de problèmes tels que la lenteur dans l'accès aux données et le risque de perte d'informations ; donc la meilleure solution pour pallier aux problèmes de gestion de ces activités est l'informatisation afin d'assurer l'accès instantané aux données et sécuriser ces dernières, ce qui simplifie le travail administratif.

De ce fait, il nous a été sollicité par les restaurateurs afin de leur concevoir un système d'information automatisé pour la gestion de données et pour réaliser cette tâche, notre choix s'est porté sur le Processus Unifié. En effet, le processus unifié est une solution de développement logiciel adaptée à tout type de projet. Ses traits distinctifs tiennent en trois notions : piloté par les

cas d'utilisation, centré sur l'architecture, itératif et incrémental.

Pour l'implémentation, le choix du langage de programmation a été dicté par le type de l'application qui devrait être réalisée d'une part, et être accessible via le réseau Internet à la portée de tout le monde, selon la technologie Client/serveur qui est largement utilisée de nos jours, et qui constitue une évolution des systèmes classiques et permet par l'utilisation de nouvelles méthodes et techniques plus flexibles et fiables.

Notre travail consiste à concevoir une application qui permet de gérer le système d'informations relative au restaurateur : contact, réservation, tables...et pour atteindre notre objectif nous avons partagé le travail comme suit :

Dans le premier chapitre analyse et spécification, on s'intéressera à la capture des besoins qui consiste à l'élaboration d'un bilan des besoins et l'analyse architecturale du système à concevoir ainsi de déterminer les aspects généraux de l'application en identifiant les principales composantes du système. La conception est présentée dans le deuxième chapitre elle est axée sur les méthodes de conception de l'application sous forme d'un ensemble de diagrammes ainsi que la conception de la base de données. Le dernier chapitre qui est l'implémentation évoque les différentes techniques, outils et environnements de développement ainsi que de représenter quelques interfaces utilisateurs de l'application.

CHAPITRE 1 : ANALYSE ET SPECIFICATION DES BESOINS

Ce n'est pas la conscience des hommes La valeur d'un homme tient dans sa capacité à donner et non dans sa capacité à recevoir. « Albert Einstein »

INTRODUCTION

L'objet de ce chapitre est de développer un modèle du système à construire, un système étant un tout constitué d'éléments munis de propriétés unis par des relations. Ce modèle est élaboré suite à des demandes des utilisateurs. À ce niveau, le rôle du concepteur est restreint à la fonction d'analyste. A ce stade, nous devons recenser les besoins, éventuellement critiquer la démarche d'élaboration de statistiques actuelles, comprendre le contexte du système, formuler les besoins fonctionnels et non fonctionnels (autour du système).

Il s'agit d'une étape cruciale dans la réalisation d'une application donnée. En général, l'objectif de l'analyse est d'arriver à la compréhension des besoins et des exigences du cahier de charges. Il s'agit de livrer des spécifications pour permettre de choisir la conception de la solution. Cette partie permet donc d'expliciter en clair les grands traits et spécifications des parties fondamentales de l'application à développer.

I. OBJECTIF

Ce projet consiste à développer une application web en J2EE pour la gestion des tâches quotidiennes personnelles et professionnelles d'un restaurateur en utilisant comme langage de programmation JSF « ***JAVA SERVER FACES*** » et API JPA comme outil de persistance des données avec une base de données MySQL.

II. CADRE DE PROJET

Hra. performance engineering est une entreprise franco-tunisienne d'édition de sites web thématiques et de développement d'applications en ligne pour ses clients internationaux. Elle mène aussi d'autres activités dans le domaine de marketing numérique et le référencement web. Elle est actuellement une start-up en pleine expansion, gérée par des compétences jeunes et sous la direction d'un promoteur français.

Les activités de **HRA** performance sont très variées, nous pouvons en mentionner quelques-unes

- Conception et développement des sites web
- Conception, sécurisation et audit de réseaux.

- Conception et mise en place des bases de données.
- Développement des applications.
- Rédaction des cahiers de charges.

III. CONTEXTE DU SYSTEME

Comme nous l'avons cité précédemment, BNS Engineering n'est pas une société consommatrice, elle est plutôt productrice. Toutes les applications développées sont destinées alors à la vente.

L'application en question s'agit d'une application de gestion des tâches quotidiennes personnelles et professionnelles de son utilisateur dédiée au restaurateur. Cette solution doit être capable d'organiser et structurer les données et les réservations qui sont faites à l'heure actuelle manuellement.

1. CRITIQUES ET INSUFFISANCES

La solution actuelle est manuelle :

- L'absence d'un système pour suivre les services proposés et les réservations en temps réel.
- Un besoin d'un nombre additionnel d'employés pour gérer les différentes tâches administratives.
- Risque de perdre des contacts (fournisseur, client): ce qui peut être fatal sur le déroulement du travail au restaurant.
- Le suivi des clients et des fournisseurs peut rencontrer beaucoup de problèmes.
- La perte de clientèle suite à la bureaucratie du système de réservation par téléphone.

IV. CAHIER DES CHARGES

Ce projet de fin d'études vise à mettre le point sur des tâches bien précises relatives à la gestion des données au profit des restaurateurs.

En effet, les tâches sont les suivantes :

- Chaque nouveau utilisateur est appelé à s'inscrire dans cette application et enregistrer les données relatives à son profil.
- Pour chaque utilisateur inscrit, un compte d'accès lui sera attribué afin qu'il puisse s'authentifier et utiliser les différents services de cette application.
- Suivi du planning de jour : les tâches journalières et les notes personnelles dans un agenda.
- Ajout des notes de rappel sous forme d'étiquettes à afficher sur l'agenda.
- Création des cartes de contacts client et fournisseur : ajout et suppression, modification des informations.
- Suivi des réservations et envoi des emails automatiques aux clients.
- Gestion du plan de tables tout en donnant la possibilité d'ajouter et modifier les données relatives à chaque table que ce soit son numéro, le nombre de personnes, service, fumeur ou non.
- Ajout un menu de jour et le publier en envoyant un email pour les utilisateurs de ce système.

V. IDENTIFICATION DES BESOINS

2. LES BESOINS FONCTIONNELS

- L'application conçue devra fonctionner en mode 3 – tiers (client léger, serveur de données, serveur d'application).
- Chaque restaurateur possédant un identifiant unique pour accéder à cette application.
- Le restaurateur peut ajouter, supprimer, modifier ses données de profils, gérer ses contacts client, fournisseurs, les affecter au répertoire bien déterminé ou à un sous-groupe de ce répertoire.
- Afficher au planning récap. du jour les réservations déjeuné, dîner.
- Créer un ou plusieurs plans de tables de salle de restaurant.
- Gestion du menu du jour : créer un nouveau menu, attribuer une date, adresser les menus aux clients.
- Journaliser les avertissements, alertes, erreurs, bugs générés et qui peuvent être utiles au débogage.

3. LES BESOINS NON FONCTIONNELS :

- L'application doit permettre de gérer les accès des utilisateurs selon un privilège et un état d'activation de chaque compte.
- Il faut garantir la sécurité d'accès à l'espace administrateur afin de protéger les données personnelles des utilisateurs.
- Prévenir contre l'accès direct avec les liens URL et définir un délai de temps pour la fermeture de session non active.
- L'interface de cette application doit être ergonome, conviviale et voire même apte à aider l'utilisateur à mieux gérer son espace de travail.

VI. RECHERCHE DES ACTEURS ET DES CAS D'UTILISATION

1. LES ACTEURS

Les acteurs n'appartiennent pas au système, mais ils interagissent avec celui-ci. Ils fournissent de l'information en entrée et/ou reçoivent de l'information en sortie, ils ne sont pas forcément des personnes.

En UML, il n'est pas à exclure la possibilité de spécialiser des acteurs à partir d'autres acteurs.

En ce qui concerne notre système, nous avons classé les acteurs en deux catégories.

Acteurs internes	Acteurs externes
Restaurateurs	client

FIGURE 1 - TABLEAU DES ACTEURS

2. LES CAS D'UTILISATION

Pour cette application, les cas d'utilisation les plus pertinents sont les suivants :

Les cas d'utilisation	Priorité

Authentification	1
Afficher le planning du jour	2
Suivie des réservations	2
Gérer le plan des tables	3
Gestion du contact client	3
Gestion du contact fournisseur	3

FIGURE 2 - TABLEAU DES CAS D'UTILISATION

VII. RAFFINEMENT DES CAS D'UTILISATION

1. CAS D'UTILISATION AUTHENTIFICATION :

Le restaurateur ou le gestionnaire se connecte au système et saisit son login et mot de passe. Le système vérifie le mot de passe introduit. Si ce dernier est correct, l'accès est autorisé et une nouvelle session sera établie.

2. CAS D'UTILISATION AFFICHAGE DE PLANNING DU JOUR

Il s'agit d'un agenda récapitulatif de notes personnelles et professionnelles du restaurateur et de lui mettre en place un suivi régulier de:

- Ses rendez-vous professionnels et personnels
- Notes « «pense bête »
- Réservations clientèles,
- Les dates marquantes pour l'utilisateur de cette application (fête, date de naissance...)

Par ailleurs il est possible de modifier l'ensemble de ces données et les supprimer directement à partir de cet agenda et avoir une alerte automatique sous forme pop-up pour informer l'utilisateur.

3. CAS D'UTILISATION SUIVI DES RESERVATIONS

Ce cas d'utilisation offre la possibilité de suivre en temps réel les réservations des tables. Après chaque réservation passée par les clients ou le restaurateur, une note sera affichée dans le tableau des tables indiquant le nom du client et l'heure de son arrivée. Lorsque l'heure limite sera dépassée, un mot dépassé devra s'afficher automatiquement à côté de l'heure limite. Comme il

est possible d'annuler ou modifier une ou plusieurs réservations enregistrées dans ce tableau de suivi.

4. CAS D'UTILISATION GESTION DE PLAN DES TABLES

Chaque restaurant dispose d'une salle de restaurateur ou une terrasse composée de quelques tables pour 2, 4,8 personnes disposées selon un plan bien défini suivant les services offerts et le type des clients convoités. Avec ce plan, il est possible de suivre en détail la disponibilité et les dispositions de ces tables. Comme il est possible d'ajouter ou supprimer une table d'un plan donné.

5. CAS D'UTILISATION GESTION DES CONTACTS

Chaque restaurateur doté d'un carnet de contacts a la possibilité d'ajouter, supprimer ou modifier des contacts clients ou fournisseur et de mener des recherches sur ces derniers selon un critère bien défini.

VIII. LE PLAN QUALITE PROJET (PQP) :

Le Plan Qualité de Projet (ou PQP) a pour objectif la définition et le suivi des dispositions à prendre dans le cadre du projet afin d'en assurer la qualité et d'atteindre les résultats attendus.

1. PLANIFICATION DES TACHES : DIAGRAMME DE GANTT :

Le diagramme de GANTT est un outil permettant de modéliser la planification de tâches nécessaires à la réalisation d'un projet.

Pour arriver à réaliser ce projet, nous avons suivi le planning suivant :

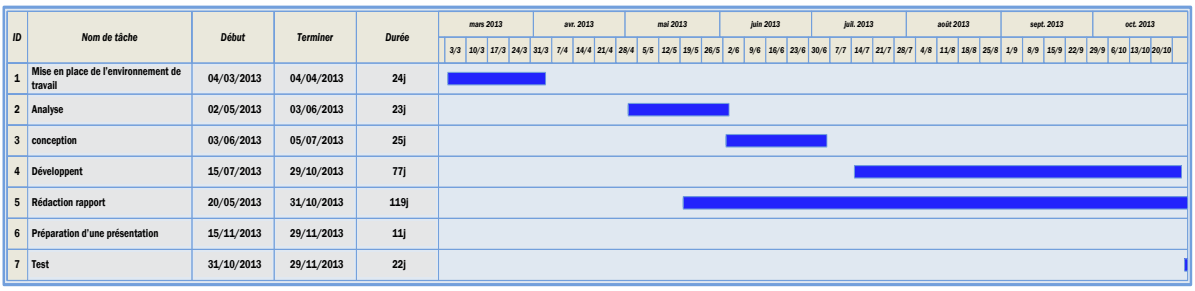

FIGURE 3 - DIAGRAMME DE GANTT

2. QUALITE : MODELE DE CYCLE DE VIE:

Le « cycle de vie d'un logiciel » désigne toutes les étapes du développement d'un logiciel, de sa conception à sa disparition. L'objectif d'un tel découpage est de permettre de définir des jalons intermédiaires permettant la validation du développement logiciel, c'est-à-dire la conformité du logiciel avec les besoins exprimés Il existe une variété de ces modèles : en cascade, W,Y.

A. MODELE DE CYCLE DE VIE EN SPIRALE

Proposé par B. Boehm en 1988, ce modèle est beaucoup plus général que le précédent. Il met l'accent sur l'activité d'analyse des risques : chaque cycle de la spirale se déroule en quatre phases :

- détermination, à partir des résultats des cycles précédents, ou de l'analyse préliminaire des besoins, des objectifs du cycle, des alternatives pour les atteindre et des contraintes.
- Analyse des risques, évaluation des alternatives et, éventuellement maquettage.
- Développement et vérification de la solution retenue, un modèle « classique » (Cascade ou en V)

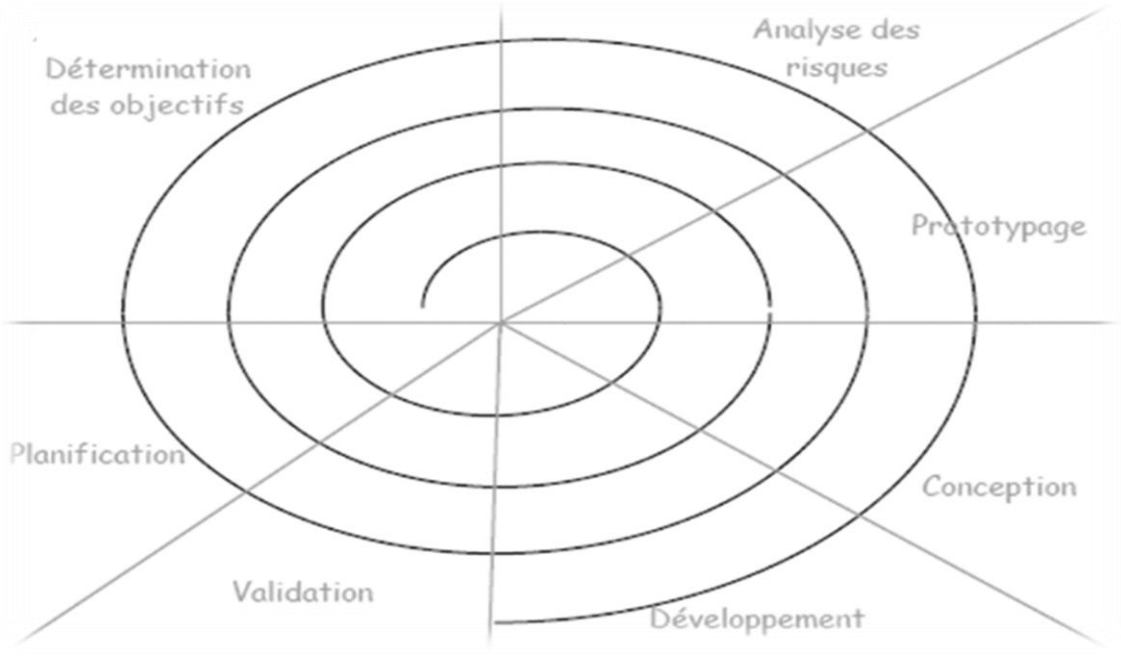

FIGURE 4 - CYCLE DE VIE EN SPIRALE

B. LE MODELE DE CYCLE DE VIE EN CASCADE

Le modèle de cycle de vie en cascade a été mis au point dès 1966, puis formalisé aux alentours de 1970. Dans ce modèle le principe est très simple : chaque phase se termine à une date précise par la production de certains documents ou logiciels. Les résultats sont définis sur la base des interactions entre étapes, ils sont soumis à une revue approfondie et on ne passe à la phase suivante que s'ils sont jugés satisfaisants. Le modèle original ne comportait pas de possibilité de retour en arrière. Celle-ci a été rajoutée ultérieurement sur la base qu'une étape ne remet en cause que l'étape précédente, ce qui est dans la pratique s'avère insuffisant.

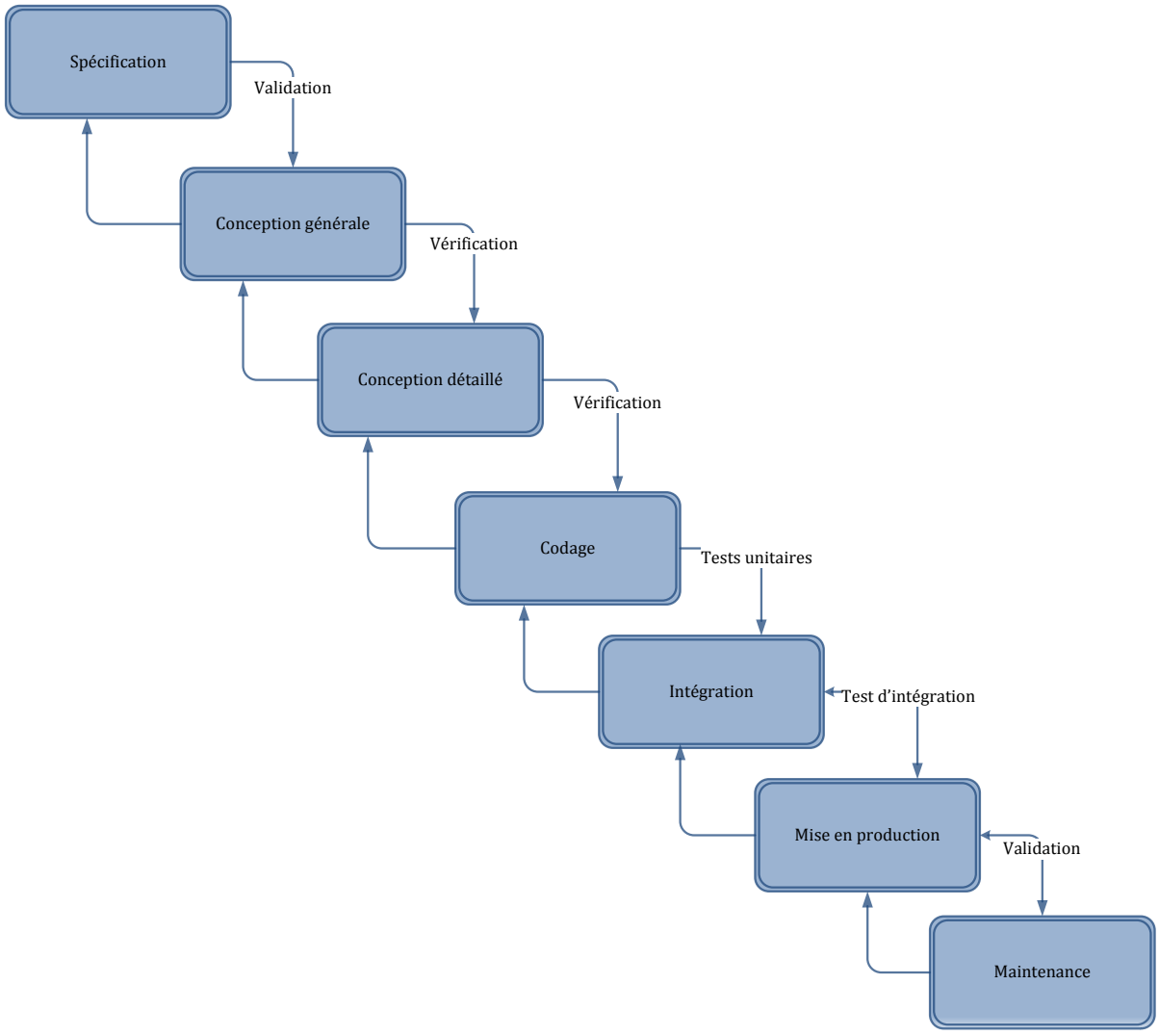

FIGURE 5 - CYCLE DE VIE EN CASCADE

C. LE MODELE DE CYCLE DE VIE EN V :

Le modèle en V demeure actuellement le cycle de vie le plus connu et certainement le plus utilisé. Le principe de ce modèle est qu'avec toute décomposition doit être décrite la recomposition, et que toute description d'un composant doit être accompagnée de tests qui permettront de s'assurer qu'il correspond à sa description.

Ceci rend explicite la préparation des dernières phases (validation-vérification) par les premières (construction du logiciel), et permet ainsi d'éviter un écueil bien connu de la spécification du logiciel : énoncer une propriété qu'il est impossible de vérifier objectivement après la réalisation.

D. CHOIX CYCLE DE VIE

La méthode adéquate que nous avons mis en œuvre dans ce travail est le modèle en V .Ce modèle est plus récent et plus proche de la réalité de l'articulation entre les activités de spécification et de conception, avec celles de validation et vérification. En effet, contrairement au modèle de cascade, ce modèle fait apparaître le fait que le début du processus de développement conditionne ses dernières étapes :

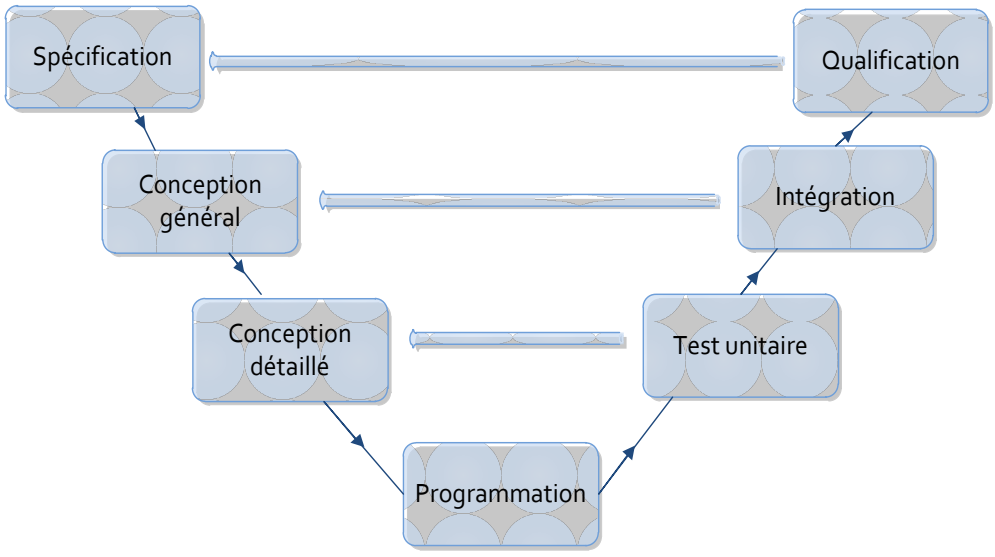

FIGURE 6 - CYCLE DE VIE EN V

CONCLUSION

Nous avons tenté dans cette partie d'analyser l'existant et d'extraire les objectifs attendus. Par la suite, nous avons procédé à l'expression des besoins pour couronner l'identification des principaux acteurs de notre système et les cas d'utilisation relatifs à chacun d'entre eux. Avec ce que nous avons pu dégager, l'application que nous devrions réaliser est devenue claire. Par conséquent, nous essayerons dans les étapes suivantes de répondre à ces besoins en entamant la phase de conception dans le chapitre suivant.

CHAPITRE 2 : CONCEPTION

C'est la qualité de l'œuvre qui doit porter et légitimer la technologie et non l'inverse. « Jean Zeitoun »

INTRODUCTION

La conception permet d'acquérir une compréhension approfondie des contraintes liées au langage de programmation, à l'utilisation des composants et au système d'exploitation. Elle détermine entre autres les principales interfaces et les transcrit à l'aide d'une notation commune en donnant un point de départ à l'implémentation.

I. CONCEPTION GENERALE :

1. ETUDE METHODOLOGIQUE

La conception d'un système d'information n'est pas évidente car il faut réfléchir à l'ensemble de l'organisation que l'on doit mettre en place. La phase de conception nécessite des méthodes permettant de mettre en place un modèle sur lequel on va s'appuyer.

E. METHODE MERISE

Merise étant une méthode de conception et de développement de système d'information daté de 1978-1979, il est basé sur la séparation des données et des traitements à effectuer en plusieurs modèles conceptuels et organisationnels, physiques.

Niveau	Statique (données)	Dynamique (traitements)	
Conceptuel	MCD	MCT	Indépendant du système: QUOI ?
Organisationnel ou logique	MLD (OU ?)	MOT (QUI ? QUAND ?)	Choix du SGBD: QUI?QUAND? OU ?
Opérationnel ou physique	MPD	MOPT	Haute connaissance du SGBD:COMMENT ?

FIGURE 7- TABLEAU RECAPITULATIF DES MODELES MERISE

F. PROCESSUS UNIFIE(UP) :

Le processus unifié est un processus de développement d'une application itérative, centré sur l'architecture, piloté par des cas d'utilisation et orienté vers la diminution des risques, il regroupe les activités à mener pour transformer les besoins d'un utilisateur en système logiciel.

C'est une méthode de prise en charge du cycle de vie d'un logiciel et donc du développement, pour les logiciels orientés objets. C'est une méthode générique, itérative et incrémentale, contrairement à la méthode séquentielle Merise.

L'objectif d'un processus unifié est de maîtriser la complexité des projets informatiques en diminuant les risques. En répondant aux préoccupations suivantes :

QUI participe au projet ?

QUOI, qu'est-ce qui est produit durant le projet ?

COMMENT doit-il être réalisé ?

QUAND est réalisé chaque livrable ?

2. LE CHOIX DE LA METHODE

Pour la réalisation de cette application, notre choix a été porté sur le Processus Unifié. En effet, le processus unifié est une solution de développement logiciel adaptée à tout type de projet. Ses traits distinctifs tiennent compte de trois notions : piloté par les cas d'utilisation, centré sur l'architecture, itératif et incrémental.

Le langage de modélisation que nous avons utilisé est UML (Unifier Modeling Language), qui est une partie intégrante de la démarche UP.

Ses diagrammes sont largement utilisés dans chaque étape et phase de ce processus de développement, il est idéal pour :

- Concevoir et déployer une architecture logicielle développée dans un langage objet (Java, C++, VB.net). Certes UML, dans sa volonté "unificatrice" a proposé des formalismes.
- Pour modéliser les données (le modèle de classe réduit sans méthodes et stéréotypé en entités), mais avec des lacunes que ne présentait pas l'entité relation de Merise.

- Pour modéliser le fonctionnement métier (le diagramme d'activité et de cas d'utilisation) qui sont des formalismes très anciens qu'avait, en son temps, amélioré Merise...
- Pour l'implémentation, le choix s'est porté sur le langage de programmation JSP/JSF implémenté dans un environnement J2EE avec une base de données MySQL.

3. ARCHITECTURE MODELE/VUE/CONTROLEUR

L'organisation d'une interface graphique est délicate. L'architecture MVC ne prétend pas à éliminer tous les problèmes, mais fournit une première approche pour le faire. Offrant un cadre normalisé pour structurer une application, elle facilite aussi le dialogue entre les concepteurs.

L'idée est de bien séparer les données, la présentation et les traitements. Il en résulte les trois parties énumérées plus haut : le modèle, la vue et le contrôleur.

Le modèle représente le cœur (algorithmique) de l'application : traitements des données, interactions avec la base de données, etc. Il décrit les données manipulées par l'application

Il est représenté dans ce projet sous forme des classes de l'api JPA, JDBC.

La vue, ce avec quoi l'utilisateur interagit. Sa première tâche est de présenter les résultats renvoyés par le modèle. Elle se contente d'afficher les résultats des traitements effectués par le modèle et d'interagir avec l'utilisateur.

Elle est représentée sous forme de pages JSP/XHTML / Primeface.

Le contrôleur prend en charge la gestion des événements de synchronisation pour mettre à jour la vue ou le modèle et les synchroniser

Il est représenté sous forme de classes contrôles : servlets

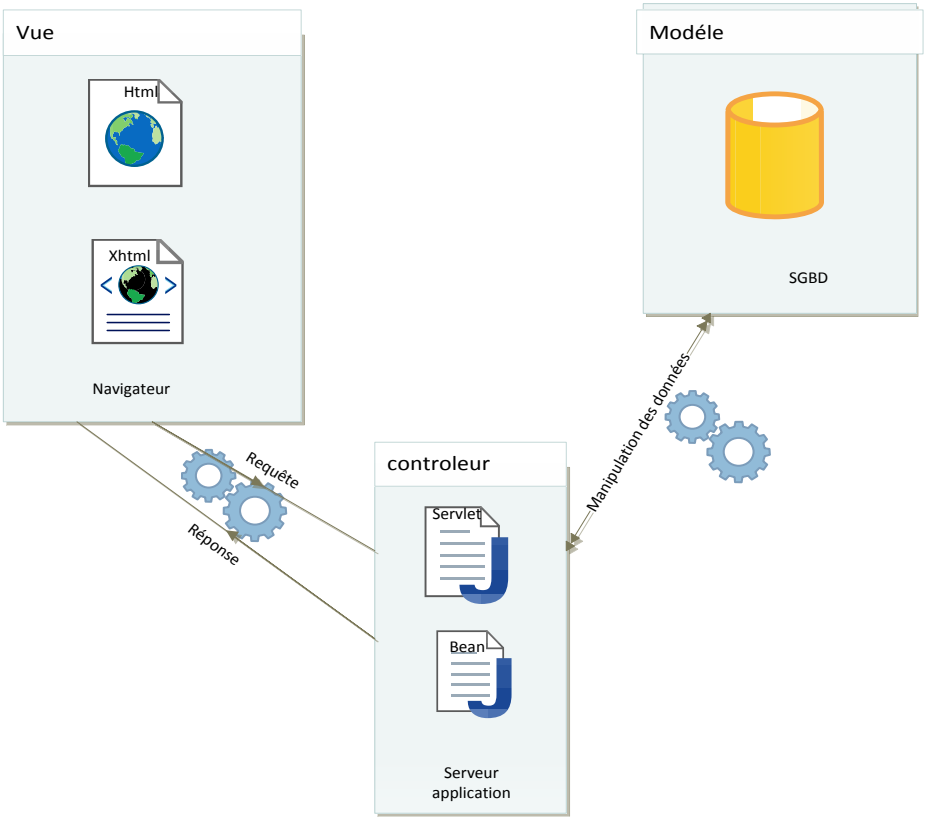

FIGURE 8 - SCHEMA MODELE MVC

II. CONCEPTION DETAILLEE

Après la conception et l'étude préliminaire des différentes fonctionnalités de cette application, nous entreprenons la modélisation des différentes activités sous forme de diagrammes UML dans ses deux aspects dynamique et statique.

1. LA MODELISATION DYNAMIQUE :

UML2 possède treize diagrammes. A la catégorie dynamique à elle seule sont associés huit diagrammes définissant le cycle de vie des objets en précisant : le comportement des objets, les différents états par lesquels ils passent et les événements qui déclenchent ces changements d'états. Dans cette application, nous allons se servir des deux seulement énoncés ci-dessus. Nous ne pouvons pas aller directement à la conception sans faire une petite description du fonctionnement de l'application.

A. DIAGRAMME DE SEQUENCES

Le diagramme de séquence vise essentiellement à représenter des interactions entre les entités

d’un système selon un point de vue temporel. Il possède une dimension temporelle mais ne représente pas explicitement les liens entre les objets. Il permet de visualiser les messages par une lecture de haut en bas. Un diagramme de séquence possède deux axes : l’axe vertical qui représente le temps, et l’axe horizontal qui représente les objets qui se collaborent. Dans la partie qui suit, nous allons exposer quelques scénarios illustrant des cas d’utilisation.

1. DIAGRAMME DE SEQUENCES « AUTHENTIFICATION »

Pour assurer la sécurité, une opération d’authentification est obligatoire. Cette opération se déclenche une fois que l’utilisateur aurait inscrit à cette application. Un formulaire lui sera envoyé, il le remplit en saisissant son login et son mot de passe fournis par l’administrateur, puis le système vérifie la disponibilité des informations. Après une vérification et une validation, le système autorise l’accès et retourne la page d’accueil destinée à l’utilisateur.

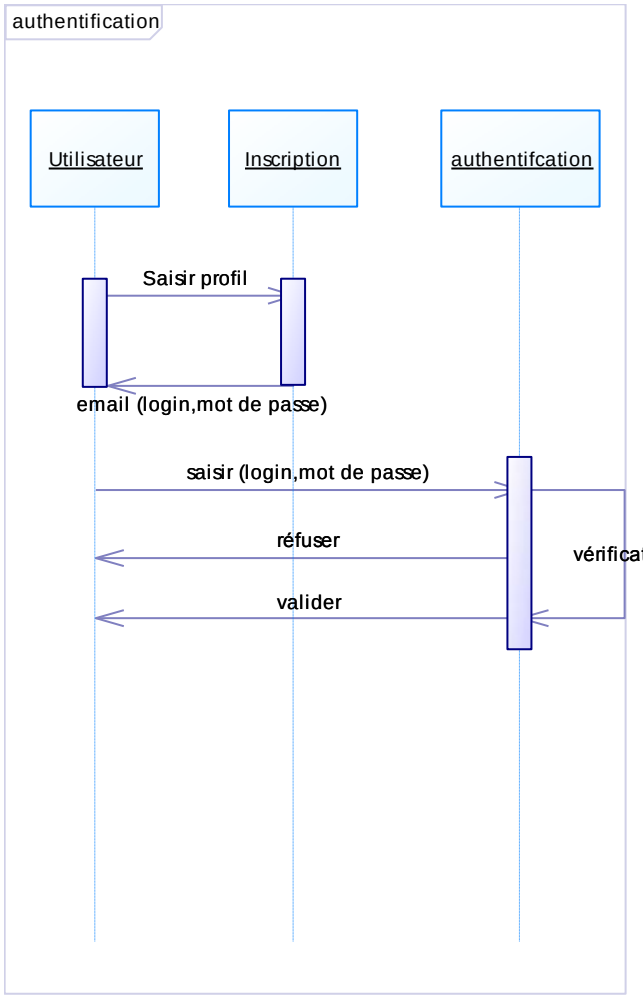

FIGURE 9 - DIAGRAMME DE SEQUENCES « AUTHENTIFICATION »

2. DIAGRAMME DE SEQUENCES « PLANNING DE JOUR»

Après la phase d'authentification, une page s'affiche contenant un agenda recensant les événements relatifs à son utilisateur tout en lui donnant la possibilité de modifier et ajouter d'autres annotations dans cet agenda .Cet agenda est utilisé afin de pouvoir donner à son utilisateur la possibilité de planifier, de noter son emploi du temps, ses rendez-vous.

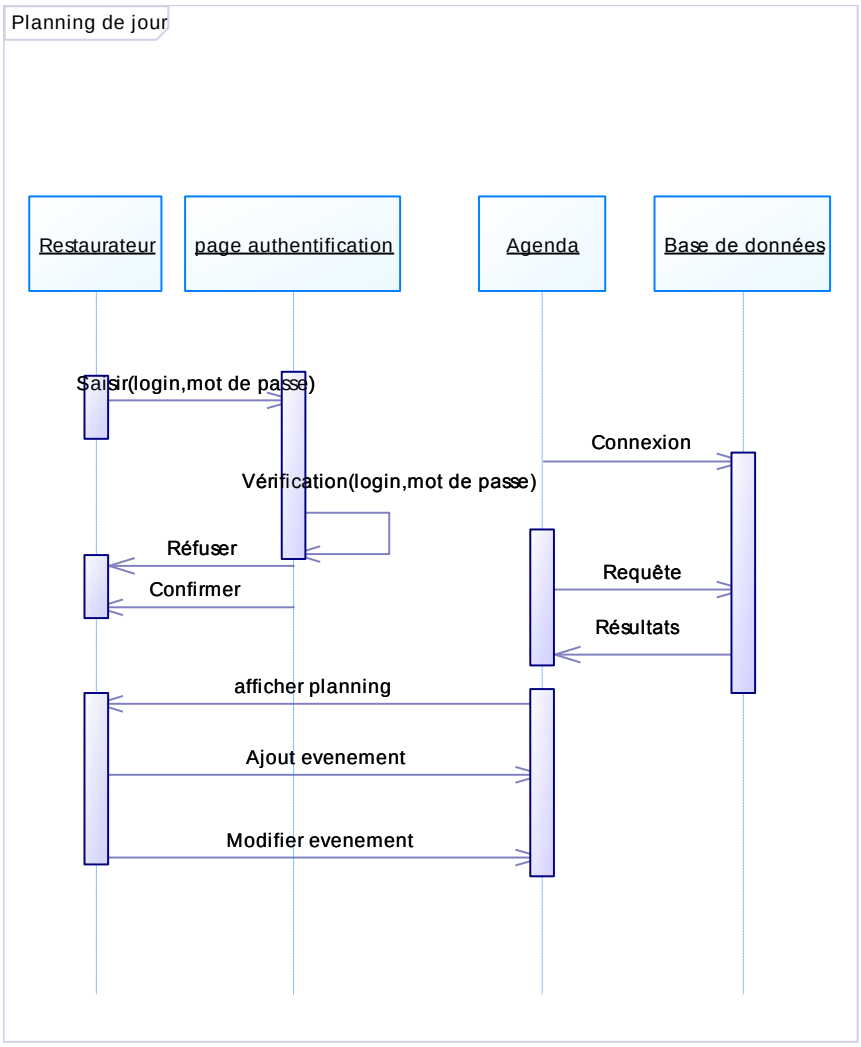

FIGURE 10 - DIAGRAMMES DE SEQUENCES « PLANNING DU JOUR »

3. DIAGRAMME DES CAS D'UTILISATION

Le but de ces diagrammes et d'avoir une vision globale sur les interfaces du futur logiciel. Ces diagrammes sont constitués d'un ensemble d'acteurs qui agit sur des cas d'utilisation. Il permet d'identifier les possibilités d'interaction entre le système et les acteurs (intervenants extérieurs au système), c'est-à-dire toutes les fonctionnalités que doit fournir le système. Il est présenté dans la

figure ci-dessous :

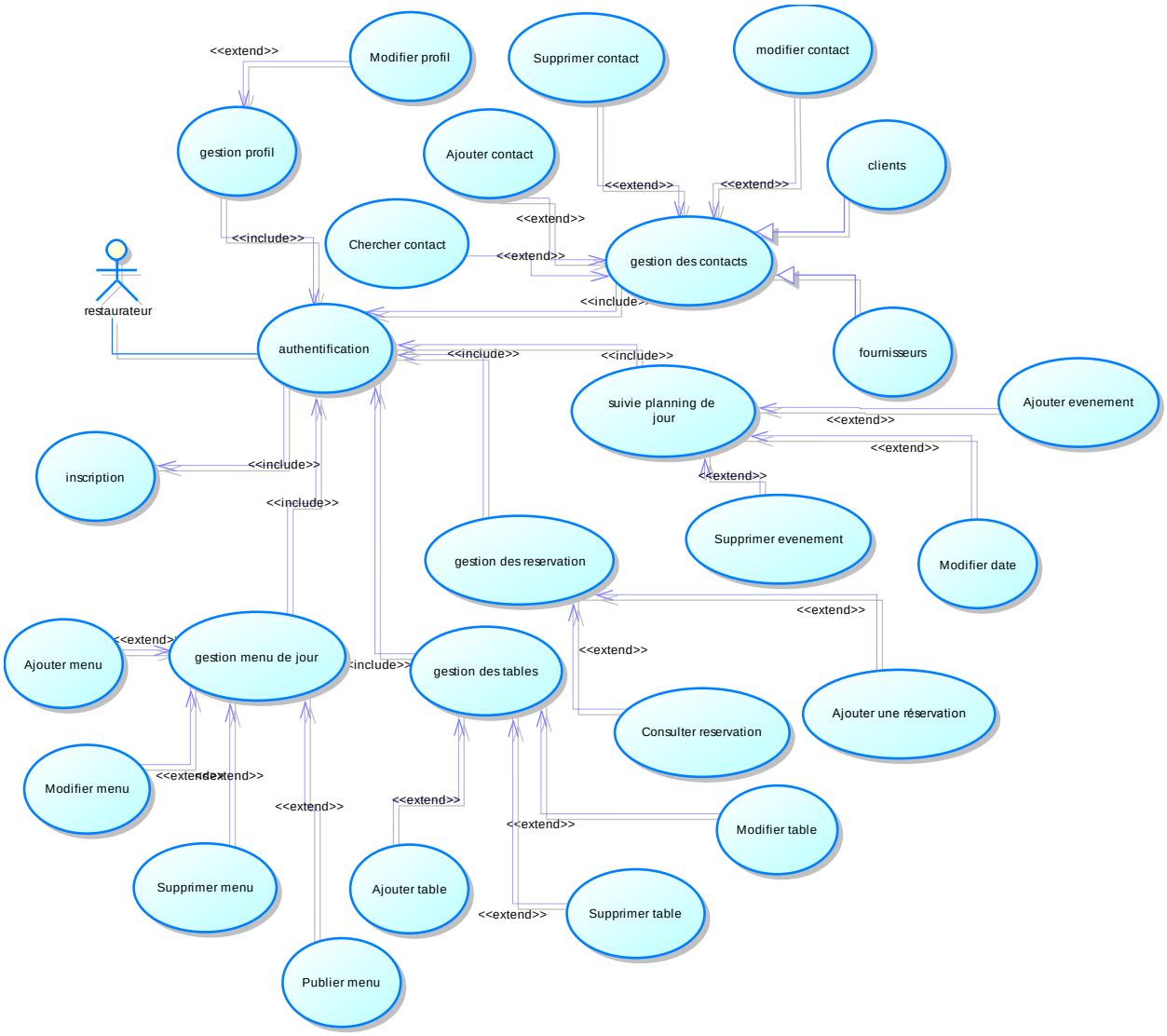

FIGURE 11- DIAGRAMME DES CAS D'UTILISATION

2. LA MODELISATION STATIQUE :

Après que nous avons présenté l'aspect dynamique de cette modélisation, nous allons aborder l'autre aspect statique, avec lequel on peut identifier les propriétés des objets et leurs liaisons avec les autres objets. L'un des diagrammes statiques nous intéresse beaucoup pour pouvoir implémenter le code, il s'agit du diagramme de classes.

A. DIAGRAMME DE CLASSE

Le diagramme de classe est un élément important dans une démarche de conception orientée

objet. Il représente les différentes entités (les classes d'objet) intervenant dans le système.

En identifiant les concepts importants de l'application, nous avons réalisé un premier diagramme de classes pour représenter ces concepts et leurs associations.

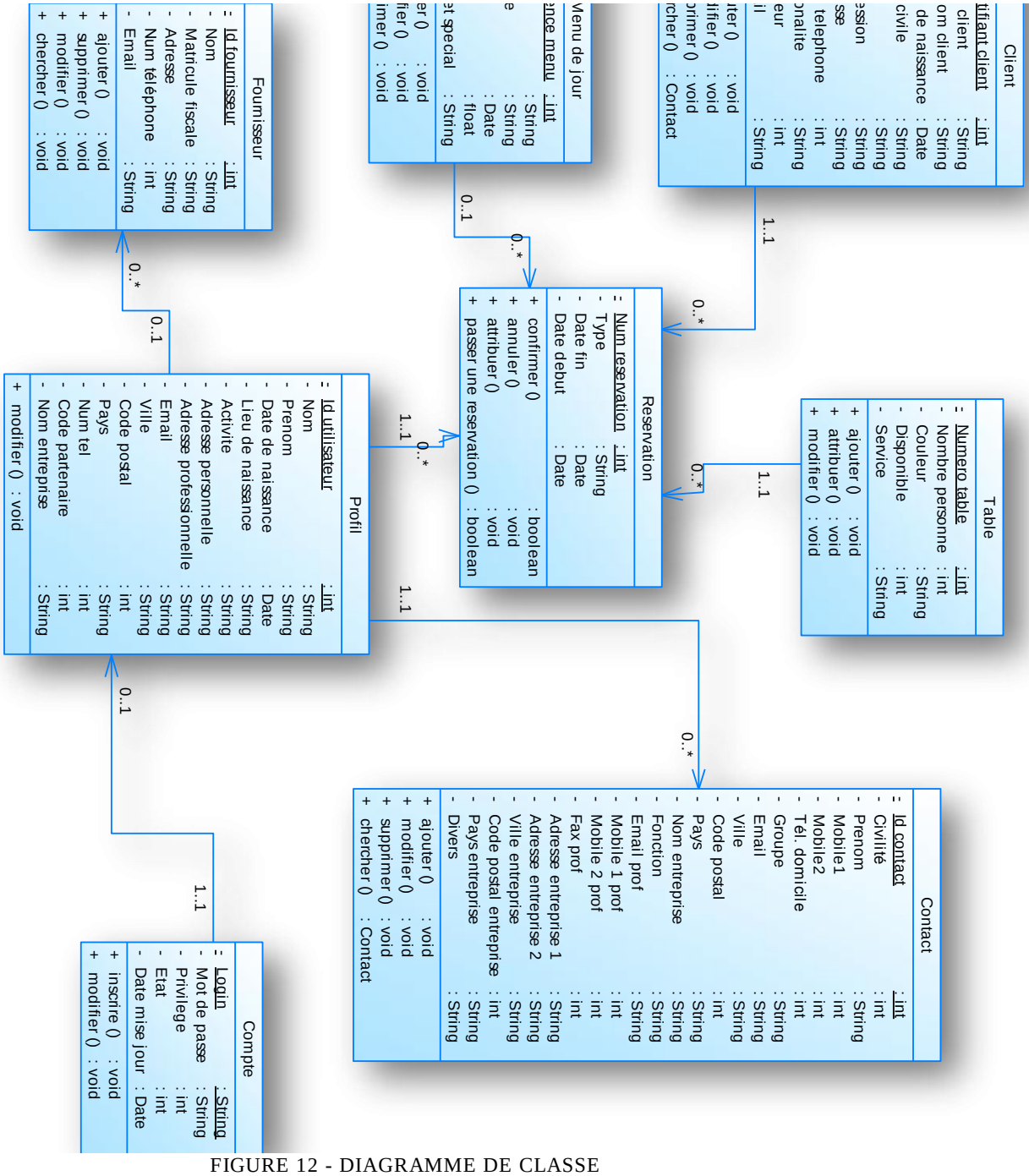

FIGURE 12 - DIAGRAMME DE CLASSE

B. CONCEPTION DE LA BASE :

Nous tenons à souligner que la base de données conçue est en troisième forme normale tout en veillant à ce que l'intégrité lors des ajouts et des suppressions de données soit maintenue d'une

manière transparente à travers un schéma de mapping qui permet d'assurer la transformation d'objets vers la base de données et vice versa que cela soit pour des lectures ou des mises à jour (création, modification ou suppression).

Concernant les cas d'héritages, nous avons opté pour un héritage par spécialisation. Nous avons également choisi de générer la classe générique et la classe spécifique sachant que dans cette dernière, seuls les attributs spécifiques figurent.

D'ailleurs, cette décision est justifiée par un fini d'optimisation, de sorte que la migration des propriétés communes n'y a pas lieu, bien sûr exception faite de la clef primaire. A cette dernière correspond une optimisation en espace de stockage par élimination de toute forme de redondance d'attributs.

Ainsi, le lien interclasses (générique et spécifique) est géré à l'aide d'une jointure sur la clé primaire .Du coups, nous avons veillé à ce que l'intégrité lors des ajouts et des suppressions soit maintenue.

CONCLUSION

L'objet de ce chapitre était l'aboutissement à la conception de la base de données. Pour ce faire, nous avons dû passer par des transformations des modèles et des structurations du système, tout en tenant adéquatement compte des contraintes techniques existantes pour l'implémentation du futur système qui fera l'objet du chapitre suivant.

CHAPITRE 3 : DEVELOPPEMENT

S'il ne fallait retenir qu'une vertu des Technologies de l'Information et de la Communication ce serait celle-ci : la possibilité d'offrir à chacun une tribune, un espace de liberté, d'expression. « André Santini »

INTRODUCTION

L’implémentation est le résultat de la conception, son importance est crucial pour le traitement

du système sous forme de composants pour chaque cas d'utilisation, et traduire les méthodes et les objets de cette application en des classes et de code source.

I. ENVIRONNEMENT DE DEVELOPPEMENT :

Il existe un panorama de technologie et de nombreuses solutions techniques pour mettre en œuvre une application web dynamique dont voici quelques-unes auxquelles nous avons notamment eu recours.

1. SGBD: MYSQL

MySQL est un serveur de bases de données relationnelles SQL développé dans un souci de performances élevées en lecture, ce qui signifie qu'il est davantage orienté vers le service de données déjà en place que vers celui des mises à jour fréquentes et fortement sécurisées. Il est multithread et multi-utilisateur.

C'est un logiciel libre développé sous double licence en fonction de l'utilisation qui en est faite: dans un produit libre ou dans un produit propriétaire. Dans ce dernier cas, la licence est payante, sinon c'est la licence publique générale GNU (GPL) qui s'applique.

2. EDI: NETBEANS

NetBeans est un environnement de développement intégré (EDI), placé en open source par Sun en juin 2000 sous licence CDDL et GPLv2 (Common Development and Distribution License). En plus de Java, NetBeans permet également de supporter différents autres langages, comme Python, C, C++, JavaScript, XML, Ruby, PHP et HTML. Il comprend toutes les caractéristiques d'un IDE moderne (éditeur en couleur, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages Web).

Conçu en Java, NetBeans constitue par ailleurs une plateforme qui permet le développement des applications web, il comprend toutes les caractéristiques d'un IDE moderne (coloration syntaxique, projets multi-langage, refactoring, éditeur graphique d'interfaces et de pages web, etc.).

3. LE SERVEUR D'APPLICATION J2EE : GLASSFISH

GlassFish est le nom du serveur d'applications Open Source Java EE 5 et qui sert de fondation au

produit Sun Java System Application Server de Sun Microsystems. Sa partie Toplink persistance provient d'Oracle. C'est la réponse aux développeurs Java désireux d'accéder aux sources et de contribuer au développement des serveurs d'applications de nouvelle génération de Sun.

II. LANGAGE & FRAMEWORK

1. Framework JSF

JavaServer Faces (abrégé en JSF) est un framework Java basé sur la notion de composants et destiné au développement d'applications Web. JSF est techniquement comparable à Swing ou SWT sauf qu'ils sont utilisés pour le développement d'application bureau, en JSF et en Swing l'état d'un composant (représenté par un objet java) est enregistré lors du rendu de la page, pour être ensuite restauré au retour de la requête.
Une page JSF est une page xhtml (ou jsp) liée aux Managed Bean via le language EL.
JSF est principalement constitué de :

- Un ensemble d'APIs pour la représentation et la gestion des composants, de leur état, des évènements, de la validation des entrées et la conversion des sorties, l'internationalisation et l'accessibilité ainsi que la navigation inter-vues.
- Deux jeux de composants standards (affichage de texte, saisie de texte, tables, zone à cocher, etc.): html et core.
- Deux bibliothèques de balises JSP (une pour chaque jeu de composants) pour permettre l'utilisation de JSP pour la construction de vues JSF.
- Un modèle évènementiel côté serveur.
- Les Managed-Beans : qui forment la couche contrôle de JSF.
- Unified Expression Language: abrégé en EL ou langage d'expressions unifié pour JSF et JSP 2.0. Il permet de lier les composants aux managed-beans.

L'objectif de JSF est de faciliter le développement des interfaces utilisateurs des applications web, or les deux de composants standards de JSF (html et core) s'avèrent limités et insuffisants pour le développement d'applications d'entreprise. Des jeux de composants additionnels qui offrent de nouveaux composants plus riches sont indispensables pour le développement en JSF, Primefaces en offre un qui a prouvé son efficacité.

C. JUSTIFICATION DE CHOIX DE LA LANGAGE DE PROGRAMMTION

JSF permet au développeur d'accroître la productivité du développement d'interfaces « client léger » tout en permettant une maintenance assez facile. JSF est un standard J2EE. JSF permet une séparation entre la couche présentation et les autres couches d'une application web une mise en place d'un mapping HTML/OBJET

- la réutilisation de composants graphiques
- une gestion de l'état de l'interface entre les différentes requêtes
- une liaison entre les actions coté « Client » et les actions des objets Java coté « Serveur »
- création de composants customs grâce à une API
- le support de différents clients (HTML, WML, XML, ...) grâce à la séparation des problématiques de construction de l'interface et du rendu de cette interface

2. API JPA 2.0 ECLIPSELINK

EclipseLink est un Framework open source de mapping objet-relationnel pour les développeurs Java. Il fournit une plateforme puissante et flexible permettant de stocker des objets Java dans une base de données relationnelle et/ou de les convertir en documents XML. EclipseLink est dérivé du projet TopLink de la société Oracle. Il supporte un certain nombre d'API relatives à la persistance des données et notamment la JPA.

3. API DE LOGGING : LOG4J

Log4j est un projet open source distribué sous la licence Apache Software initialement créé par CekiGülcü et maintenu par le groupe Jakarta. Cette API permet aux développeurs d'utiliser et de paramétrer un système de gestion de journaux (logs). Il est possible de fournir les paramètres dans un fichier de configuration ce qui rend sa configuration facile et souple. Log4j est compatible avec le JDK 1.1. et supérieur.

4. PRIMEFACES

PrimeFaces est un framework open source, qui peut être assimilé à une suite de composants d'interfaces riches pour JSF.

JSF apporte des éléments très basiques d'UI par contre PrimeFaces va plus loin et propose plus de 100 composants, plus de 35 thèmes, des composants mobiles et bien plus encore.

C'est actuellement de loin la suite pour JSF la plus populaire, ainsi qu'un outil graphique très populaire partout pour le développement de Java Web.
Ces bibliothèques complémentaires :

- PrettyFaces
- OmniFaces (bibliothèque utilitaire pour JSF)
- PrimeFaces Extensions

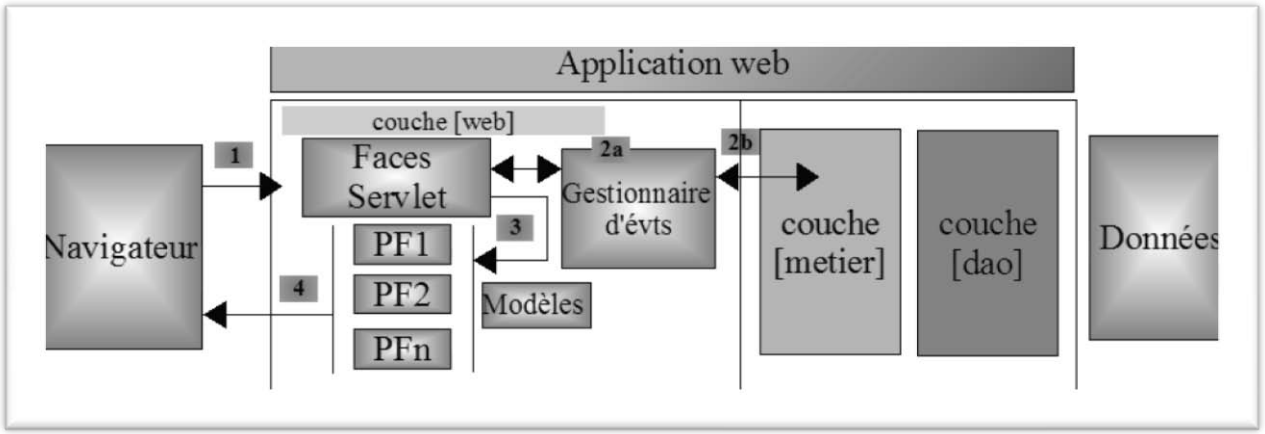

FIGURE 13- ARCHITECTURE D'UNE PAGE PRIMEFACES

II. ERGONOMIE ET CONCEPTION DES INTERFACES GRAPHIQUES

La conception d'un interface utilisateur doit prendre en compte les besoins du restaurateur et répondre aux exigences ergonomiques standards pour les applications web :

- ✓ **Organisation visuelle** :c'est la manière dont votre site est construit : il faut avoir une cohérence et clarté dans la mise en page et pouvoir identifier en un clin d'oeil les différents pavés ou blocs qui constituent cette application et leurs fonctionnalités
- ✓ **Cohérence** :c'est avoir un espace bien organisé et stable qui soit homogène tout au long de la visite de l'utilisateur.
- ✓ **L'assistance** : c'est guider les utilisateurs selon leurs besoins et leur attentes à un

moment précis.

III. QUELQUES INTERFACES DE CETTE APPLICATION

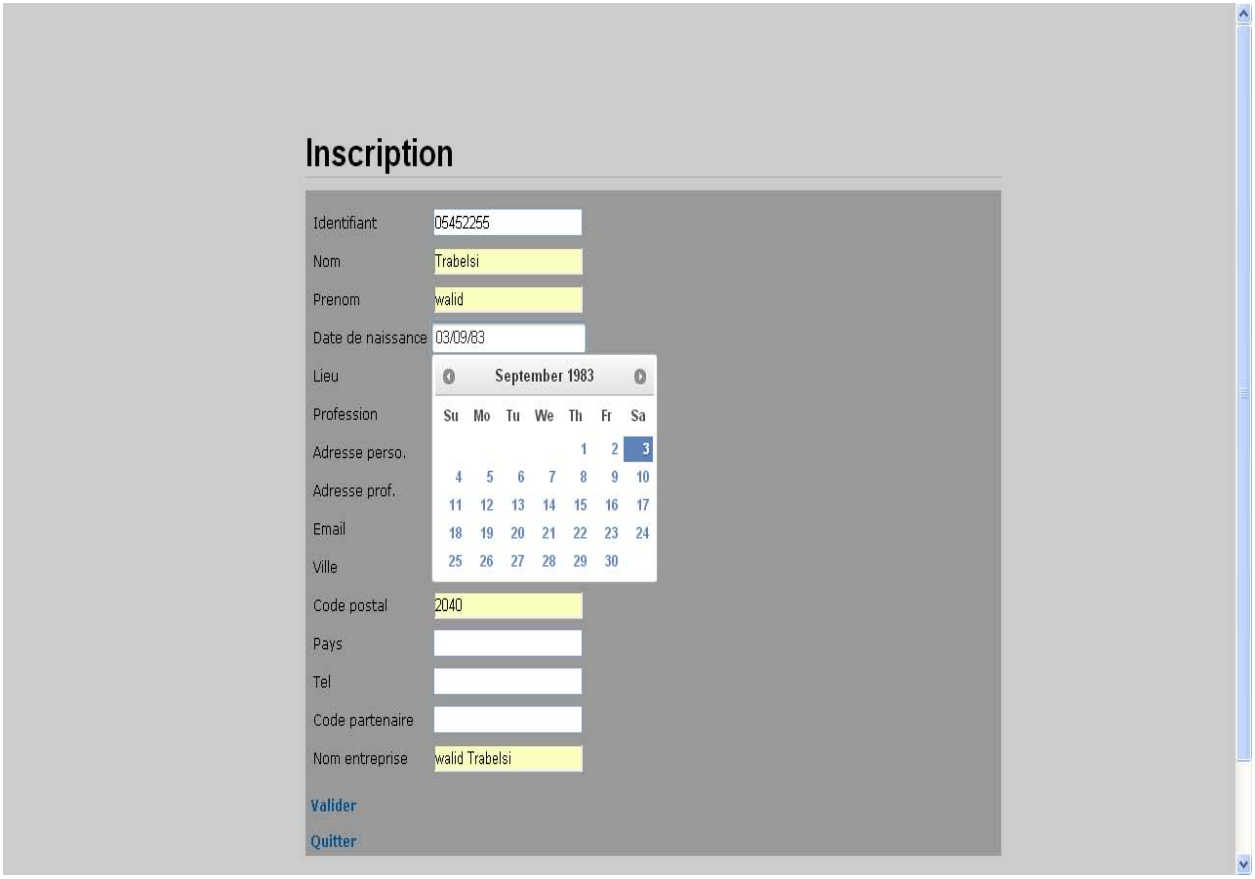

FIGURE 14 - LA PAGE D'INSCRIPTION

Chaque nouvel utilisateur est appelé à s'inscrire dans cette application en remplissant ce formulaire, après la validation de son inscription il recevra un email qui contiendra les paramètres de son compte (login , mot de passe).

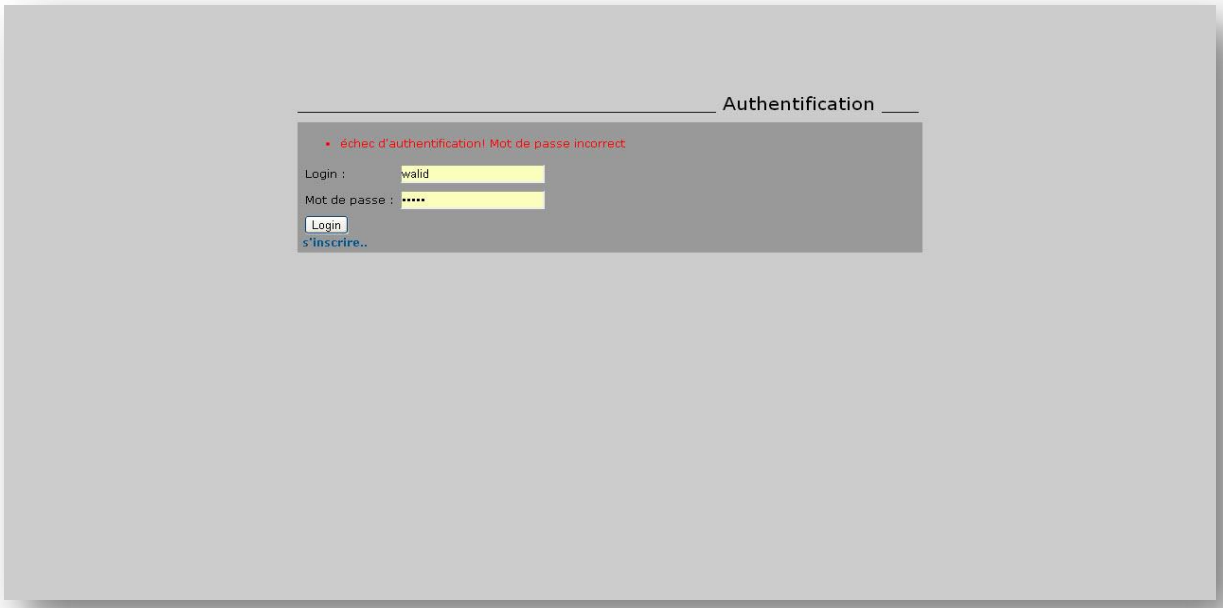

FIGURE 15 - PAGE AUTHENTIFICATION

C’est la première page qui s’affiche à l’utilisateur, elle lui offre la possibilité de s’authentifier en saisissant son login et mot de passe obtenu par son inscription.

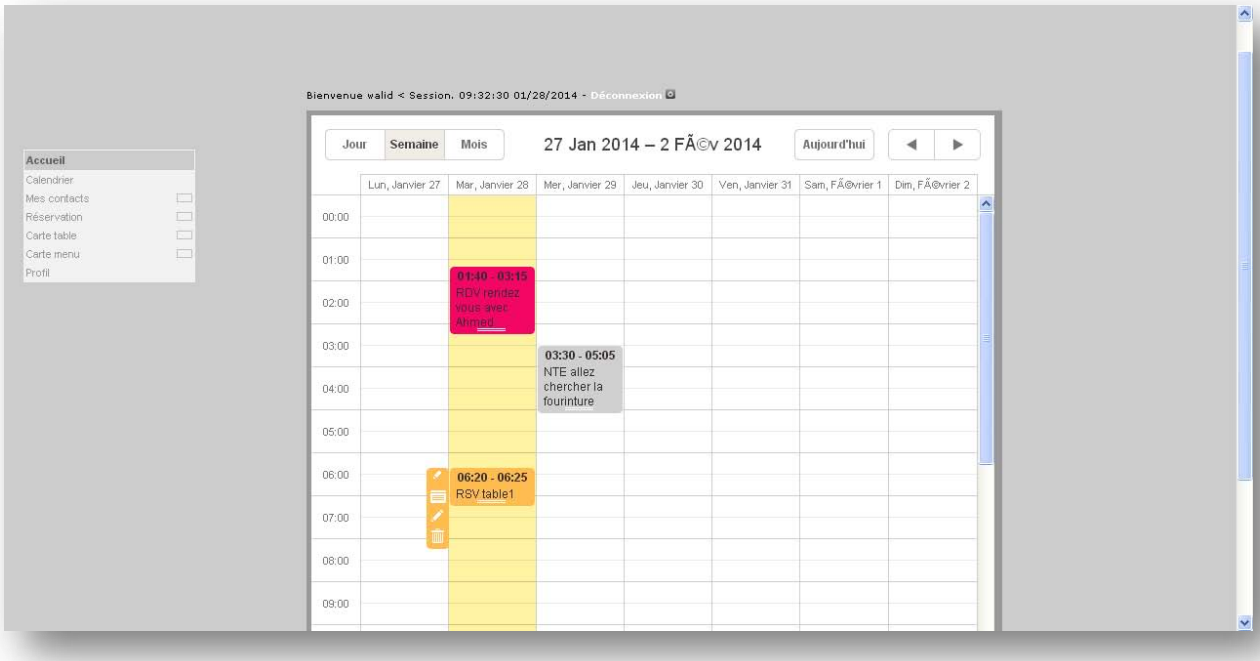

FIGURE 16- AGENDA DES EVENEMENTS

Par le biais de cet agenda il est possible de suivre les événements : les Rendez-vous, les réservations et les notes importantes associées à son utilisateur en temps réel. Comme il est possible de changer les dates des événements en déplaçant quelques cartes de notes sur cet agenda.

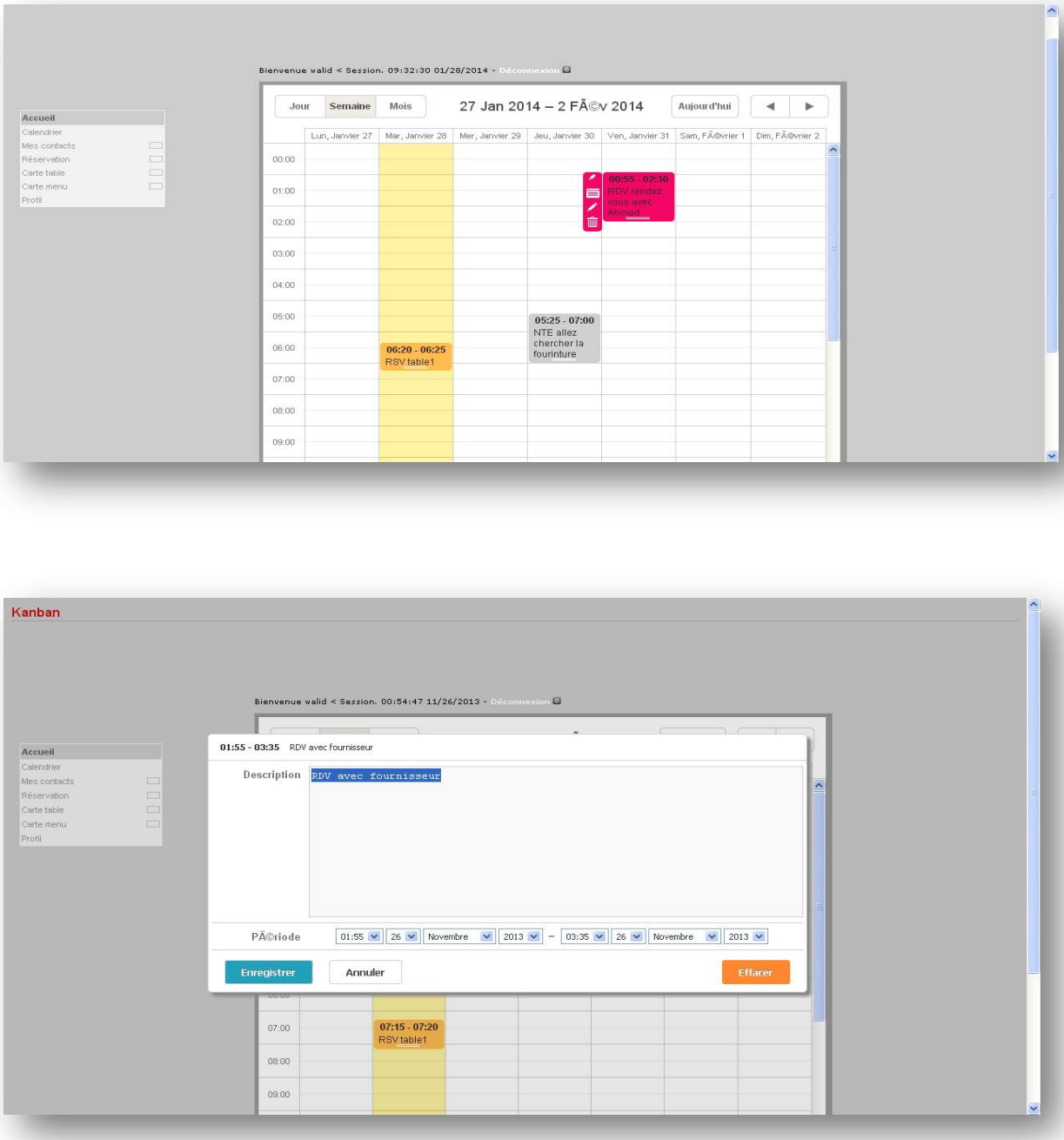

FIGURE 17 - PAGE AJOUTER UNE NOTE

Parmi les fonctionnalités offertes par notre application, nous citons l'ajout des notes avec une description et une période de temps bien déterminée selon les besoins de l'utilisateur.

FIGURE 18 - PAGE LISTE DES TABLES

Cette interface permet à l'utilisateur de consulter l'ensemble de ces tables ainsi que leurs états de disponibilité.

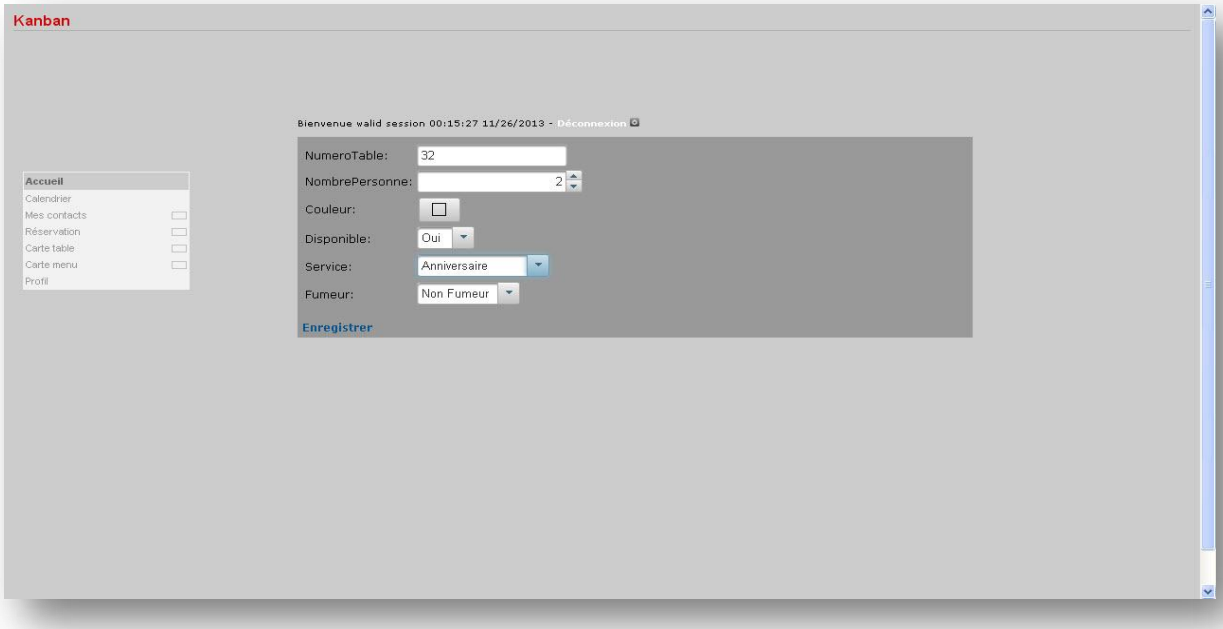

FIGURE 19 - PAGE AJOUTER UNE TABLE

Il est possible d'ajouter une nouvelle table identifiée par un numéro unique ou de modifier ses données.

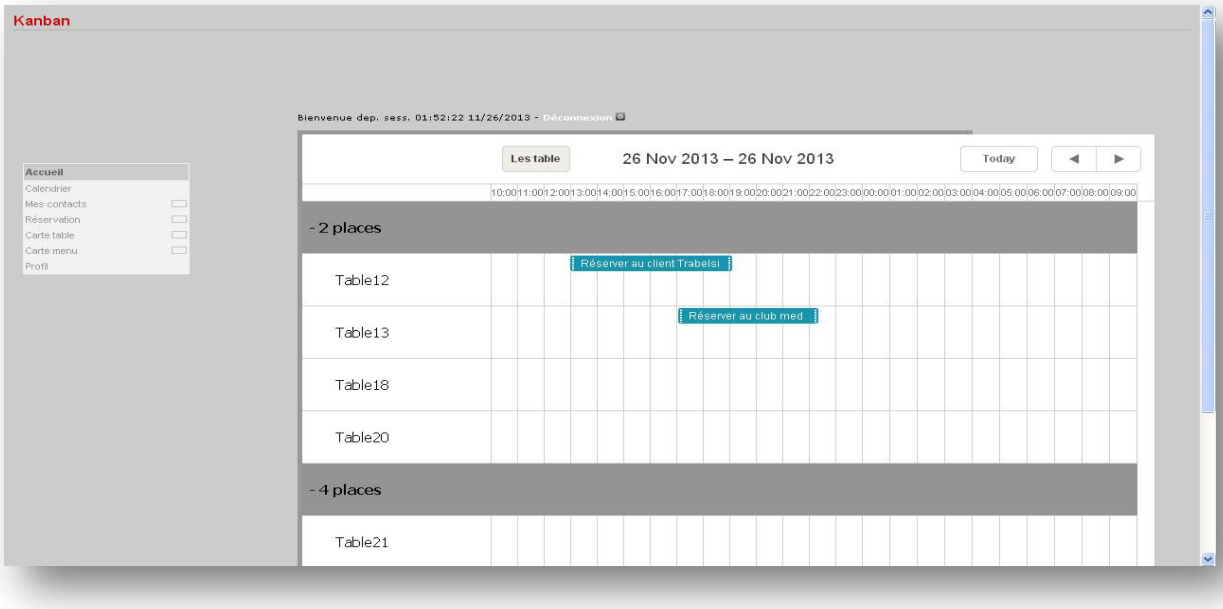

FIGURE 20- PAGES SUIVIE DES RESERVATIONS

On peut suivre avec ce tableau toutes les réservations relatives à chaque table, classées selon le nombre de places disponibles.

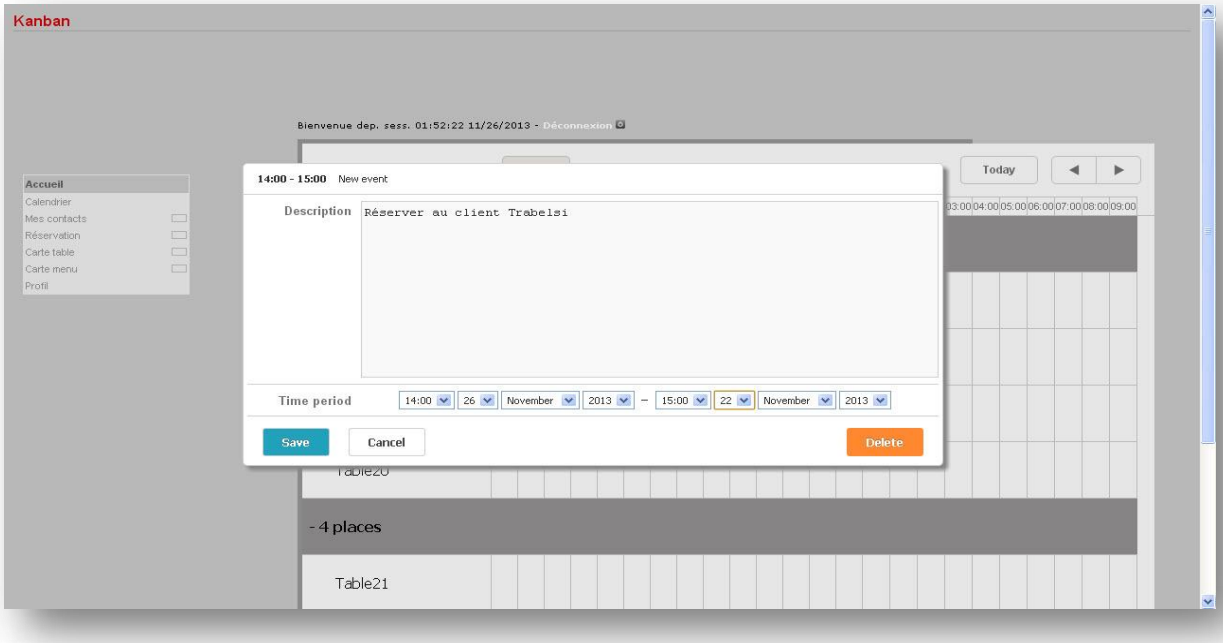

FIGURE 21 - AJOUTER RESERVATION

On peut réserver une table, à une période bien donnée, pour un client avec des détails sur cette réservation.

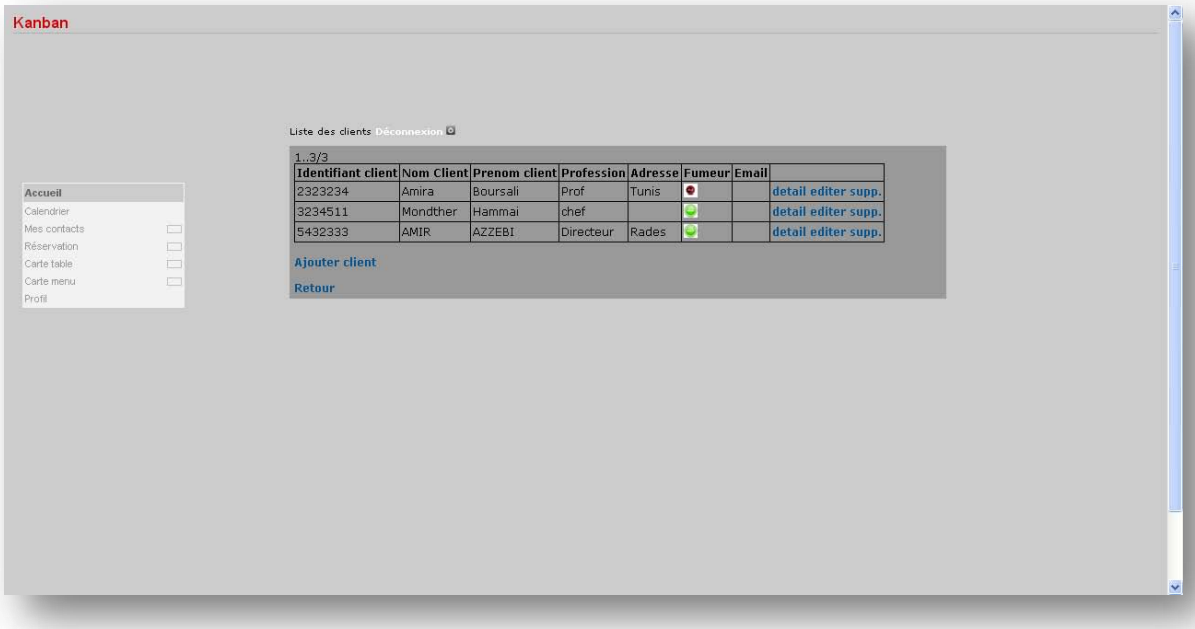

FIGURE 22– UNE PAGE POUR SUIVRE LA LISTE DE CONTACTS CLIENTELE

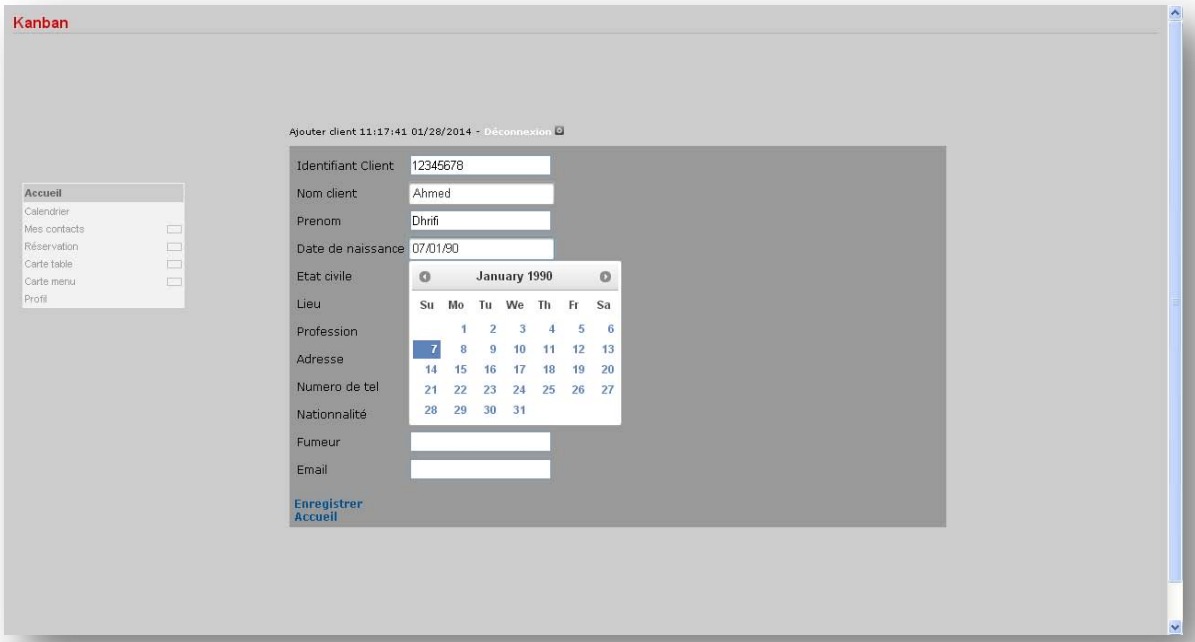

FIGURE 23–UNE INTERFACE POUR AJOUTER UN NOUVEAU CLIENT

On peut ajouter, via cette page, un nouveau client en fournissant ses différents détails : identifiant, nom, prénom…

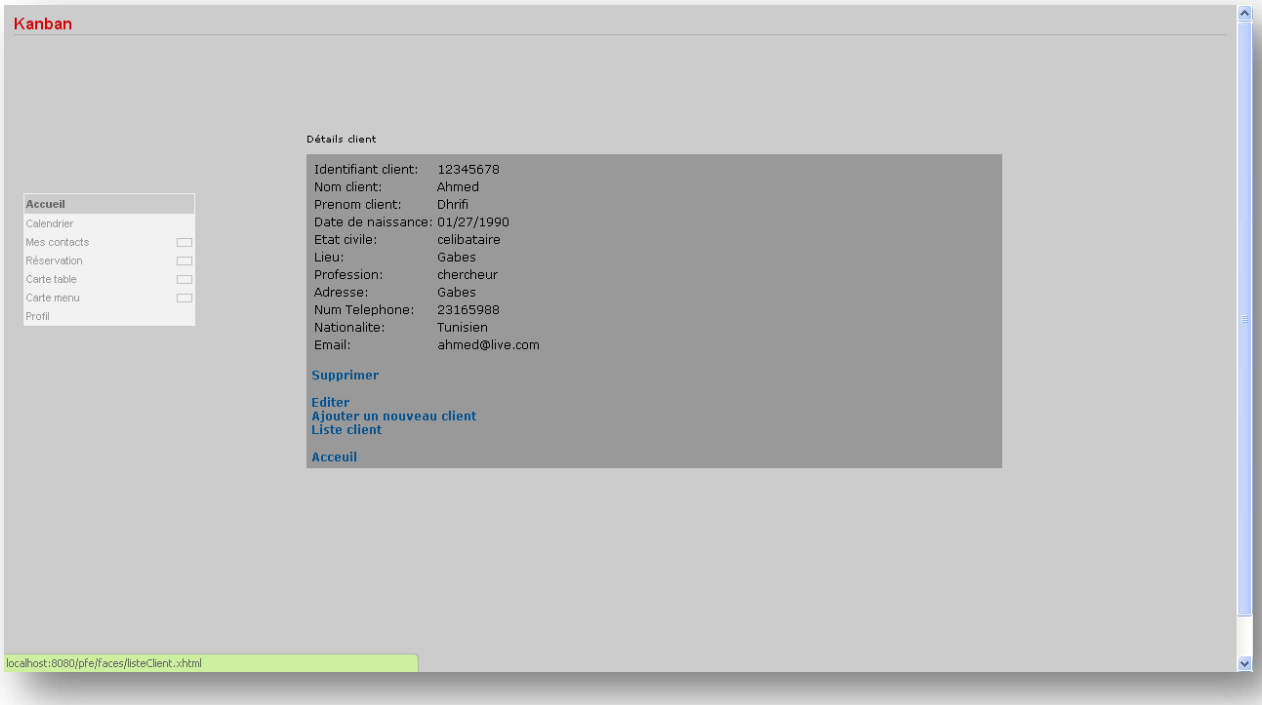

FIGURE 24–LA PAGE D'INFO. CLIENT

Il est possible de suivre, à travers cette interface, les différents détails de chaque client à part.

CONCLUSION

Dans ce chapitre, nous avons élaboré une étude préliminaire sur l'architecture logicielle et physique de notre application tout en se focalisant sur la description de l'environnement matériel et logiciel qui nous a permis d'implémenter notre application.

CONCLUSION GENERAL

Au terme de ce rapport, nous pouvons déduire que ce projet de fin d'études nous a constitué une occasion opportune ; Il nous a permis de confronter l'acquis théorique à l'environnement pratique et nous a menés à une recherche sur le plan scientifique. C'est là où à notre avis, réside la valeur d'un tel projet de fin d'études qui illustre les exigences de la vie professionnelle au côté bénéfique de l'enseignement pratique que nous avons eu à L'UVT.

Durant ce projet, nous étions chargés pour la conception et la réalisation d'une solution de gestion des données pour restaurateurs. Comme nous venons de le voir, la mise en place de cette application n'est pas forcément complexe, mais, elle exige tout de même qu'on suit une démarche structurée et rigoureuse.

De ce fait, un travail considérable de recherche sur Internet, une étude minutieuse sur les outils de travail et une analyse de l'environnement où fonctionne notre application, ont été faits afin de dégager les différents besoins et exigences du public cible et de choisir l'architecture informatique la mieux adaptée au système.

Ainsi, nous sommes passés à l'implémentation de notre application, à ce stade, nous avons prouvé de grands efforts pour entamer la réalisation dans un délai plus proche afin de passer à l'étape de validation. Cette validation doit être interactive et fluide afin de garantir aussi bien la facilité d'utilisation que l'utilité d'utilisation.

Pour conclure, nous confirmons que notre projet nous a permis d'une part de mobiliser nos différentes facultés de compréhension et d'intelligence, et d'autre part de savoir quoi, où et quand chercher l'information souhaitée. En effet, notre projet qui est polyvalent nous a poussés à explorer divers domaines qui semblent divergents, mais qui trouvent dans notre cas de figure un meilleur champ de convergence. Cette exploration était structurée et organisée de façon à préserver le bon enchaînement des travaux.

GLOSSAIRE

A

API: Application Programming Interface

C

CGI : Common Gateway Interface

D

DAO : Data Access Object

E

EJB : Entreprise Java Bean

EDI: environnement de développement intégré

H

Html: Hypertext Markup Language

J

JDK : Java Developpement Kit

J2EE : Java 2 Entreprise Edition

JSP : Java Server Page

JVM : Javavirtual Machine

JPA: Java Persistence API

JSF : Java Server Faces

L

LMD : langage de manipulation de données

M

MVC : modèle-vue-contrôleur

MCT : Modèle conceptuel des traitements

O

ODBC : Open Database Connectivity

ORM : Object-relational mapping

R

Row : enregistrement

S

SGBD : système de gestion de base de données

SQL : Structured Query Language

U

UML : Unified Modeling Language

UP : Unified Process

URL : Uniform Resource Locator

X

XHTML : Extensible HyperText Markup Language

BIBLIOGRAPHIE

- Penser en Java par Bruce Eckel.
- Programmer en Java par Claude Delannoy Editions Eyrolles.
- Des articles du site que tous les développeurs d'applications connaissent : Developpez.com.
- Java EE 5 :EJB 3.0 - JPA - JSP - JSF - Web services - JMS - GlassFish 3 - Maven 3 de Antonio Goncalves
- JSF 2.0 Cook book de Anghel Leonard.
- Java Efficace Guide de Programmation de Joshua Bloch ,Édition : Vuibert.
- Bien programmer en Java 7 : Auteur(s) : Emmanuel Puybaret , Editeur(s) : Eyrolles Collection : Blanche.
- Programmation Orienté Aspect pour Java / J2EEde R.Pawlak, J.Ph.Retaillé et L.Seinturier Édition : Eyrolles.

NETOGRAPHIE

http://www.java.com/fr/

http://www.mysql.fr/

http://glassfish-samples.java.net/

http://jawher.net/i-wrote/creation-dune-application-de-type-crud-avec-jsf-et-jpa/

http://www.tutorialspoint.com/log4j/log4j_quick_guide.htm

http://fr.wikipedia.org/

http://java.developpez.com/

http://codes-sources.commentcamarche.net/s/java

http://www.oracle.com/fr/technologies/java/overview/index.html

http://www.pourlesnuls.fr/

http://fr.openclassrooms.com/informatique/cours/apprenez-a-programmer-en-java

ANNEXE

JEE : Java Enterprise Edition, ou Java EE (anciennement J2EE), est une spécification pour la technique Java de Sun plus particulièrement destinée aux applications d'entreprise. Ces applications sont considérées dans une approche multi-niveaux1. Dans ce but, toute implémentation de cette spécification contient un ensemble d'extensions au framework Java standard (JSE, Java Standard Edition) afin de faciliter la création d'applications réparties.
Pour ce faire, Java EE définit les éléments suivants :

- Une plate-forme (Java EE Platform), pour héberger et exécuter les applications.
- Une suite de tests (Java EE Compatibility Test Suite) pour vérifier la compatibilité.
- Une réalisation de référence (Java EE Reference Implementation), qui est GlassFish.
- Un catalogue de bonnes pratiques (Java EE BluePrints).

UP Le processus unifié est un processus de développement logiciel itératif, centré sur l'architecture, piloté par des cas d'utilisation et orienté vers la diminution des risques. C'est un patron de processus pouvant être adaptée à une large classe de systèmes logiciels, à différents domaines d'application, à différents types d'entreprises, à différents niveaux de compétences et à différentes tailles de l'entreprise. Le document suivant présente sous la forme d'une note les concepts associés à ce processus.

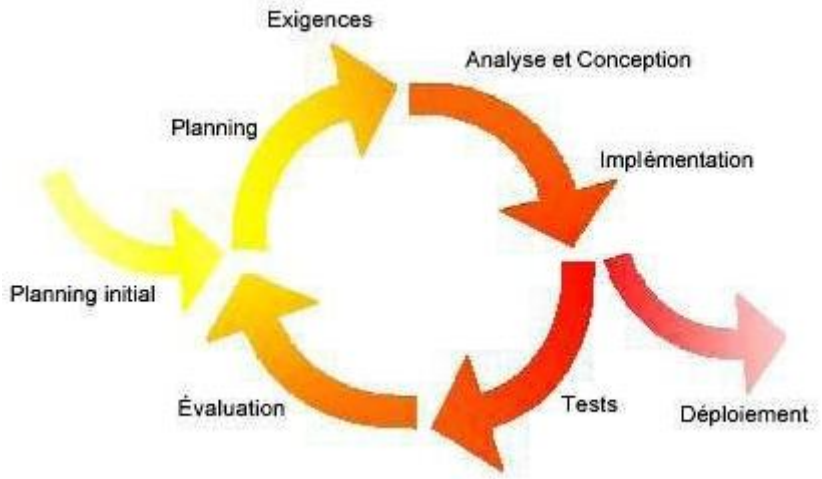

FIGURE 25 - LES PROCESSUS DE DEVELOPPEMENT D'UN LOGICIEL SELON UP

Le Kanban (kb) est un mode de gestion de flux créé par l'industrie automobile Japonaise (Toyota). Kanban signifie "étiquette", et comme c'est une méthode asiatique, elle implique deux principes:

- une très forte connotation visuelle,

- une recherche de la perfection au moindre coût (C'est un outil des principes « PULL »).

Rappelons tout d'abord que le stock est nécessaire pour faire face à l'irrégularité de la demande, même si la méthode PULL est appelée "Méthode zéro stock".

Les étiquettes sont donc le vecteur d'informations et sont rangées sur les emballages et un tableau « récapitulatif ».

Lorsque le tableau est vide, ceci signifie que toutes les étiquettes sont accrochées aux emballages. Le stock est alors à son zénith.

Au fur et à mesure de la consommation des emballages, les étiquettes reviennent et sont remises dans le tableau en commençant toujours par le même sens et atteignent des zones de couleurs différentes (Vert, Orange et Rouge, comme pour la circulation, plus le nombre d'étiquettes atteint le rouge, plus il y a urgence).

Il est recommandé de ne pas toucher la zone orange ni rouge car cela signifie qu'il ne reste que peu d'emballages entre le fournisseur et son client.

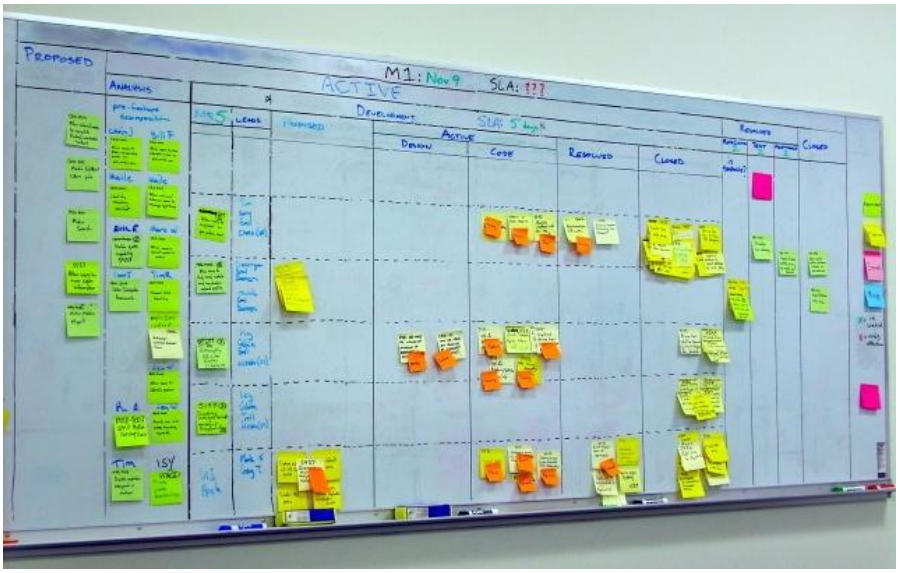

FIGURE 26–UN TABLEAU KANBAN

UML (sigle désignant l'Unified Modeling Language ou « langage de modélisation unifié ») est un langage de modélisation graphique à base de pictogrammes. Il est apparu dans le monde du génie logiciel, dans le cadre de la « conception orientée objet ». UML est couramment utilisé dans les projets logiciels. UML est l'accomplissement de la fusion de précédents langages de modélisation objet : Booch, OMT, OOSE. Principalement issu des travaux de GradyBooch, James Rumbaugh et Ivar Jacobson, UML est à présent un standard défini par l'Object Management Group (OMG). La dernière version diffusée par l'OMG est UML 2.4.1 depuis août 20111.

Le Java Development Kit (JDK) désigne un ensemble de bibliothèques logicielles de base du langage de programmation Java, ainsi que l'environnement dans lequel le code Java est compilé pour être transformé en bytecode afin que la machine virtuelle Java (JVM) puisse l'interpréter.
Il existe en réalité plusieurs JDK, selon la plate-forme Java1 considérée (et bien évidemment la version de Java ciblée) :
JSE pour la Java 2 Standard Edition également désignée J2SE ;
JEE, sigle de Java Enterprise Edition également désignée J2EE ;
JME 'Micro Edition', destinée au marché mobiles ;
À chacune de ces plateformes correspond une base commune de Development Kits, plus des bibliothèques additionnelles spécifiques selon la plate-forme Java que le JDK cible, mais le terme de JDK est appliqué indistinctement à n'importe laquelle de ces plates-formes.

La Java Persistence API (abrégée en JPA), est une interface de programmation Java permettant aux développeurs d'organiser des données relationnelles dans des applications utilisant la plateforme Java.La Java Persistence API est à l'origine issue du travail du groupe d'experts JSR 220.La persistance dans ce contexte recouvre 3 zones :l'API elle-même, définie dans le paquetage javax.persistence le langage Java Persistence,Query (JPQL),l'objet/les métadonnées relationnelles

JavaBeans est une technologie de composants logiciels écrits en langage Java. La spécification JavaBeans de Sun Microsystems définit les composants de type JavaBeans comme « des composants logiciels réutilisables manipulables visuellement dans un outil de conception ».Ils

sont utilisés pour encapsuler plusieurs objets dans un seul objet (= le « bean » ou haricot en français). Le « bean » regroupe alors tous les attributs des objets encapsulés, et peut définir d'autres attributs si besoin. Ainsi, il représente une entité plus globale que les objets encapsulés de manière à répondre à un besoin métier. En dépit de quelques similarités, les JavaBeans ne doivent pas être confondus avec les Entreprise JavaBeans (EJB), une technologie de composants côté serveur faisant partie de J2EE (Java2 Entreprise Édition).

JPA :

Littéralement « Java Persistence API », il s'agit d'un standard faisant partie intégrante de la plate-forme Java EE, une spécification qui définit un ensemble de règles permettant la gestion de la correspondance entre des objets Java et une base de données, ou autrement formulé la gestion de la persistance.

Ce mécanisme qui gère la correspondance entre des objets d'une application et les tables d'une base de données se nomme ORM, pour « Object Relational Mapping ». Ainsi dans le sens le plus simpliste du terme, ce que nous avons réalisé dans les chapitres précédents à travers nos DAO n'est rien d'autre qu'un ORM… manuel !

Persistance de données :

Au sens général, il s'agit simplement du terme utilisé pour décrire le fait de stocker des données d'une application de manière… persistante ! Autrement dit, de les sauvegarder afin qu'elles ne disparaissent pas lorsque le programme se termine. Rien de bien compliqué, en somme.

En Java, lorsqu'on parle d'une « solution de persistance des données », on évoque couramment un système permettant la sauvegarde des données contenues dans des objets. En réalité, vous connaissez donc déjà tous un moyen de persistance : le stockage dans une base de données relationnelle via JDBC !

Configuration de JSF :

Cette configuration revient à:

- Ajouter une implémentation JSF au classpath . Dans l'exemple fourni avec cet article, J'utilise la Sun Reference Implementation 1.2.6 téléchargeable ici . Cette version supporte JSF 1.2.

- Déclarer la Faces Servlet dans web.xml et l'associer à *.jsf . Cette servlet joue le rôle de la Front Servlet du MVC/Model 2 .
- Ajouter le fichier faces-config.xml qui permet de configurer l'application JSF (déclaration des managed-beans / controllers , déclaration des règles de navigation, etc.)

Après avoir déclaré la Faces Servlet ainsi que son affectation aux urls de type *.jsf , voici ce que devient web.xml:

```
<?xmlversion="1.0"encoding="UTF-8"?>
<web-appxmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns="http://java.sun.com/xml/ns/javaee"
        xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
        xsi:schemaLocation="http://java.sun.com/xml/ns/javaeehttp://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"
        id="WebApp_ID"version="2.5">
        <display-name>jsf-crud</display-name>
        <welcome-file-list>
                <welcome-file>index.jsp</welcome-file>
        </welcome-file-list>
        <servlet>
                <servlet-name>Faces Servlet</servlet-name>
                <servlet-class>javax.faces.webapp.FacesServlet</servlet-class>
                <load-on-startup>1</load-on-startup>
        </servlet>
        <servlet-mapping>
                <servlet-name>Faces Servlet</servlet-name>
                <url-pattern>*.jsf</url-pattern>
        </servlet-mapping>
</web-app>
```

Et voici le fichier faces-config.xml (vide pour l'instant, mais il sera mis à jour au fur et à mesure de l'ajout de nouvelles fonctionnalités à l'application):

```
<?xml version="1.0" encoding="UTF-8"?>
<faces-config xmlns="http://java.sun.com/xml/ns/javaee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-facesconfig_1_2.xsd"
    version="1.2">
</faces-config>
```

Configuration de JPA

La configuration de JPA consiste à:

Ajouter une implémentation JPA au classpath. J'utilise Toplink 2.41 comme implémentation.

Créer le descripteur de persistance qui sert à spécifier les paramètres de connexion à la base de données (url de connexion, login, mot de passe, etc.) ainsi qu'à la déclaration des classes persistantes.

J'utilise Toplink comme implémentation JPA et HSQLDB comme base de données. Il faut donc mettre toplink.jar et hsqldb.jar dans le classpath de l'application.

Il faut ensuite créer un descripteur de persistance JPA (persistence.xml) dans un dossier META-INF à la racine du dossier source.

```
<persistencexmlns="http://java.sun.com/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/persistence
 http://java.sun.com/xml/ns/persistence/persistence_1_0.xsd"
version="1.0">
<persistence-unit name="jsf-crud">
<properties>
<property name="toplink.logging.level" value="INFO" />
<property name="toplink.target-database" value="MYSQL" />
<property name="toplink.jdbc.driver"
                        value="com.mysql.jdbcDriver" />
<property name="toplink.jdbc.url"
                        value="jdbc:hsqldb:file:test" />
<property name="toplink.jdbc.user" value="sa" />
<property name="toplink.ddl-generation"
value="create-tables" />
</properties>
</persistence-unit>
</persistence>
```

Ce fichier ne contient que le nom de l'unité de persistance (jsf-crud) ainsi que les paramètres de connexion à la base de données qui sont:

PiloteJDBC : "com.mysql.jdbcDriver"

URL de connexion: " jdbc:mysql:file:test " qui indique à MYSQL de stocker la base de données dans un fichier nommé " test ". Il est aussi possible de stocker la base dans la mémoire vive (

RAM) en utilisant un chemin du type ” jdbc:mysql:mem:test “. Mais les données seraient alors perdues à l’arrêt de l’application.

Login: “sa” (l’équivalent du root dans MYSQL)

PiloteJDBC : ” com.mysql.jdbcDriver “

toplink.ddl-generation : ” create-tables ” indique à Toplink de créer les tables si elles n’existent pas.

Couche métier DAO

Un objet d'accès aux données (en Anglais Data Access Object ou DAO) est un patron de conception (c'est-à-dire un modèle pour concevoir une solution) utilisé dans les architectures logicielles objet. la partie Model de l’application devrait être séparée en deux sous parties: DAO (Data Access Object) pour l’accès aux données et Service pour faire la logique métier. Mais dans le cadre de cet article, on se limitera à la couche DAO sans passer par la couche service vu que l’on n’a pas de logique métier à proprement parler.

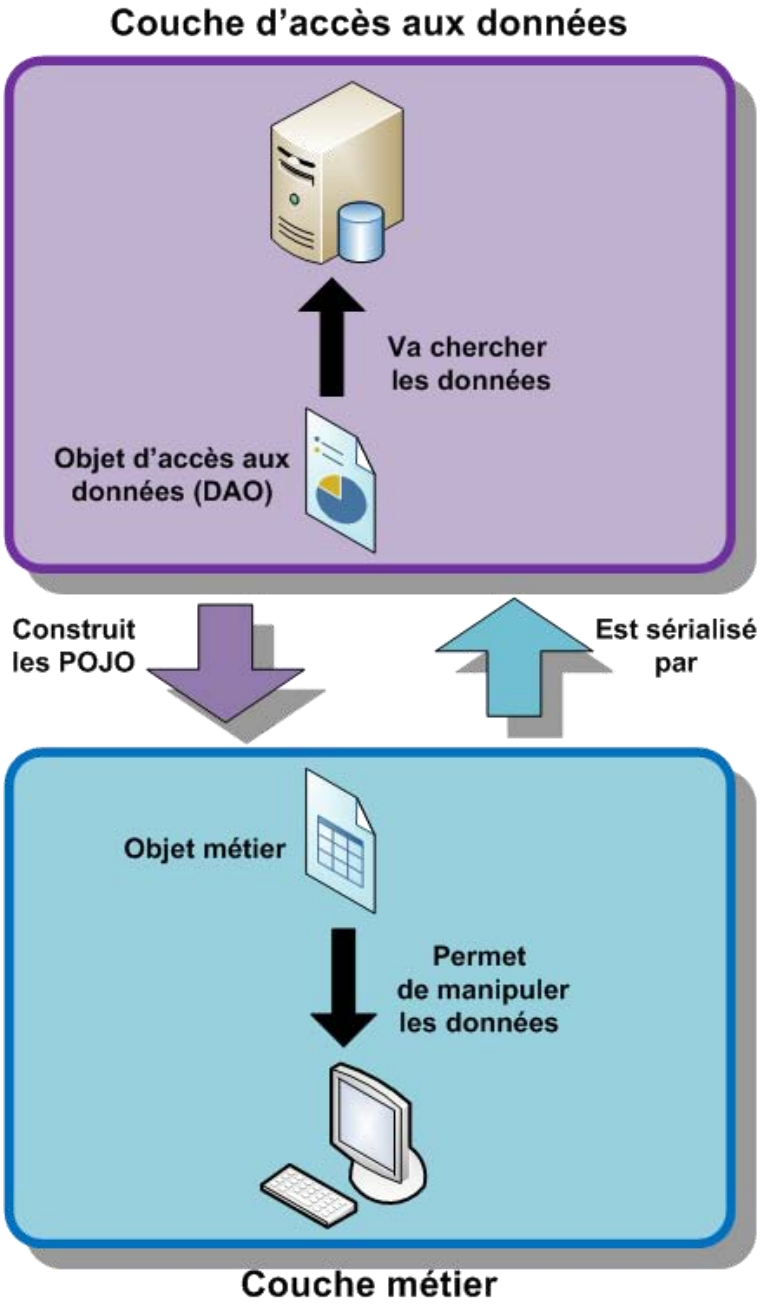

FIGURE 27 COUCHE D'ACCES AUX DONNEES

```
package model.dto;
 import javax.persistence.Entity;
import javax.persistence.GeneratedValue;
import javax.persistence.Id;

@Entity
public class Person {
private Long id;
private String firstName;
	private String lastName;
	@Id
	@GeneratedValue
	public Long getId() {
		return id;
	}
	 public void setId(Long id) {
		this.id = id;
	}
	 public String getFirstName() {
		return firstName;
	}
	 public void setFirstName(String firstName) {
this.firstName = firstName;
	}
	 public String getLastName() {
		return lastName;
	}
	public void setLastName(String lastName) {
this.lastName = lastName;
	}
}
```

Cette classe est annotée avec JPA pour pouvoir être persistée ou retrouvé dans la base de données:

- L'annotation @Entity sur la classe Person pour la déclarer comme classe persistante. Cette annotation est obligatoire pour une classe persistante.
- L'annotation @Id sur le getter du champ id pour le déclarer comme l'identifiant. Cette

annotation est obligatoire pour une classe persistante.

- L'annotation @GeneratedValue sur le getter du champ id pour déclarer qua sa valeur est auto générée.

Encapsuler les opérations de persistance (création, modification, suppression et lecture) sur l'entité Person dans un DAO

```
package model.dao;
 import java.util.List;
 import javax.persistence.EntityManager;
import javax.persistence.Persistence;
import model.dto.Person;

 public class PersonDao {
      private static final String JPA_UNIT_NAME = "jsf-crud";
      private EntityManagerentityManager;

      protected EntityManagergetEntityManager() {
            if (entityManager == null) {
entityManager = Persistence.createEntityManagerFactory(
                        JPA_UNIT_NAME).createEntityManager();
}
            return entityManager;
      }
}
```

Le DAO qu'on va développer va utiliser l'API de JPA pour réaliser les opérations de persistance. Pour pouvoir l'utiliser, il faut d'abord créer une instance de la classe EntityManager . Pour le faire, on passe par une fabrique (Factory) qu'on récupère via la méthode statique Persistence.createEntityManagerFactory() (Je sais, encore une autre fabrique :)) en lui passant comme paramètre le nom de l'unité de persistance (le même déclaré dans persistence.xml):

Log4j est une API de journalisation très répandue dans le monde Java. Il s'agit d'un système hautement configurable, que ce soit au niveau de ce qui doit être enregistré ou de la destination des enregistrements (serveur de logging, fichiers tournants, etc.). Pour cet article, je me suis appuyé sur la version 1.2.11, la version 1.3 devrait voir le jour en octobre 2005, elle introduit

certains changements au niveau de l'API ; néanmoins, les recommandations concernant la compatibilité avec la future API ont été appliquées.

Cette API est fréquemment utilisée comme système de journalisation sous-jacent en combinaison avec des autres API

Logger comme son nom l'indique, le Logger est l'entité de base pour effectuer la journalisation, il est mis en œuvre par le biais de la classe org.apache.log4j.Logger. L'obtention d'une instance de Logger se fait en appelant la méthode statique Logger. getLogger: Il est possible de donner un nom arbitraire au Logger, cependant, comme nous le verrons lorsque nous parlerons des hiérarchies de Loggers et de la configuration, il est préférable d'utiliser le nom de la classe pour des raisons de facilité.

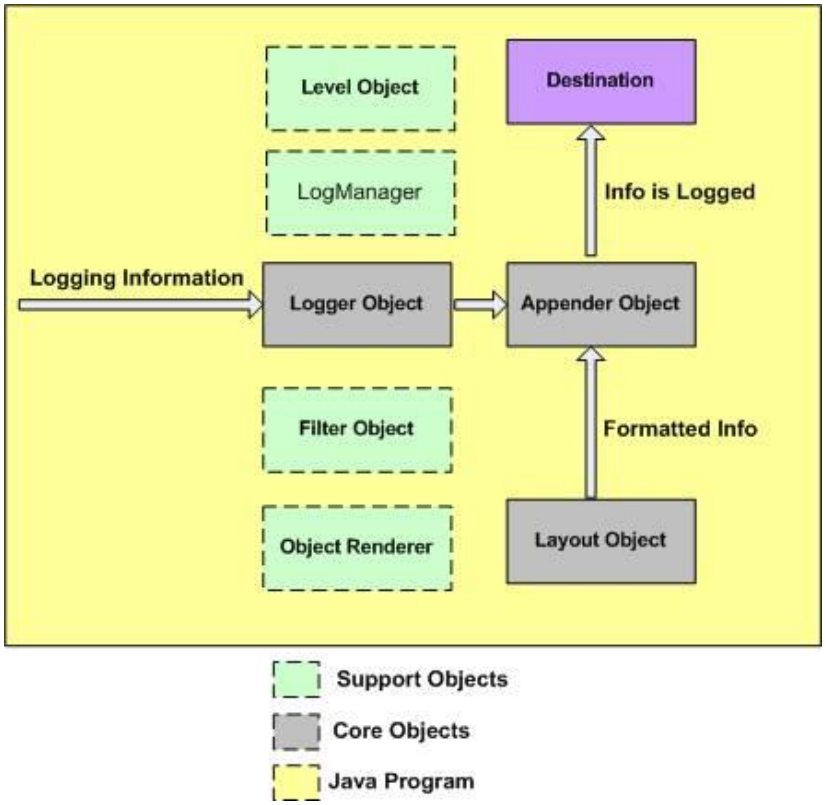

FIGURE 28 - LES COMPOSANT DE FRAMEWORK LOG4J

RESUME

Ce projet consiste à concevoir et à développer une application web d'architecture applicative 3-tiers pour la gestion des données restaurateurs : les contacts, les réservations, les rendez-vous, les tables, les notes, les activités planifiées avec un agenda électronique simple et facile à manipuler, les tableaux pour la suivi des contacts avec les fournisseurs ou les clients ainsi que leurs réservations. Ce travail est réalisé à l'aide du langage de programmation jsf , en interaction avec l'API « JPA » selon le modèle MVC.

Mots clés : Kanban , Agenda , Contact , JSF , J2EE , JPA , MYSQL

ملخص

هذا المشروع هو تصميم وبرمجة لتطبيق ويب , تم تصميمه وفق خاصية هندسية البرامج 3 طبقات لإدارة بيانات المطاعم: الاتصالات، والتحفظات، والتعيينات والجداول والملاحظات المخطط لها مع جدول لأعمال والأنشطة إلكترونية بسيطة من السهل التعامل معها، وجداول لتعقب جهات الاتصال مع الموردين أو العملاء وتحفظاتها.

يتم ذلك باستخدام لغة البرمجة JSF، و بتفاعل مع « API JPA » وفقا للنموذج. MVC

الكلمات الرئيسيةKanban :، التقويم، الاتصال، JSF, JPA، J2EE،MYSQL.

ABSTRACT

Our project can be summarized in designing and developing a web application that matches with the Three-tier management architecture, which is dedicated to the restoration of data such as booking, appointments, tables, notes. These data types would be planned with a simple electronic diary and activities that are easy to handle and track contacts such as suppliers or customers who are booking.

Key words: Kanban , J2EE, JSF, MYSQL, JPA, MVC.