

SARL UTOPIA & CO

RAPPORT DE STAGE

PROJET TESTENLIGNE

Joachim ZADI

27/11/2016



Maître de Stage : Jean-André REUSSITE

RAPPORT DE STAGE CDI	3
PREAMBULE	3
SARL UTOPIA & CO SAS	3
IDENTIFICATION	3
ACTIVITES	3
LE STAGIAIRE	3
IDENTIFICATION	3
MON PROJET	3
PRESENTATION	3
ENJEUX	4
ECHEANCIER PREVISIONNEL	4
CHOIX TECHNIQUES	4
CONCEPTION	5
EXPRESSION DU BESOIN	5
L'APPLICATION « TESTENLIGNE »	5
SPECIFICATIONS FONCTIONNELLES	6
LES ACTEURS	6
RECAPITULATIFS DES DIFFERENTS BESOINS PAR ACTEUR	7
LES CAS D'UTILISATIONS DES DIFFERENTS ACTEURS	8
« USE CASE UC01 » : SE CONNECTER	9
MAQUETTE DU « UC01 »	9
« USE CASE UC02 » : GESTION DES THEMES	10
SCENARIO DU « UC02 »	10
MAQUETTE DU « UC02 »	11
REMARQUE	11
« USE CASE UC09 » : PASSER UN EXAMEN	11
MAQUETTE DU « UC09 »	12
MODELISATION	13
ENTITES DE L'APPLICATION	13
MCD (Model Conceptuel de Données)	13
MPD (Model Physique de Données)	14
SCRIPT SQL (Voir annexe)	14
DIAGRAMMES DE SEQUENCE	15
DIAGRAMME DE SEQUENCE BOITE NOIRE	15
DIAGRAMME DE SEQUENCE BOITE BLANCHE	17
DIAGRAMME DES CLASSES	20
L'APPLICATION WEB	20
JAVA SERVER FACES	21
BREVE PRESENTATION	21
MAQUETTE DES PAGES DE L'APPLICATION	23
LE TEMPLATE DES PAGES	23
Page de login	24
Page accueil admin	25
Page de liste d'entité	26
Page de création d'entité	27
Page de modification d'entité	28
Page accueil user	29
Page de test	30
Page de résultat	31

DIAGRAMME DE NAVIGATION	32
Cas 1 : Echec de login	32
Cas 2 : Login admin	33
Cas 3 : Créer un thème	34
Cas 4 : Modifier un thème	35
Cas 5 : Supprimer un thème	35
Remarque	36
Cas 6 : Login user	36
Cas 7 : Passer le test	38
DESCRIPTION DU MENU	39
DEVELOPPEMENT	39
DESCRIPTION TECHNIQUE DE L'APPLICATION	39
MISE EN PLACE DU PROJET	39
Arborescence générale du projet	39
Arborescence des composants du modèle.	40
Arborescence des composants DAO.	41
Arborescence des composants « managebeans »	42
Arborescence des composants « Vue »	44
Intégration des composants avec Spring et Hibernate	45
Extrait de codes java	47
CONCLUSION	51
ANNEXE	52
SCRIPT SQL DE LA BASE DE DONNEES « STAGE »	52
SGDBR : MySQL	52

RAPPORT DE STAGE CDI

PREAMBULE

Ce document a pour but de présenter mon rapport de stage effectué au sein de l'entreprise **SARL UTOPIA & CO** au travers du développement d'une application web en Java/Java EE avec les Framework JSF 2.0, Hibernate 4.0 et Spring 3.2. Il n'est autre que l'achèvement de ma formation de « Concepteur Développeur Informatique » débuté à l'ESIC Paris 12, le 10 janvier 2013 et se veut avant tout pratique que théorique.

SARL UTOPIA & CO SAS

IDENTIFICATION

SARL UTOPIA & CO est l'entreprise au sein de laquelle s'est déroulé mon stage. C'est une SS2I, immatriculée au RCS (registre de commerce des sociétés) de Créteil sous le numéro 533 923 207. Elle est située au 30 Avenue Gallieni 94100 Saint Maur des Fossés. Elle est également représentée à Londres et en Afrique.

ACTIVITES

L'entreprise propose à ses clients des services intégrés alliant conseil pour leurs processus métiers, mais aussi développement et exploitation de leurs systèmes d'information. Au fil des années, elle a développé une solide base de connaissances autour des projets Java & Java/JEE apportant ainsi une forte valeur ajoutée aux services proposés à ses clients.

LE STAGIAIRE

IDENTIFICATION

Je suis Joachim ZADI, âgé de 55 ans, actuellement stagiaire à l'ESIC de Paris Levallois dans le département 92300 en région parisienne. Je termine la formation de Concepteur Développeur Informatique, débutée en septembre 2016 et s'achève à l'issue de ce stage. Je suis encadré actuellement par Monsieur Jean-André REUSSITE, ingénieur architecte, comme maître de stage. Ses conseils, sa connaissance du monde de l'entreprise et sa supervision m'ont permis de mener à bien et à terme le présent projet, en me guidant dans les technologies les plus utilisées dans le milieu de l'informatique.

MON PROJET

PRESENTATION

Le projet à réaliser est une application web qui a pour but de faire passer des tests de recrutement en ligne aux candidats souhaitant intégrer SARL UTOPIA & CO. L'issue de ces tests déclenchera éventuellement un entretien d'embauche avec le service RH de l'entreprise par la suite.

Cette démarche s'inscrit dans le cadre de l'évolution de l'entreprise qui souhaite améliorer son service de ressources humaines afin d'être plus efficace dans le recrutement de ces futurs collaborateurs.

ENJEUX

Aujourd'hui comme toujours d'ailleurs, le souci d'une entreprise prestataire de services comme SARL UTOPIA & CO est d'apporter un service de meilleure qualité à ses clients, ce qui suppose avoir des collaborateurs de bonnes compétences. La compétence dans le monde informatique et notamment dans celui du développement doit-elle se résumer au diplôme ? A cette réponse, le service RH a répondu par la négative, d'où la mise en place du projet afin de construire un mécanisme souple efficace et perfectible permettant de remplir les objectifs d'un bon recrutement.

ECHEANCIER PREVISIONNEL

Nom de la tâche	Durée en jour	Début
Planification des tâches	1	26/08/2013
Recueil de l'expression des besoins	1	
Recherche des solutions applicables	1	
Validation des solutions applicables	1	
Mise en place des diagrammes UML des cas d'utilisation	1	
Elaboration d'une instance du MCD	1	
Elaboration des diagrammes UML de séquence	2	
Elaboration du diagramme UML des classes	1	
Validation et adaptation des différents diagrammes	1	
Validation de la conception	1	
Elaboration du design du site (CSS3)	1	
Choix et validation des outils de travaux	1	
Création du projet Maven	1	
Mise en place de la convention de nommage	1	
Développement des classes java	2	
Configuration des web.xml, beansConfig.xml et securityConfig.xml	1	
Mise de l'interface IHM avec JSF	1	
Déploiement de l'application et validation des services	1	
Rédaction du rapport de stage	9	

CHOIX TECHNIQUES

Pour réaliser le projet, de concert avec mon maître de stage, nous avons convenu des choix techniques suivants :

Niveau	Produits
Système d'exploitation	Windows 7
SGBDR	MySQL 5
EDI	Eclipse / Netbeans 7.3
Langage de développement	Java/JEE
Journalisation	API Log4j
Serveur Web	Apache Tomcat 7

Conteneur	Spring IOC
Moteur de Persistance	Hibernate 4
Structure du projet	Maven
Pages web	JSF 2
Conception	PowerAMC et Visual Paradigm for UML

L'application sera développée suivant le design pattern MVC 2. Le Framework JSF 2 pour le développement de l'IHM est adapté de façon naturelle à ce mode de développement.

CONCEPTION

EXPRESSION DU BESOIN

Afin de décrire les besoins du service RH, nous avons utilisé des diagrammes de cas d'utilisation et d'activité du formalisme UML (*Unified Modeling Language*), complétés par des schémas de conception Merise.

L'APPLICATION « TESTENLIGNE »

DESCRIPTION

L'application *TESTENLIGNE*, comme indiqué précédemment, a pour but de permettre à une personne candidate à un poste chez SARL UTOPIA & CO de passer un test de présélection. Elle met donc de facto en jeu deux (2) volets dans la structure de l'application à développer :

- Une partie « user » (Front Office) accessible par le candidat, lui permettant d'effectuer les tests.
- Une partie « admin » (Back Office) accessible par le service RH permettant de gérer les tests à suggérer, ainsi que la création des candidats.

PREREQUIS A LA REALISATION DU PROJET

La réalisation du projet a nécessité une mise à niveau de mes connaissances en informatique. Il s'est imposé à moi, avec les conseils de mon maitre de stage, l'utilisation de certains Frameworks.

Le planning suivant a été retenu pour faire ma mise à niveau vers les outils couramment utilisés par l'entreprise.

Fiche suivi de la mise à niveau									
Nom : [REDACTED]	Durée prévue : 280				Date point d'avancement : [REDACTED]		Les temps sont exprimés en heures		
Tâches	Ressources	Date début prévue	Date fin prévue	Date début réelle	Temps prévu	Temps Effectif	Ecart	Causes écarts	Temps restant à passer
PRISE DE CONTACT : PRESENTATION DU PERSONNEL DE L'ENTREPRISE									
APPRENTISSAGE DES OUTILS DE TRAVAIL UTILISES DANS L'ENTREPRISE									
Api Log4j	TOUS	[REDACTED]	[REDACTED]	[REDACTED]	40	40	0		0
Maven	TOUS	[REDACTED]	[REDACTED]	[REDACTED]	40	40	0		0
Hibernate	TOUS	[REDACTED]	[REDACTED]	[REDACTED]	80	80	0		0
Spring	TOUS	[REDACTED]	[REDACTED]	[REDACTED]	120	120	0		0

NB : dans le tableau ci-dessus, il faudra retenir que le mot « TOUS » de la colonne « ressources » est un abus.

SPECIFICATIONS FONCTIONNELLES

LES ACTEURS

Après une série d'entretiens sur le fonctionnement de l'application et ses objectifs, nous avons ébauché le schéma suivant :

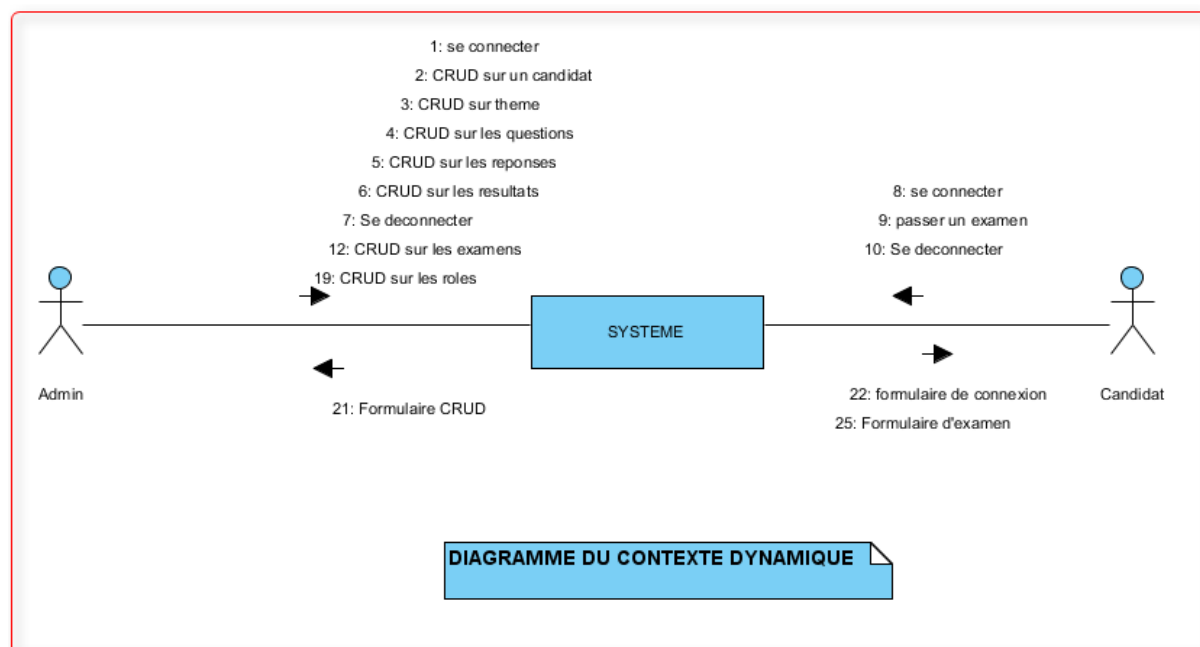


Figure 1 : Diagramme du contexte dynamique

La figure ci-dessus nous montre de façon non exhaustive, les différentes interactions qui entrent en jeu entre le **système** représenté ici par « l'application », et les différents « **acteurs** » :

- L'acteur « **Admin** » qui représente le service RH
- L'acteur « **Candidat** » qui représente un utilisateur candidat à un poste

RECAPITULATIFS DES DIFFERENTS BESOINS PAR ACTEUR

ACTEUR	BESOINS	FONCTIONS PRINCIPALES
Admin	Se connecter	<ul style="list-style-type: none"> Accéder à l'espace de gestion de l'application
	Gérer les Thèmes	<ul style="list-style-type: none"> Créer un nouveau thème Modifier un thème Supprimer un thème Lister les thèmes
	Gérer les Examens	<ul style="list-style-type: none"> Créer un nouvel examen Modifier un examen Supprimer un examen Lister les examens Associer un examen à un thème
	Gérer les Questions	<ul style="list-style-type: none"> Créer une nouvelle question Modifier une question Supprimer une question Lister les questions Associer une question à un examen
	Gérer les Réponses	<ul style="list-style-type: none"> Créer une nouvelle réponse Modifier les réponses Supprimer une réponse Lister les réponses Associer une réponse à une question
	Gérer les Résultats	<ul style="list-style-type: none"> Créer un résultat Lister les résultats
	Gérer les « User »	<ul style="list-style-type: none"> Créer un « user » Supprimer un « user » Modifier un « user » Lister les « user » Associer un rôle à un « user »
Candidat	Gérer les Rôles	<ul style="list-style-type: none"> Créer un profil Modifier un profil Supprimer un profil Lister les profils
	Passer un examen	<ul style="list-style-type: none"> Se connecter et passer le test

LES CAS D'UTILISATIONS DES DIFFERENTS ACTEURS

On résume le diagramme du contexte dynamique élaboré ci-dessus, dans les scenarios de cas d'utilisation présentés sur la figure suivante :

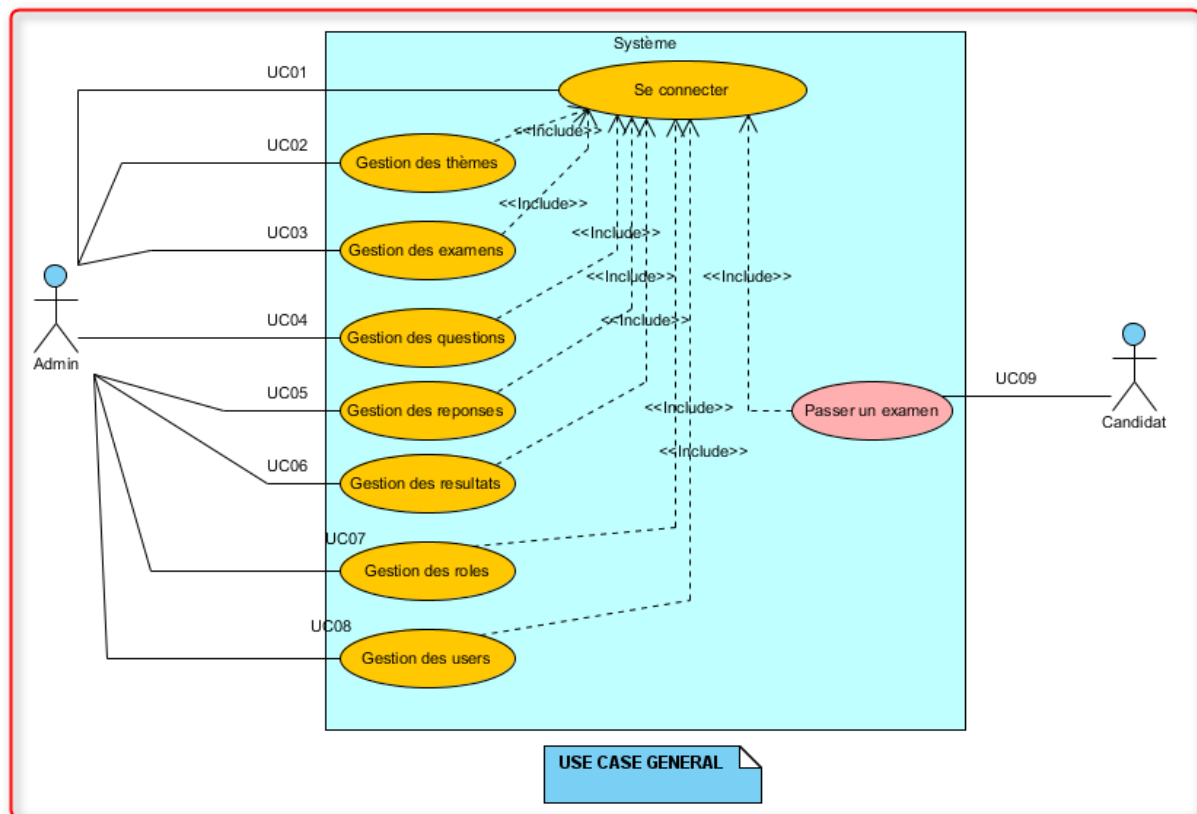


Figure 2 : Diagramme des cas d'utilisation

Neuf(9) cas d'utilisation sont recensés sur le système, dont sept (7) réservés à l'acteur « Admin », un(1) à l'acteur « Candidat », et un cas d'utilisation commun aux deux acteurs.

« USE CASE UC01 » : SE CONNECTER

Détaillons dans le tableau ci-dessous, le cas d'utilisation « UC01 » relatif à la connexion d'un usager du site.

UC01	SE CONNECTER
Acteurs	« Admin » et « Candidat »
But	S'identifier en tant qu'administrateur ou candidat
Pré-conditions	Avoir un compte
Post-conditions	Etre connecté au système
Déclenchement	Clic sur le lien « <i>connexion</i> »
Scénario	<ol style="list-style-type: none">1. Renseignement des champs « Login » et « mot de passe »2. Le système vérifie les saisies des différents champs<ol style="list-style-type: none">a. « Login » et « mot passe » sont reconnus :<ul style="list-style-type: none">◦ Affiche l'interface administrateur si le rôle affecté au « Login » est « Admin »◦ Affiche l'interface du candidat si le rôle affecté au « Login » est « Candidat ».b. « Login » et « mot passe » non reconnus :<ul style="list-style-type: none">◦ Exception
Exceptions	<ul style="list-style-type: none">▪ Affichage du formulaire de connexion
Contexte d'utilisation	<ul style="list-style-type: none">▪ « Admin » : exécution des « UC02, UC03 ... à UC08 ».▪ « Candidat » : exécution du « UC09 »

MAQUETTE DU « UC01 »

On en déduit une maquette du cas d'utilisation de la fenêtre de connexion au système.

Figure 3 : Maquette de la fenêtre de connexion au système

Figure 3 : Maquette de la fenêtre de connexion au système

« USE CASE UC02 »: GESTION DES THEMES

Le « UC02 » est relatif à la gestion des thèmes. Pour rappel, il est exécuté par l'acteur « Admin ». Ce cas d'utilisation possédant plusieurs scenarios, nous allons utiliser un diagramme d'activité du formalisme UML pour présenter son scenario.

UC02	GESTION DES THEMES
Acteurs	« Admin »
But	Créer, modifier, supprimer un thème
Pré-conditions	Etre connecté au système : avoir exécuté l'« UC01 »
Post-conditions	Lister les thèmes
Déclenchement	Clic sur l'un des liens « Ajouter », « Modifier » ou « Supprimer »
Exceptions	<ul style="list-style-type: none">▪ Si tentative de création en double d'un thème
Contexte d'utilisation	<ul style="list-style-type: none">▪ Création d'un nouveau thème▪ Suppression d'un thème▪ Modification d'un thème

SCENARIO DU « UC02 »

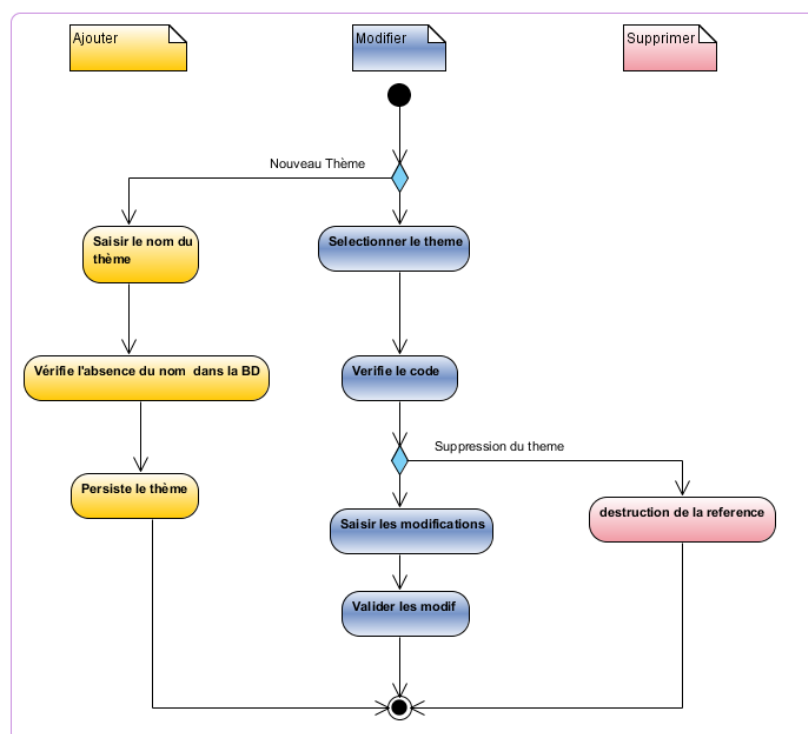


Figure 4 : Scénario de l'UC02

MAQUETTE DU « UC02 »

On en déduit une ébauche des maquettes des différents écrans intervenant dans l'« UC02 »

La maquette de l'UC02 de la gestion des thèmes est présentée dans une zone délimitée par une bordure rouge. Elle contient trois écrans distincts :

- Liste des thèmes** : Un tableau à deux colonnes. La première colonne, intitulée 'Nom', liste 'Thème 1', 'Thème 2', 'Thème 3' et 'Thème 4'. La seconde colonne, intitulée 'Opérations', contient pour chaque thème les liens 'Modifier', 'Supprimer' et 'Gérer les examens'. En dessous du tableau se trouve un bouton 'Ajouter un thème'.
- Ajouter un thème** : Un formulaire avec un champ 'Nom' suivi de la légende 'Entrez le nom du thème' et un bouton 'Ajouter'.
- Modifier un thème** : Un formulaire similaire au précédent, avec un champ 'Nom' suivi de la légende 'Entrez le nom du thème' et un bouton 'Modifier'.

Figure 5 : Maquette de l'UC02 de la gestion des thèmes

REMARQUE

L'« UC02 » étant semblable aux « UC03, UC04,..., UC08 » nous ne représentons pas ces derniers cas d'utilisation schématiquement dans notre rapport afin de ne pas être répétitif. On représentera donc dans la suite, le « UC09 ».

« USE CASE UC09 » : PASSER UN EXAMEN

Le tableau ci-dessous détaille le cas d'utilisation du « UC09 »

UC09	PASSER UN EXAMEN
Acteurs	« Candidat »
But	S'identifier en tant que candidat
Pré-conditions	Avoir un compte déjà créé par l'administrateur
Post-conditions	Un résultat est mis à jour.

Déclenchement	Clic sur le lien « <i>Commencer</i> »
Scénario	<ol style="list-style-type: none"> Lecture du message de bienvenue, <ol style="list-style-type: none"> vérifier que les nom et prénom sont bien ceux du candidat. Cliquer sur « Commencer » Tant qu'il y a des questions, le candidat coche les réponses et clic sur le bouton « validez la question». Au terme des questions, il obtient son résultat. A la fin, il se déconnecte du système
Exceptions	<ul style="list-style-type: none"> aucune
Contexte d'utilisation	<ul style="list-style-type: none"> « Candidat » : exécution des « UC09 »

MAQUETTE DU « UC09 »

Une ébauche des écrans de l'« UC09 » est représentée sur la figure suivante.

Fenêtre de test

Bienvenue Nom Prénom pour votre test de recrutement.
 Nous sommes le Date.
 Pour commencer, cliquer sur le bouton "commencer"

Commencer

Fenêtre de test

Question 01	Les spécifications Java décrivent :
<input type="checkbox"/>	un langage de programmation
<input type="checkbox"/>	un navigateur internet
<input type="checkbox"/>	un ensemble de bibliothèques
<input type="checkbox"/>	un logiciel d'exploitation

Validez

Figure 6 : Maquette de l'UC09

MODELISATION

ENTITES DE L'APPLICATION

MCD (Model Conceptuel de Données)

L'analyse détaillée des différents cas d'utilisation vus précédemment nous permet de dégager les entités suivantes comme « entités participant » à notre application :

Entités	Description
Thème	Représentation des différents thèmes que peut comporter un examen
Examen	Représentation des différents examens auxquels peut participer un candidat
Résultat	Représentation de différents résultats d'un examen
User	Présentation des différents utilisateurs du système
Rôle	Représentation des différents rôles attribués à un utilisateur
Question	Représentation des différentes questions pouvant être posées à un examen
Réponse	Représentation des réponses à une question
Sujet	Représentation de la série de questions sélectionnées pour le test d'un candidat

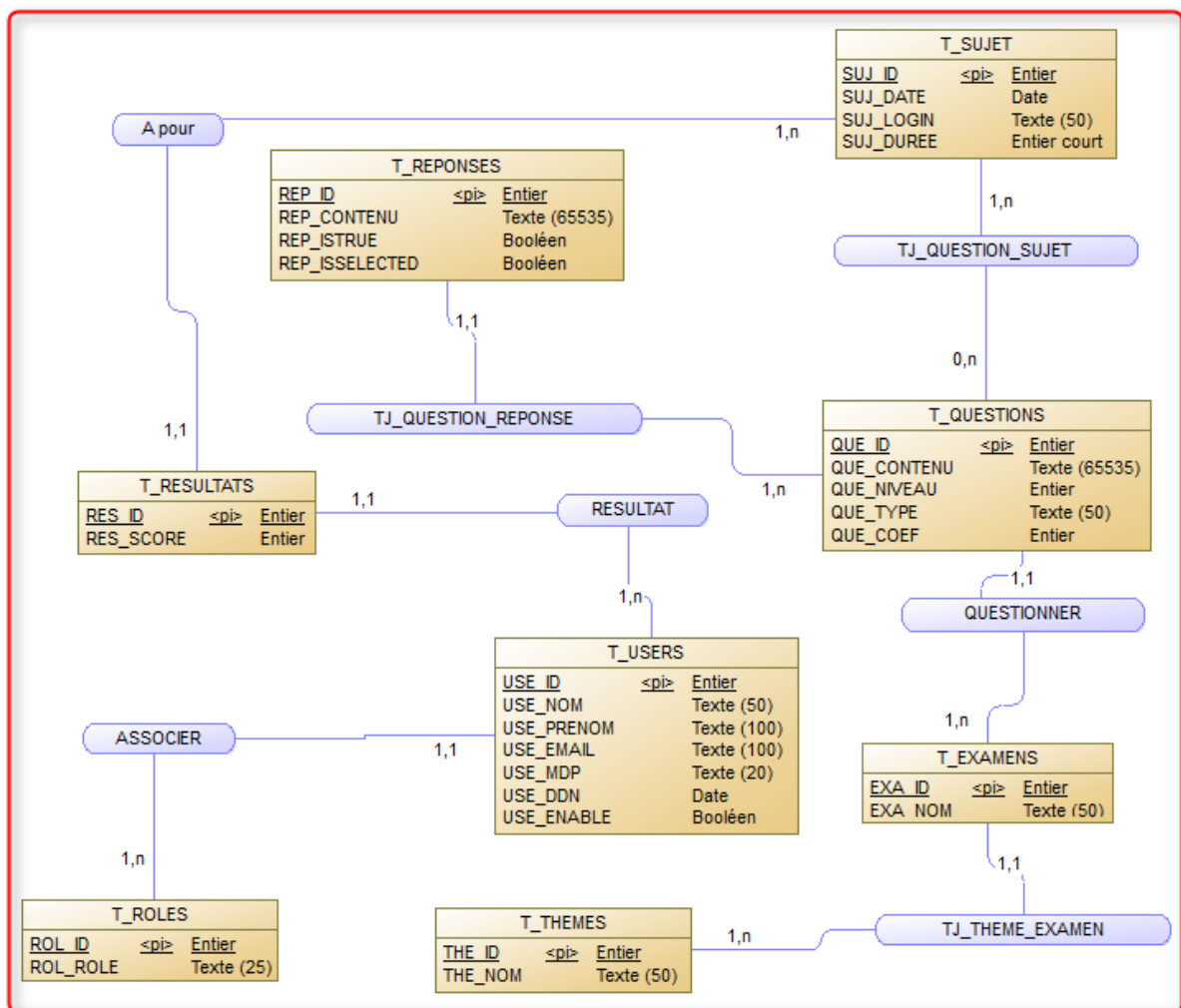


Figure 7 : MCD de TESTENLIGNE

Toutes les entités qui interviennent dans l'application sont persistées en base de données.

MPD (Model Physique de Données)

Le MPD, modèle physique des données nous donne un aperçu de l'implémentation des tables de la base de données à utiliser. Il est représenté sur la figure suivante.

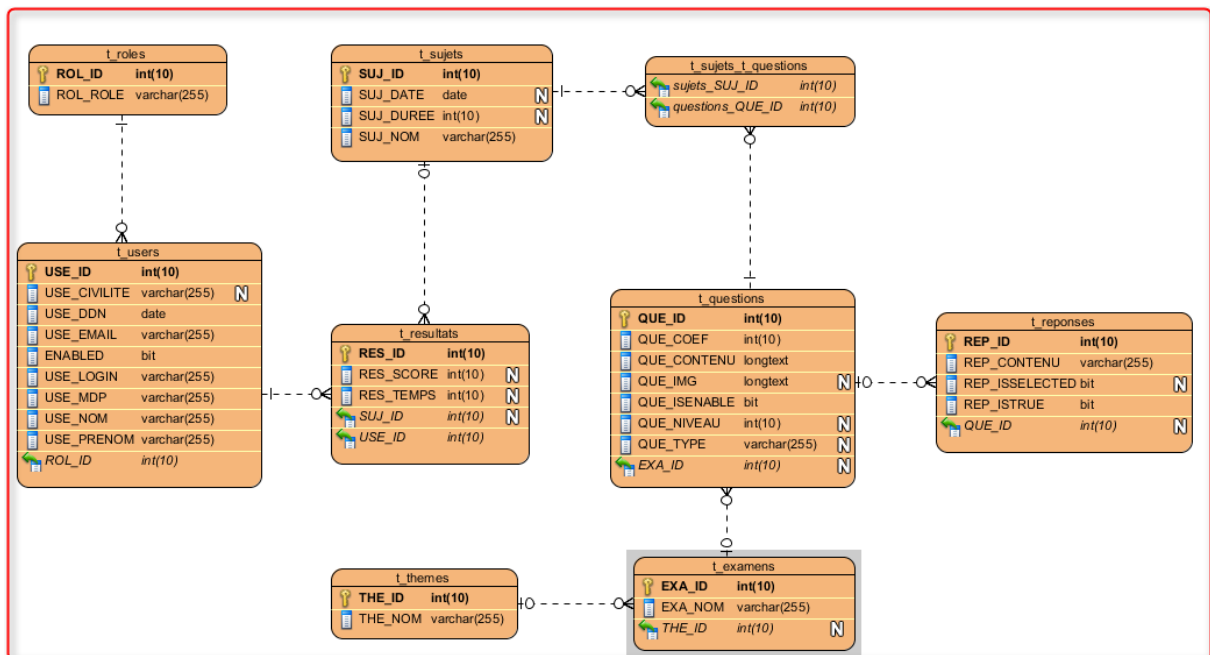


Figure 8 : MPD de TESTENLIGNE

SCRIPT SQL (Voir annexe)

DIAGRAMMES DE SEQUENCE

Afin de dégager les différents échanges qui s'effectuent de façon concrète entre les utilisateurs et l'application, nous avons réalisé les diagrammes suivants, dénommés « DIAGRAMME DE SEQUENCE » dans le formalisme UML.

DIAGRAMME DE SEQUENCE BOITE NOIRE

Ce diagramme va nous permettre d'identifier les différents messages qui sont échangés entre l'application et les acteurs. Pour chaque cas d'utilisation identifié plus haut, réalisons son diagramme de séquence « boîte noire ».

« UC01 » : Se connecter

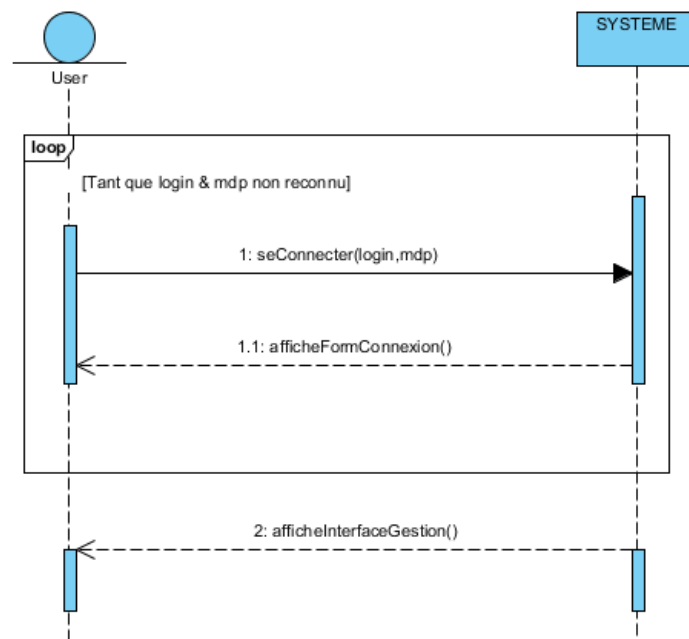


Figure 9 : Boîte noire de l'UC01

Il apparaît dans l'« UC01 », un seul contrat d'opération entre l'utilisateur et le système: *seConnecter (login, mdp)* ;

« UC02 » : Gérer les thèmes

L'analyse du scénario du « UC02 » nous montre que plusieurs cas sont à distinguer. Nous allons présenter sur un même diagramme de séquence « boîte noire », chaque opération réalisable dans ce « use case ».

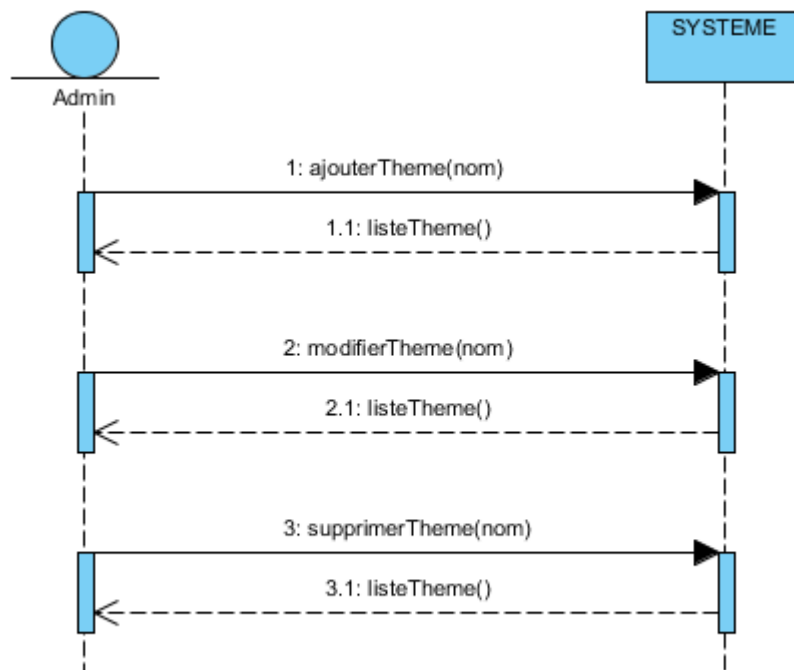


Figure 9 : Boite noire de l'UC02

Le diagramme de la figure 9 met en évidence trois contrats d'opérations dans le « UC02 » : ajouterTheme (nom), modifierTheme (nom), supprimerTheme (nom).

Remarquons que l'UC02 étant semblable aux UC03,..., UC08, nous retrouvons les mêmes opérations avec des appellations différentes. Nous nous abstenons donc de représenter les diagrammes de séquence boîte noire de ces « Use Case ».

« UC09 » : Passer un examen

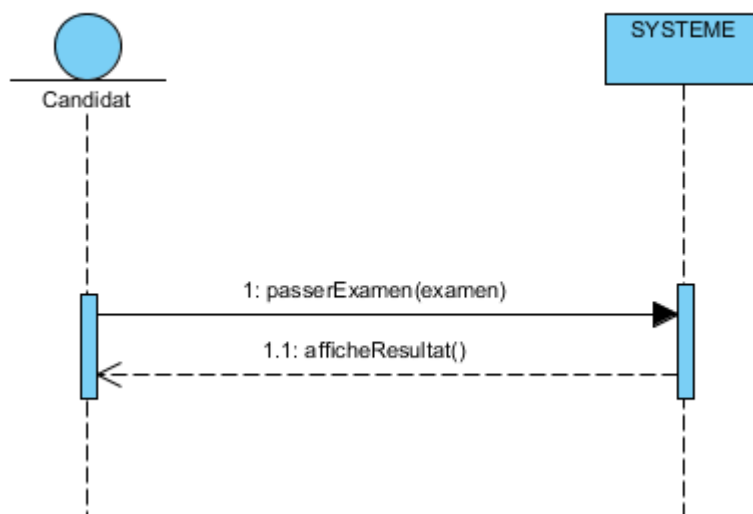


Figure 10 : Boite noire de l'UC09

Un seul contrat est identifié ici : passerExamen(examen)

Tableau récapitulatif des contrats d'opération

OPERATIONS	PRECONDITIONS	POST CONDITIONS
<i>seConnecter(login, mdp)</i>	Aucune	Aucune
<i>ajouter (nom)</i>	Aucune	Un objet « Theme » est créé et persisté dans la BD
<i>modifier (nom)</i>	Aucune	Un objet « Theme » est mis à jour dans la BD
<i>Supprimer (nom)</i>	Aucune	Un objet « Theme » est supprimé de la BD
<i>passerExamen(examen)</i>	Aucune	Un objet « Resultat » est créé et persisté en BD

DIAGRAMME DE SEQUENCE BOITE BLANCHE

Pour chaque opération du tableau récapitulatif précédent, établissons le diagramme de séquence boîte blanche de celle-ci.

seConnecter (login, mdp)

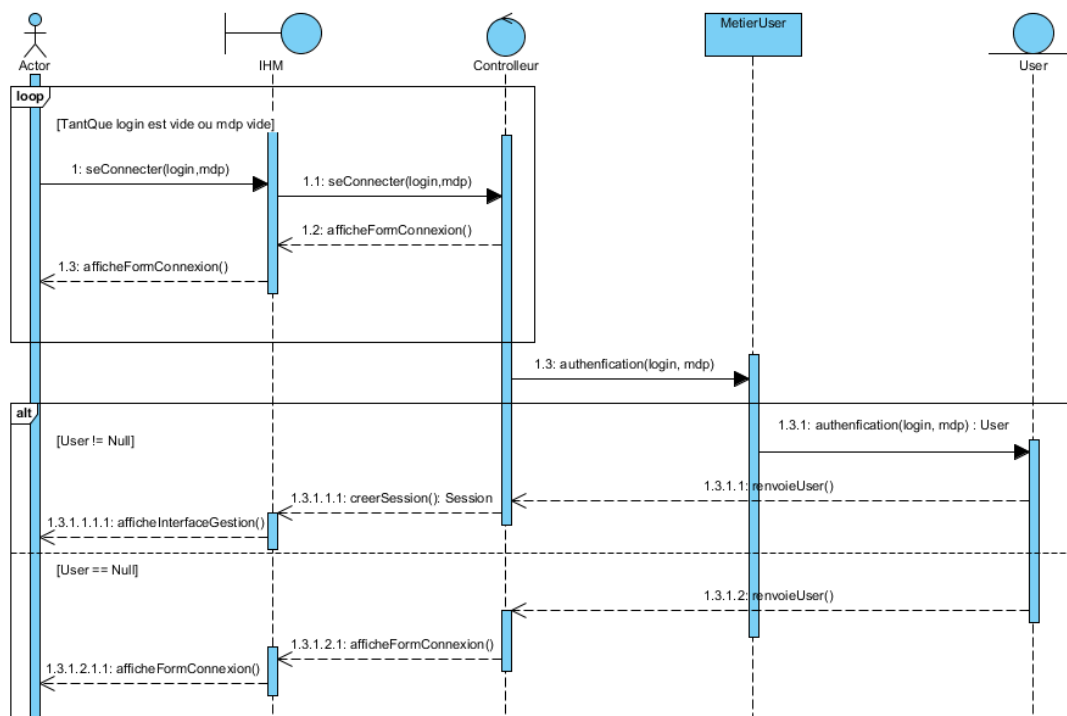


Figure 11 : Boite Blanche seConnecter(login, mdp)

ajouterTheme (nom)

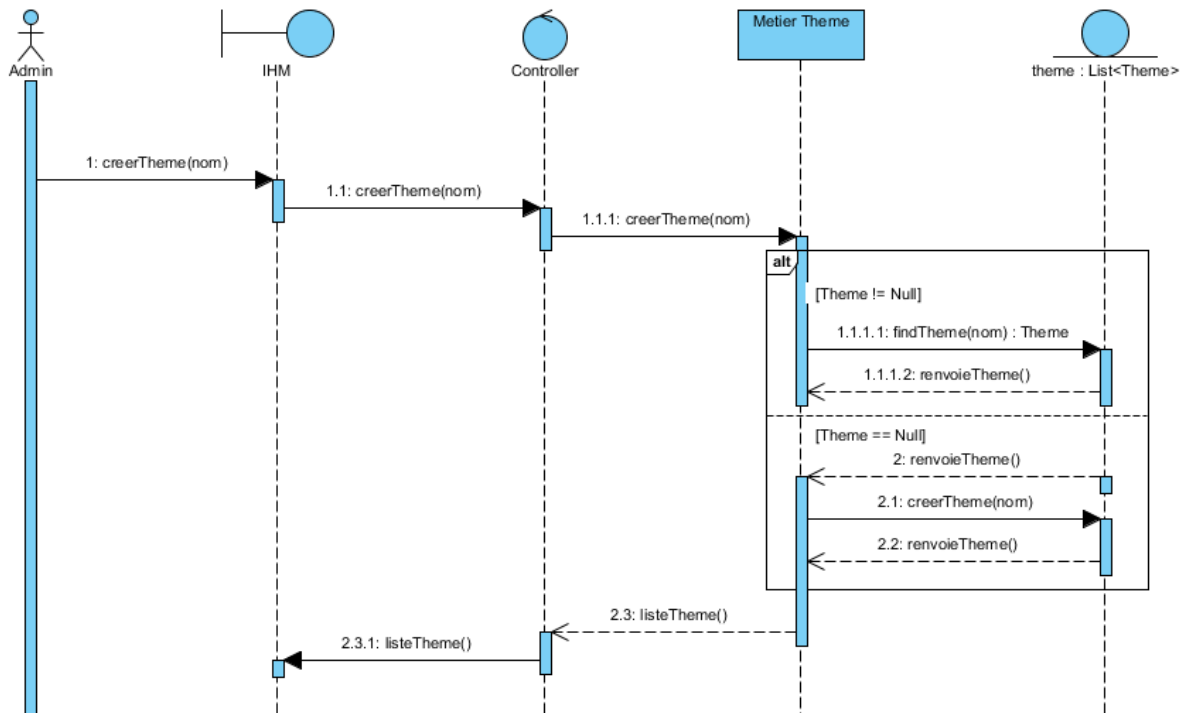


Figure 12 : Boite Blanche ajouterTheme(nom)

modifierTheme(nom)

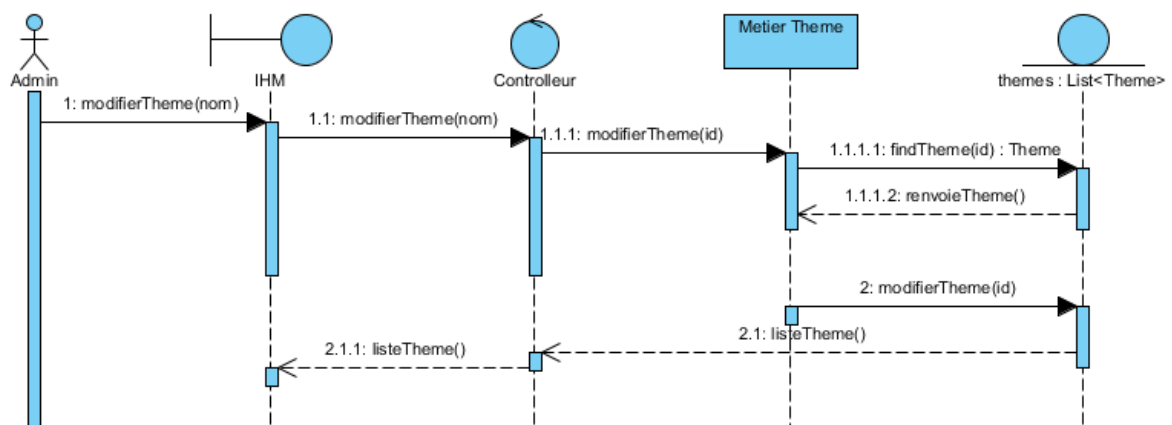


Figure 13 : Boite Blanche modifierTheme(nom)

SupprimerTheme(nom)

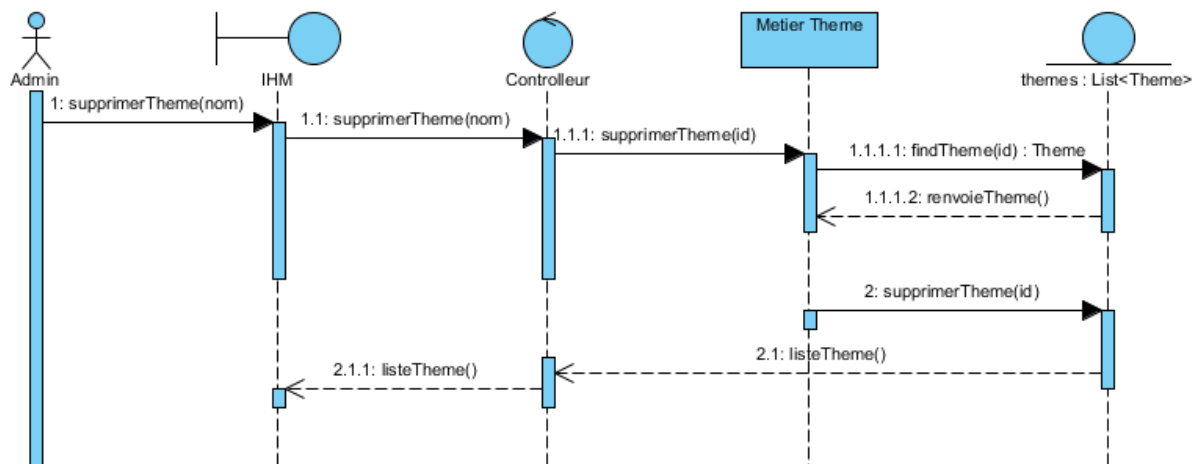


Figure 14 : Boite Blanche supprimerTheme(nom)

passerExamen (examen)

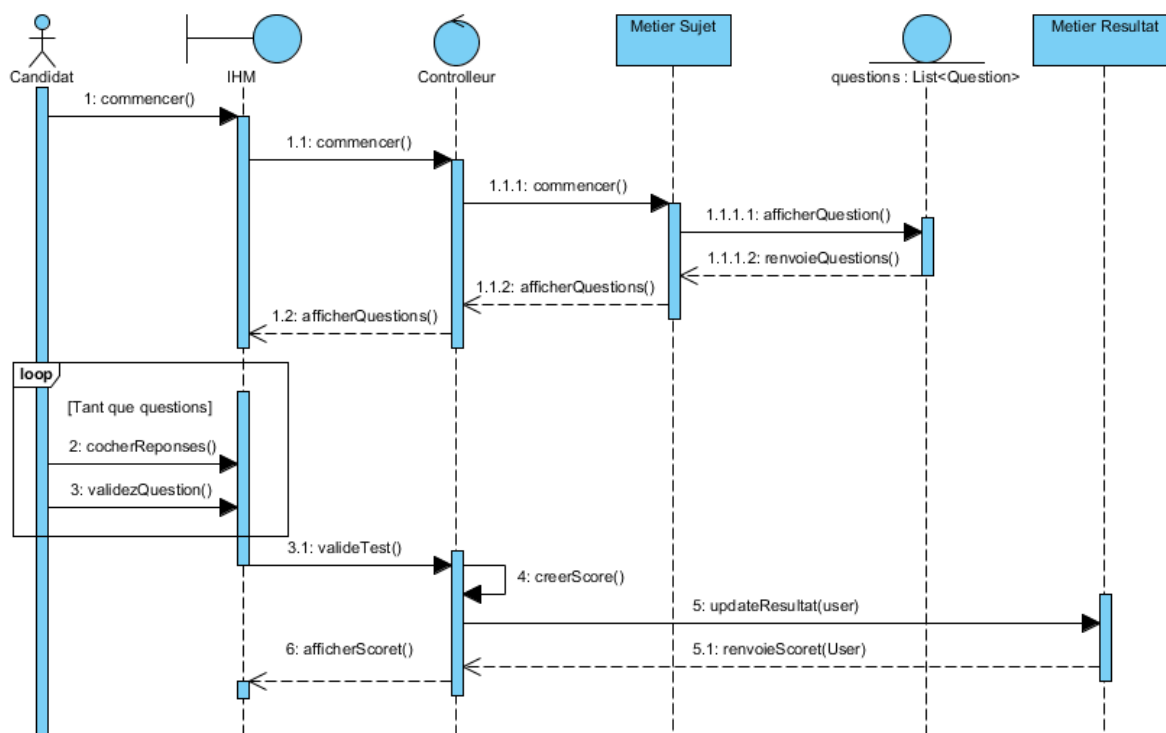


Figure 15 : Boite Blanche passerExamen(examen)

DIAGRAMME DES CLASSES

De ce qui précède, on déduit le diagramme suivant :

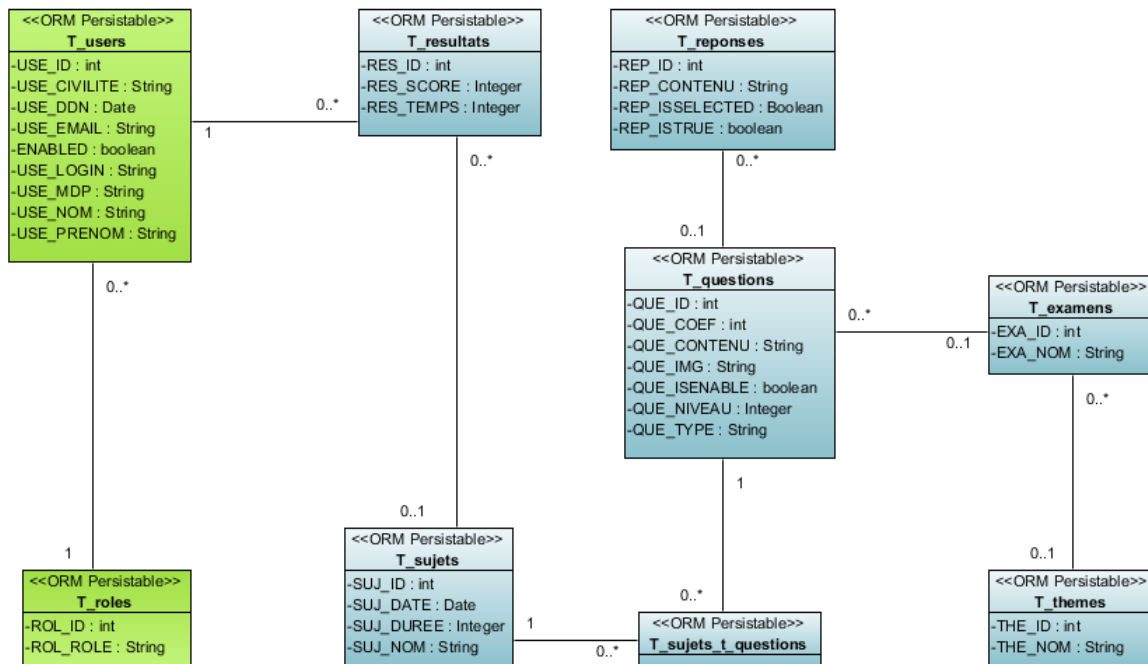


Figure 16 : Diagramme des classes participantes

L'APPLICATION WEB

L'application Web du projet est une IHM (interface Homme-Machine), qui au regard des analyses précédentes est principalement composée de 5 types de pages :

- Une page de connexion au site.
- Une page d'accueil pour l'administration du site. Sur cette page, on trouvera la liste des candidats et leurs résultats.
- Une page pour lister une entité.
- Une page pour la création d'entités.
- Une page pour la modification d'une entité.
- Une page pour le passage du test.

JAVA SERVER FACES

BREVE PRESENTATION

Nous présentons maintenant le Framework Java Server Faces. Ce sera la version 2 qui sera utilisée pour le développement de notre application.

PLACE DE JSF DANS UNE APPLICATION WEB

Le plus souvent, une application web sera bâtie sur une architecture multi-couche telle que la suivante :

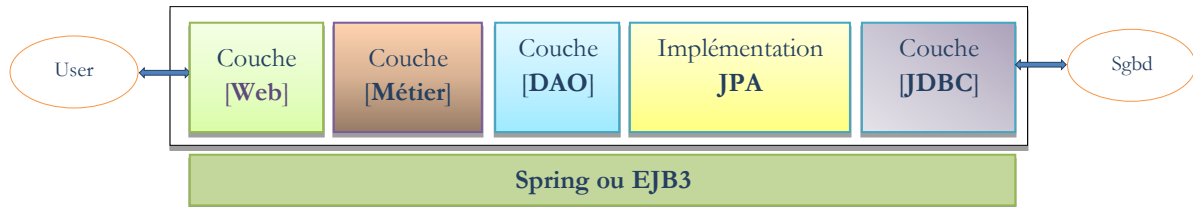


Figure 17 : architecture multi-couche

- la couche [**Web**] est la couche en contact avec l'utilisateur de l'application web. Celui-ci interagit avec l'application web au travers de pages web visualisées par un navigateur. C'est dans cette couche que se situe JSF et uniquement dans cette couche.
- La couche [**Métier**] implémente les règles de gestion de l'application, tels que le calcul d'une note ou l'âge d'un utilisateur. Cette couche utilise des données provenant de l'utilisateur via la couche [**Web**] et du Sgbd via la couche [**DAO**]
- La couche [**DAO**] (Data Access Objects), la couche [**JPA**] (Java Persistence Api) et le pilote **JDBC** gèrent l'accès aux données du Sgbd. La couche [**JPA**] sert d'ORM (Object Relational Mapper). Elle fait un pont entre les objets manipulés par la couche [**DAO**], les lignes et les colonnes des données d'une base de données relationnelle.
- L'intégration des couches peut être réalisée par un conteneur **Spring** ou **EJB3** (Enterprise Java Bean).

LE MODELE DE DEVELOPPEMENT MVC DE JSF

JSF implémente le design pattern MVC (Modèle-Vue-Contrôleur) de la façon suivante :

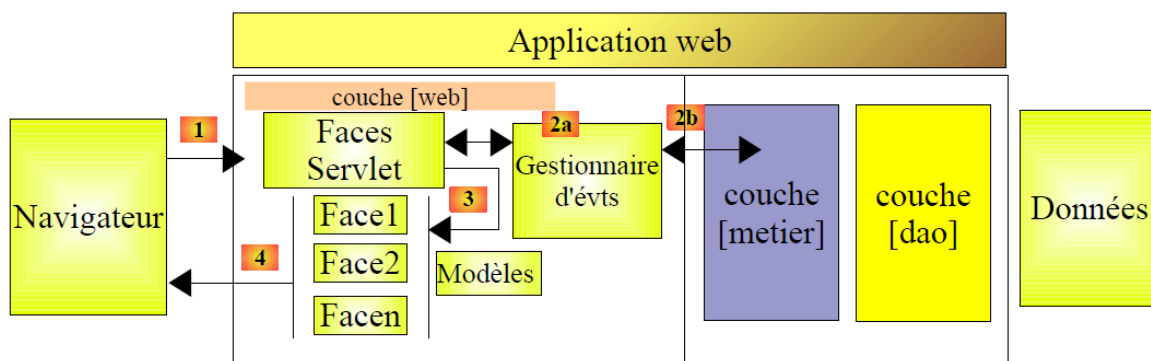


Figure 18 : Architecture JSF

Le traitement d'une demande d'un client se déroule selon les quatre(4) étapes suivantes :

1. **Demande** : le client navigateur fait une demande au contrôleur [Faces Servlet]. Celui-ci voit passer toutes les demandes des clients. C'est la porte d'entrée de l'application. C'est le **C** de MVC.
2. **Traitement** : le contrôleur C traite cette demande. Pour ce faire, il se fait aider par des gestionnaires d'événements spécifiques à l'application écrite [2a]. Ces gestionnaires peuvent avoir besoin de l'aide de la couche métier [2b]. Une fois la demande du client traitée, celle-ci peut appeler diverses réponses. Un exemple classique est :
 - une page d'erreurs si la demande n'a pu être traitée correctement
 - une page de confirmation sinon
3. **Navigation** : le contrôleur choisit la réponse (= Vue) à envoyer au client. Choisir la réponse à envoyer au client nécessite plusieurs étapes :
 - choisir la Facelet qui va générer la réponse. C'est ce qu'on appelle la vue **V**, le V de MVC. Ce choix dépend en général du résultat de l'exécution de l'action demandée par l'utilisateur.
 - fournir à cette Facelet les données dont elle a besoin pour générer cette réponse. En effet, celle-ci contient le plus souvent des informations calculées par le contrôleur. Ces informations forment ce qu'on appelle le modèle **M** de la vue, le M de MVC.L'étape 3 consiste donc en le choix d'une vue V et en la construction du modèle M nécessaire à celle-ci.
4. **Réponse** : le contrôleur C demande à la Facelet choisie de s'afficher. Celle-ci utilise le modèle M préparé par le contrôleur C pour initialiser les parties dynamiques de la réponse qu'elle doit envoyer au client. La forme exacte de celle-ci peut être diverse : ce peut être un flux HTML, PDF, Excel, ...

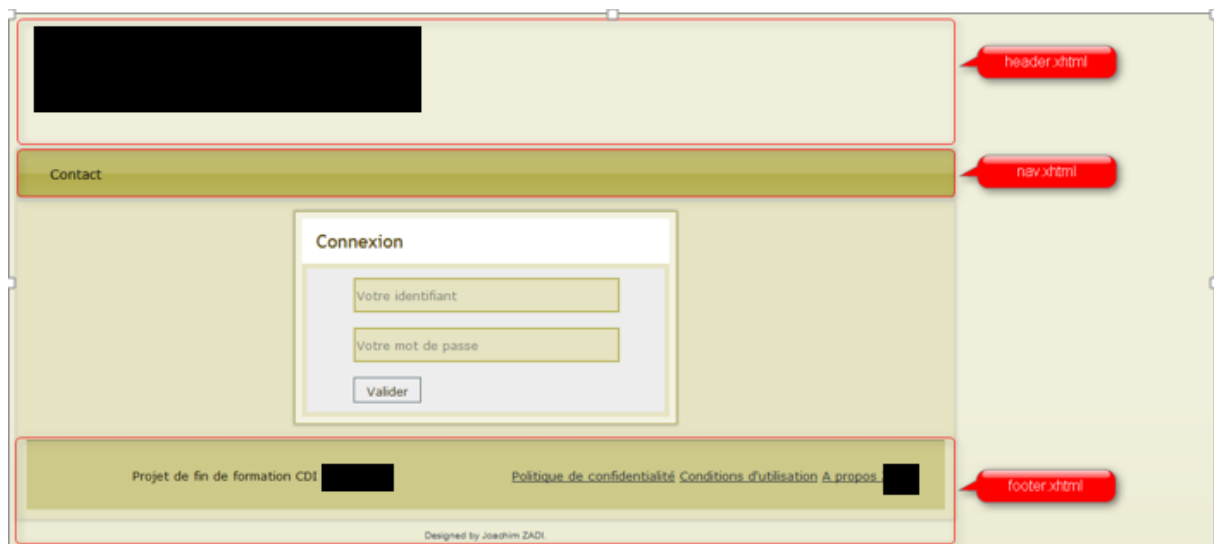
Dans un projet JSF :

- le contrôleur C est la servlet [javax.faces.webapp.FacesServlet].
- les vues V sont implémentées par des pages utilisant la technologie des Facelets
- les modèles M et les gestionnaires d'événements sont implémentés par des classes Java souvent appelées "Backing Bean" ou plus simplement Beans.

MAQUETTE DES PAGES DE L'APPLICATION

LE TEMPLATE DES PAGES

L'application « TESTENLIGNE » sera découpée selon le modèle suivant avec un entête, un menu de navigation et un pied de page.



Page de login

Représentée par la page « login.xhtml », cette page est l'entrée de l'application. Elle permet à un utilisateur de se connecter à l'application.

The screenshot shows a web application interface. At the top, there's a black rectangular area. Below it, a green header bar contains the word 'Contact'. The main content area features a white box titled 'Connexion'. Inside this box, there are two input fields: 'Votre identifiant' and 'Votre mot de passe', followed by a 'Valider' button. The footer of the page is green and contains the text 'Projet de fin de formation CDI' followed by a black rectangle, and a series of links: 'Politique de confidentialité', 'Conditions d'utilisation', 'A propos', followed by another black rectangle.

Tables utilisées dans la page login

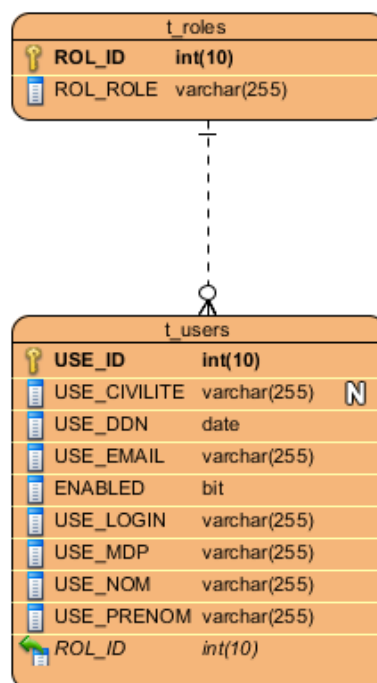


Figure 19 : Tables utilisées pour l'authentification

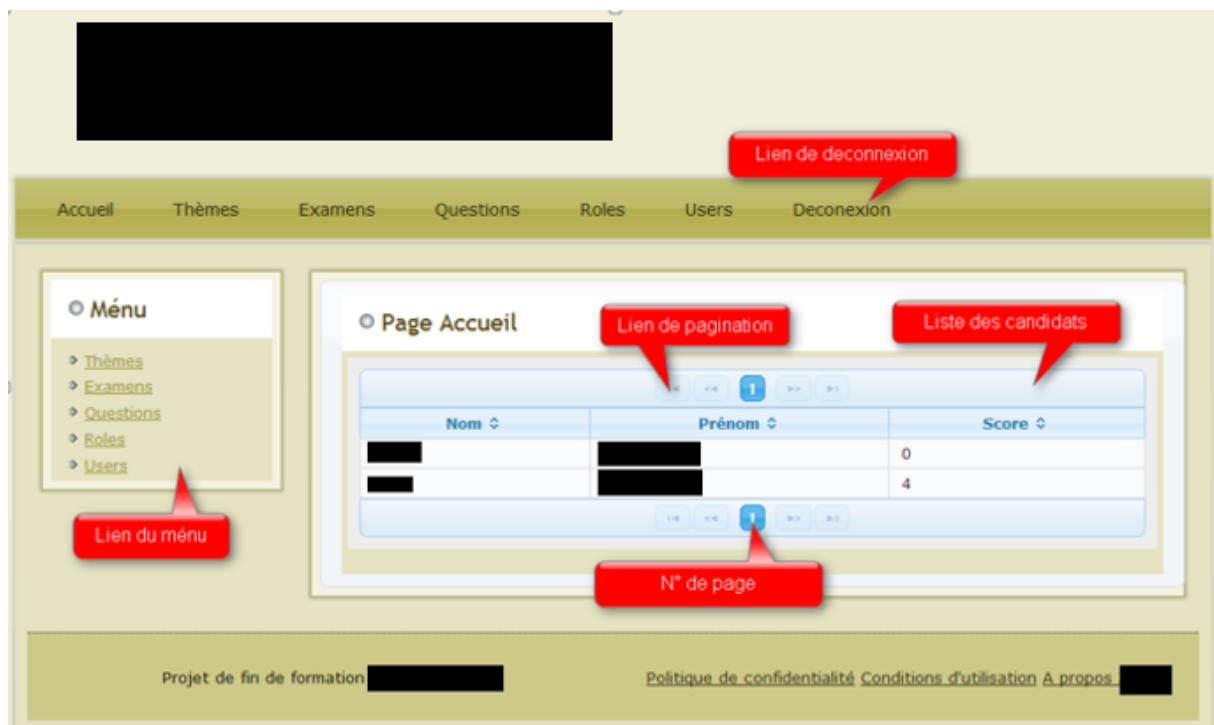
Représentation des données maquette/tables

Colonne Table	Objet graphique	Définition	Structure
USE_LOGIN	InputText	Votre identifiant	varchar(255)

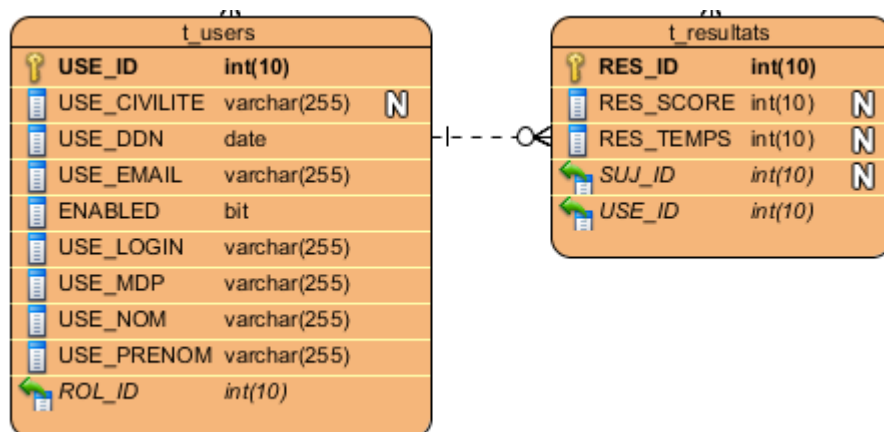
USE_MDP	InputText	Votre mot de passe	varchar(255)
---------	-----------	--------------------	--------------

Page accueil admin

Cette page est représentée par la « index.xhtml » dans les ressources de l'administrateur. C'est la page vers laquelle est redirigé tout utilisateur ayant les droits « ROLE_ADMIN ».



Tables utilisées dans la page accueil admin

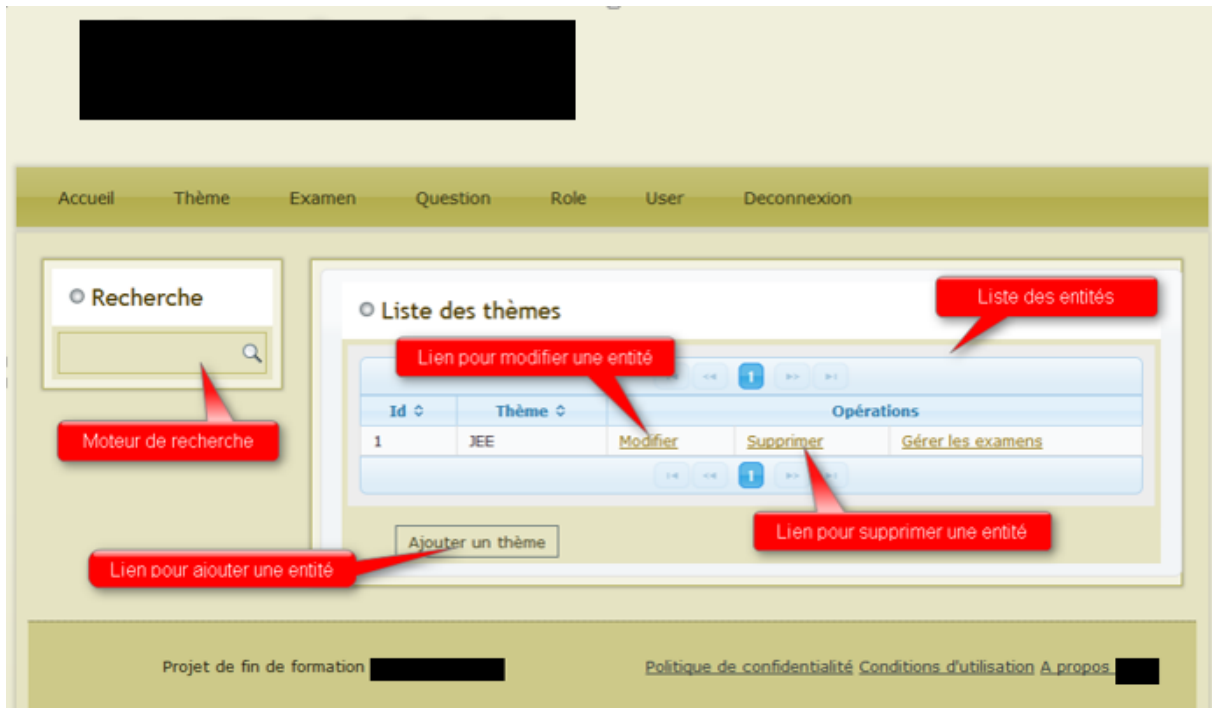


Représentation des données maquette/tables



Colonne Table	Objet graphique	Définition	Structure
USE_NOM	Table	Liste des thèmes	varchar(255)
USE_PRENOM			varchar(255)
RES_SCORE			int(10)

Page de liste d'entité

Cette page représente le modèle choisi pour lister des entités. Elle est représentée par les pages « liste.xhtml ». Nous présentons à titre d'exemple, la page de « liste.xhtml » de l'entité « Theme ».



Tables utilisées dans la page liste.xhtml

t_themes	
 THE_ID	int(10)
 THE_NOM	varchar(255)

Représentation des données maquette/tables

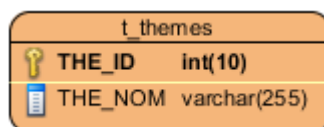
Colonne Table	Objet graphique	Définition	Structure
THE_NOM	InputText	Recherche	varchar(255)

Page de création d'entité

Cette page représente le modèle choisi pour créer une nouvelle entité. Elle est représentée par les pages « create.xhtml ». Nous présentons à titre d'exemple, la page de « create.xhtml » de l'entité « Theme » afin de ne pas être répétitif. Toutes les pages des autres entités sont conçues de la même manière.



Tables utilisées dans la page create



Représentation des données maquette/tables

Colonne Table	Objet graphique	Définition	Structure
THE_NOM	InputText	NOM	varchar(255)

Page de modification d'entité

Cette page représente le modèle choisi pour modifier une nouvelle entité. Elle est représentée par les pages « update.xhtml ». Nous présentons à titre d'exemple, la page de « update.xhtml » de l'entité « Theme ».

Accueil Thème Examen Question Role User Deconnexion

○ Modifier un thème

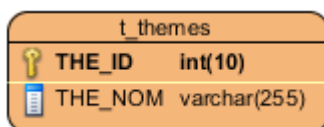
Nouveau thème

JEE

Valider

Projet de fin de formation Politique de confidentialité Conditions d'utilisation A propos

Tables utilisées dans la page update.xhtml



Représentation des données maquette/tables

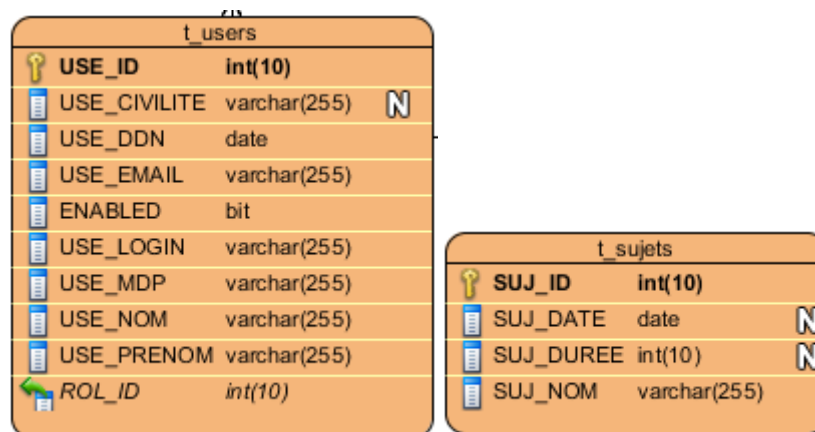
Colonne Table	Objet graphique	Définition	Structure
THE_NOM	InputText	NOM	varchar(255)

Page accueil user

Cette page est représentée par la page « index.xhtml » dans les ressources de l'utilisateur candidat. C'est la page vers laquelle est redirigé tout utilisateur ayant les droits « ROLE_USER ». Un message d'accueil à l'intention du candidat y est inscrit.



Tables utilisées

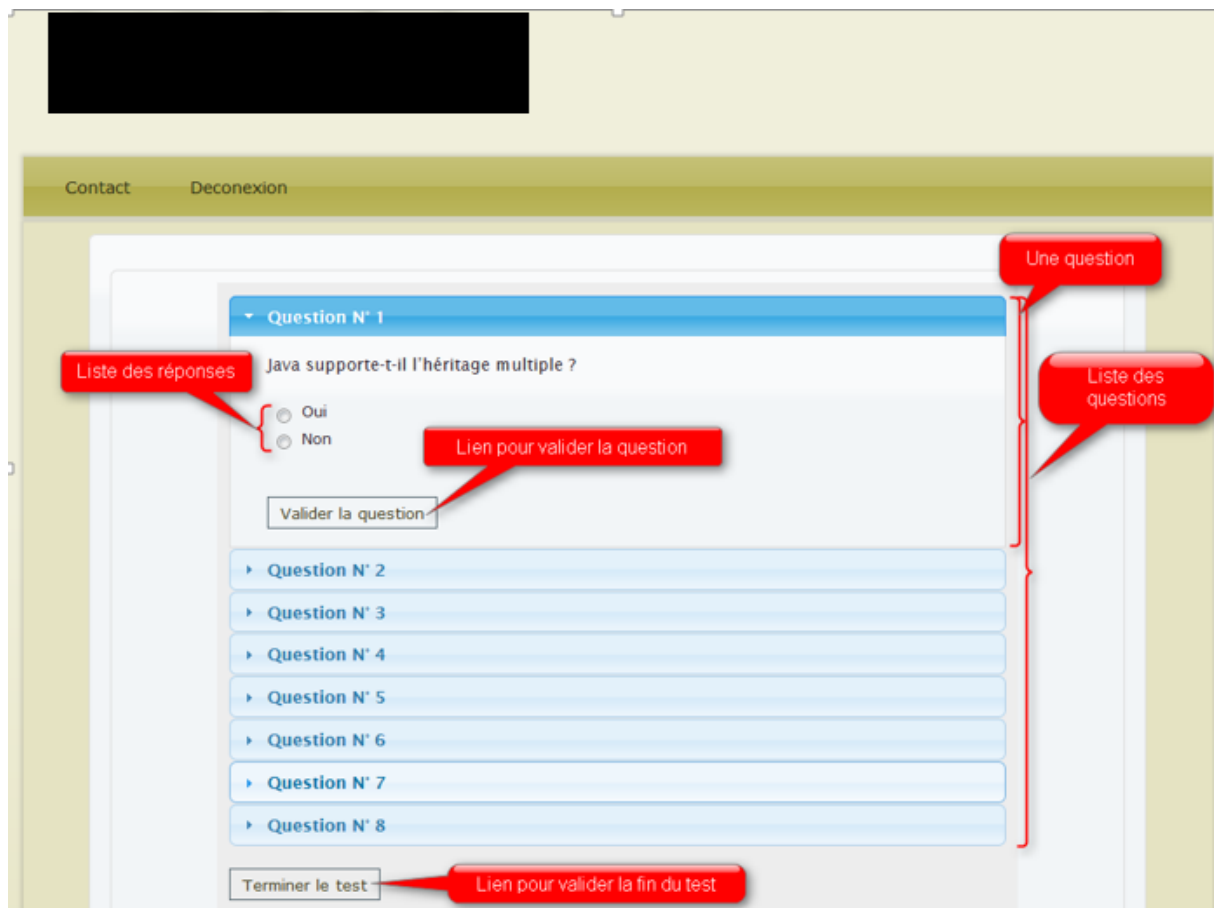


Représentation des données maquette/tables

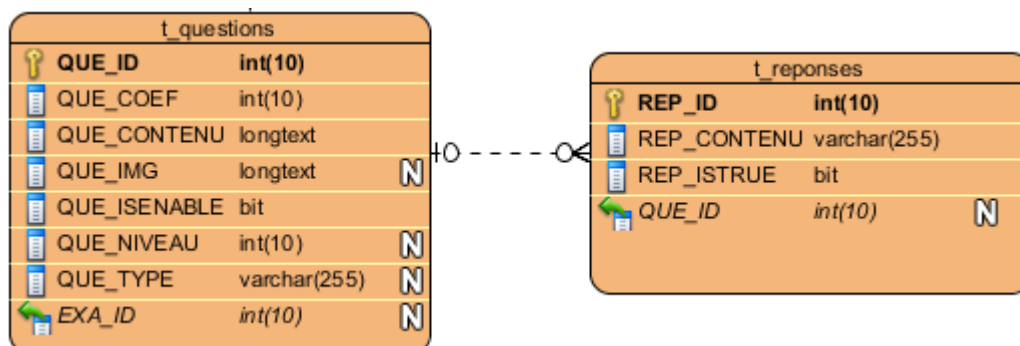
Colonne Table	Objet graphique	Définition	Structure
USE_NOM	OutputText		varchar(255)
USE_PRENOM	OutputText		varchar(255)
SUJ_DUREE	OutputText		int(10)

Page de test

Cette page est dédiée au passage du test. Elle contient les questions. La présentation des questions est sous la forme d'accordéon. Pour voir le contenu de la question, on déroule l'accordéon correspondant.



Tables utilisées

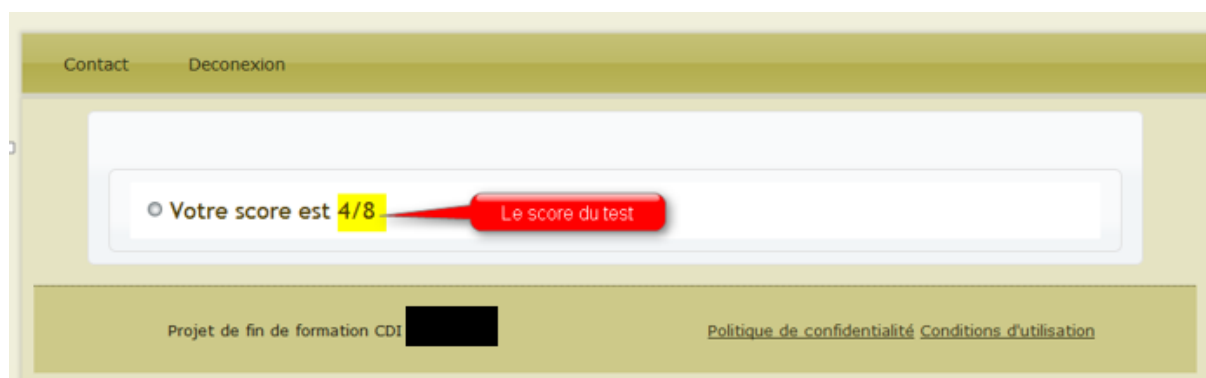


Représentation des données maquette/tables






Colonne Table	Objet graphique	Définition	Structure
QUE_CONTENU	Accordéon	Question	longtext
REP_CONTENU	Radio	Réponse	varchar(255)

Page de résultat

Cette page affiche le résultat à la fin du test



Tables utilisées

t_resultats			
	RES_ID	int(10)	
	RES_SCORE	int(10)	N
	RES_TEMPS	int(10)	N
	SUJ_ID	int(10)	N
	USE_ID	int(10)	

Représentation des données maquette/tables

Colonne Table	Objet graphique	Définition	Structure
RES_SCORE	OutputText	Question	int(10)

DIAGRAMME DE NAVIGATION

Le diagramme suivant nous montre les différents cas de navigation qu'offre l'application.

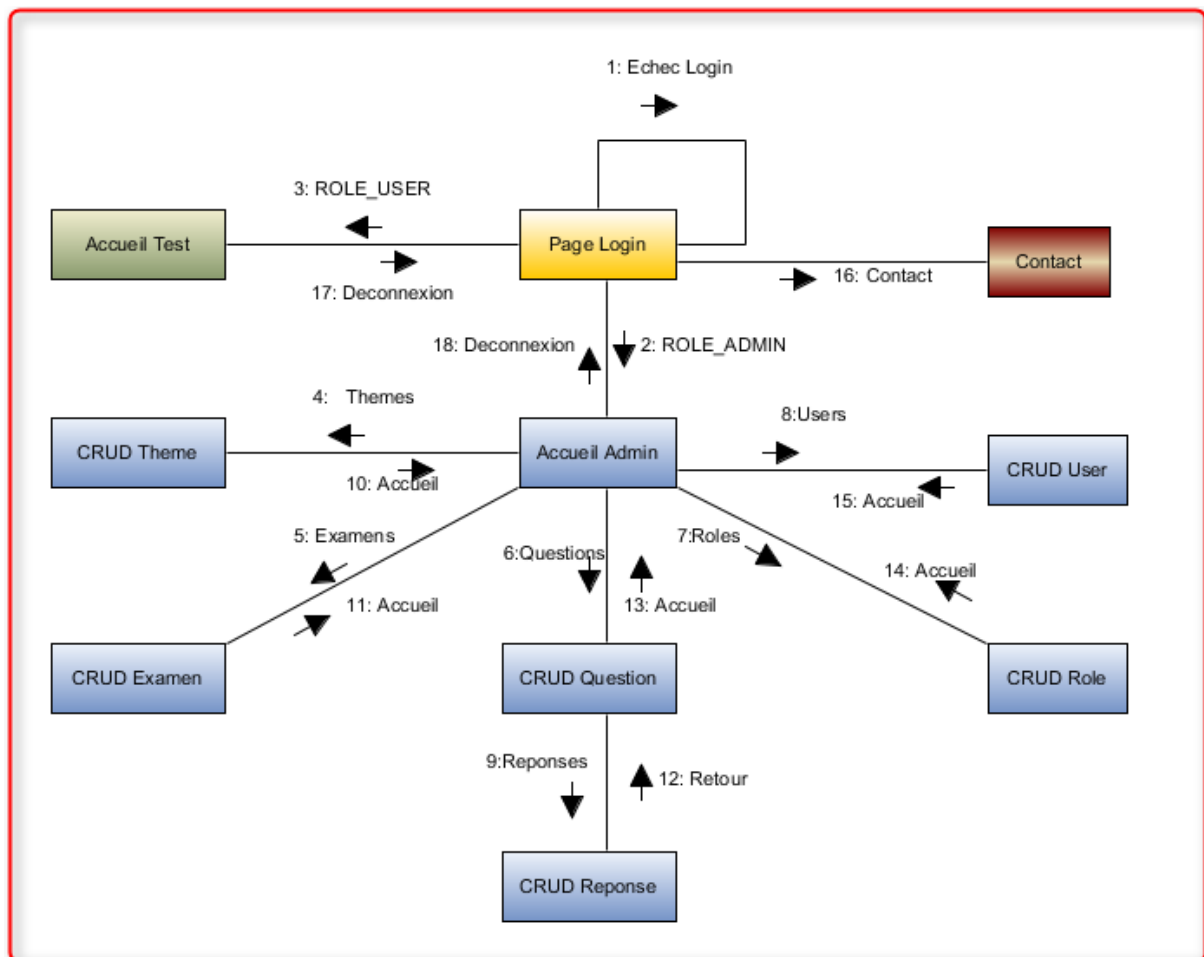
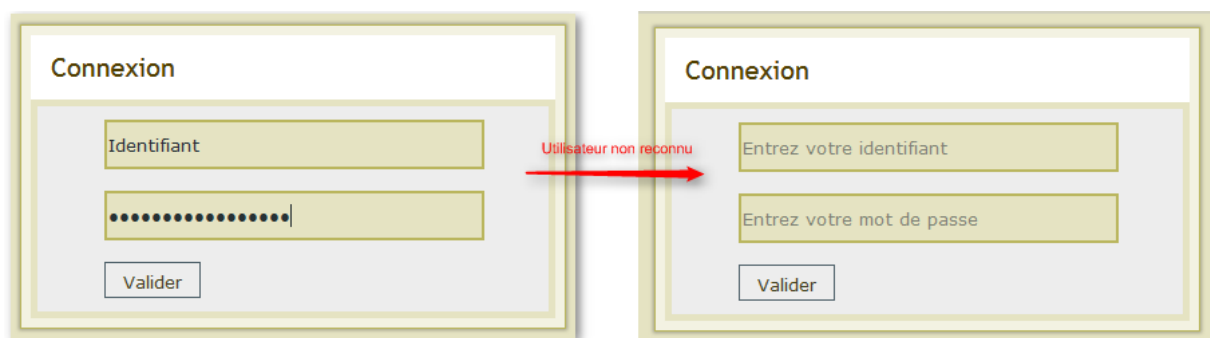


Figure 20: Diagramme de navigation

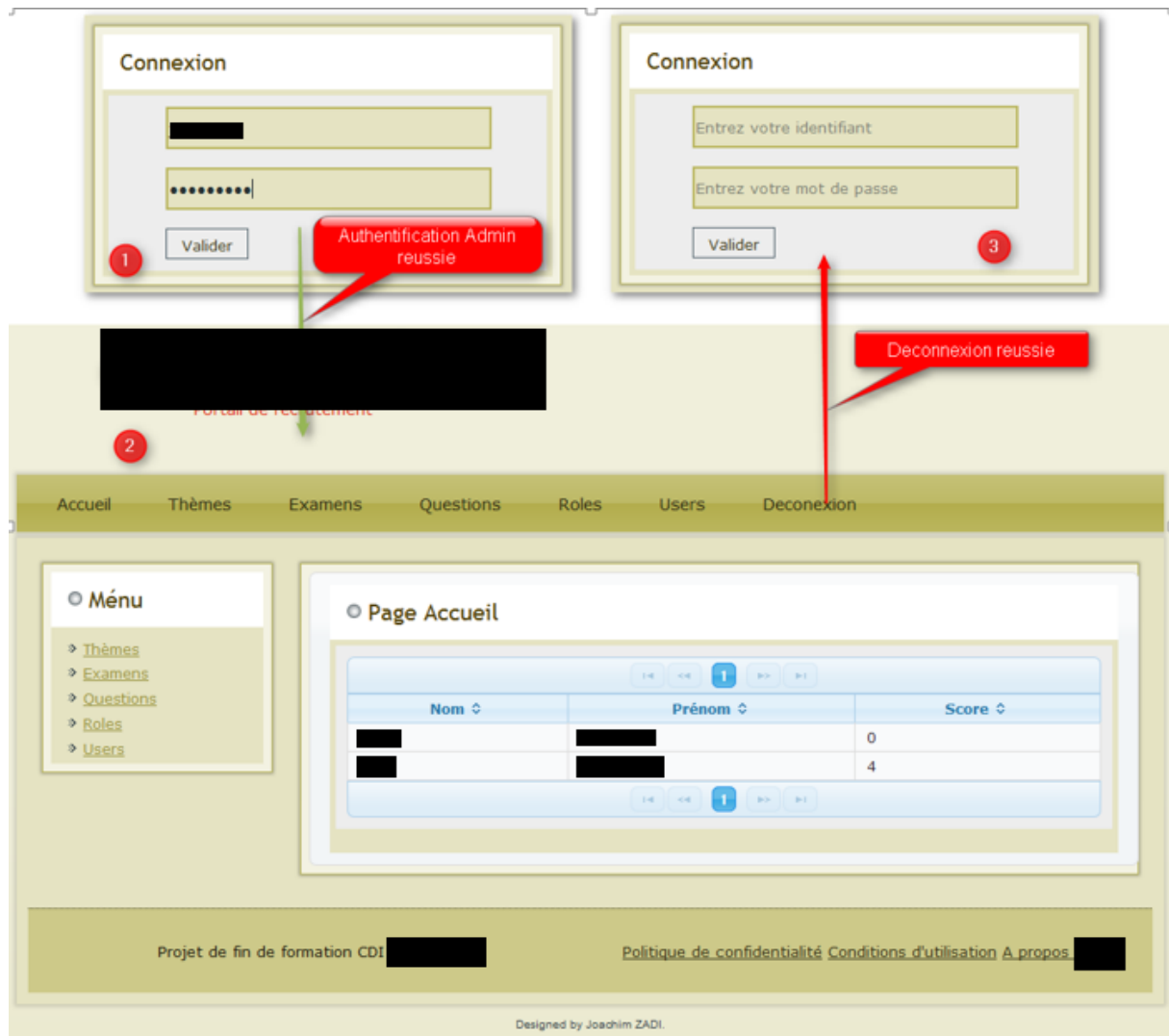
Cas 1 : Echec de login

Le cas de navigation suivant est les cas dans lequel un utilisateur n'est pas reconnu par le système.



Cas 2 : Login admin

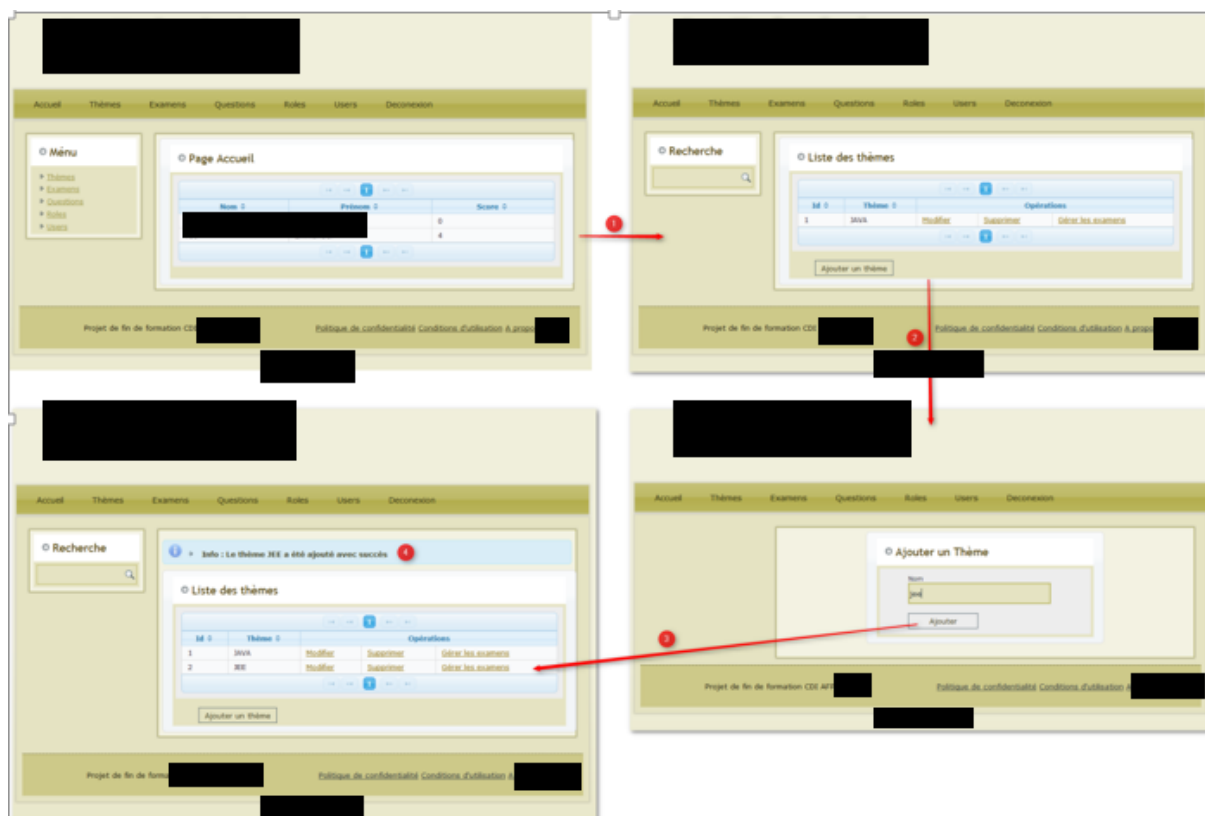
Le présent cas présente la navigation d'un utilisateur ayant le ROLE_ADMIN, qui a été reconnu par le système.



- 1 : L'utilisateur Admin entre son login et mdp. Il valide et est reconnu par le système
- 2 : La page d'accueil de l'administration des tests. Pour se déconnecter du système, on clique sur le lien « [Déconnexion](#) ».
- 3 : En cas de déconnexion réussie, la page de login s'affiche de nouveau

Cas 3 : Créer un thème

Pour ce cas, nous traiterons seulement l'exemple de la manipulation de l'entité thème.



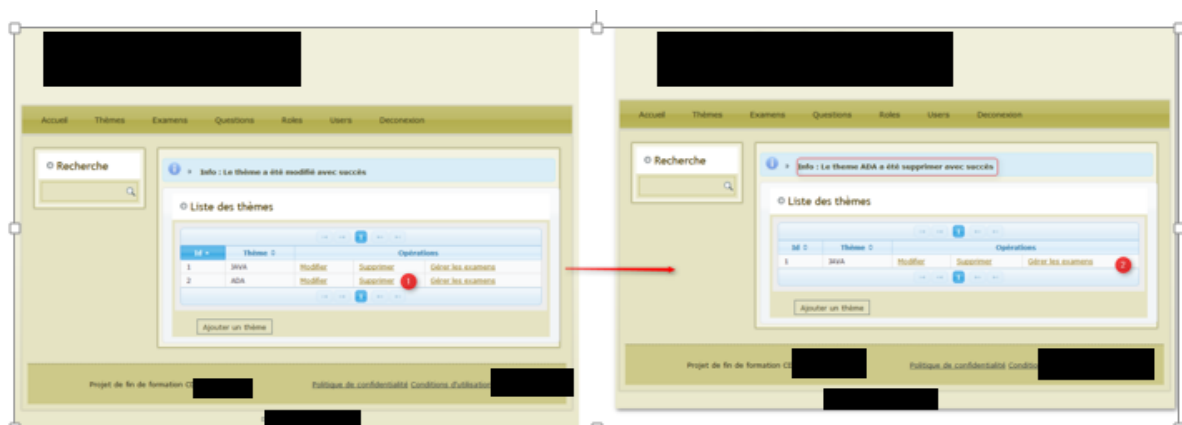
- 1 : L'administrateur clique sur le lien « [thèmes](#) » pour avoir accès à la console de gestion des thèmes. Cette page affiche la liste des thèmes existant dans le système.
- 2 : L'utilisateur clique sur « [Ajouter un thème](#) ».
- 3 : L'utilisateur saisi le nom du nouveau thème. Il valide sa création en cliquant sur le lien « [Ajouter](#) »
- 4 : La console de gestion des thèmes est à nouveau réaffichée avec la liste des thèmes mis à jour. Un message d'information indique la réussite de l'opération.

Cas 4 : Modifier un thème



- 1 : L'administrateur clique sur le lien « Modifier » pour avoir accès à la console de modification des thèmes.
- 2 : La console de modification est affichée avec le nom de l'ancien thème.
- 3 : L'utilisateur saisi le nom du nouveau thème. Il valide sa création en cliquant sur le lien « Valider »
- 4 : La console de gestion des thèmes est à nouveau réaffichée avec la liste des thèmes mis à jour. Un message d'information indique la réussite de l'opération.

Cas 5 : Supprimer un thème



- 1 : L'administrateur clique sur le lien « [Supprimer](#) » pour supprimer le thème ADA
- 2 : La console de gestion des thèmes est à nouveau réaffichée avec la liste des thèmes mis à jour. Un message d'information indique la réussite de l'opération.

Remarque

Nous venons de montrer le cas de navigation du CRUD thème. Ce cas étant semblable aux cas de gestion des entités, nous nous abstenons de les réaliser dans ce rapport.

Cas 6 : Login user



- 1 : Le candidat saisi son login et mot de passe. Il est reconnu par le système
- 2 : La console de bienvenue s'affiche avec le nom et prénom du candidat. Elle lui mentionne le temps qui lui est impartie pour passer le test.
- 3 : Le candidat clique sur le lien « [Déconnexion](#) » et est redirigé vers la page de login.

Cas 7 : Passer le test



- 1 : Le candidat saisi son login et mot de passe. Il est reconnu par le système
- 2 : La console de bienvenue s'affiche avec le nom et prénom du candidat. Elle lui mentionne le temps qui lui est imparti pour passer le test.
- 3 : Le candidat clique sur le lien « [Commencer](#) » pour commencer le test. Il est redirigé vers la page de contenant les questions.
- 5 : Le candidat coche la réponse à la question
- 6 : Le candidat valide sa réponse en cliquant sur le lien « [Validez la question](#) »
- 7 : La question validée n'est plus disponible.
- 8 : Une fois le questionnaire terminé, le candidat valide son test en cliquant sur « [terminer le test](#) »

9 : Le score est alors affiché. Le candidat ne sera plus autorisé à se connecter.

DESCRIPTION DU MENU

Menu	Description
<i>Contact</i>	<i>Ce menu permet à un utilisateur d'envoyer un message à l'administrateur pour signaler un problème.</i>
<i>Accueil</i>	<i>Il permet à l'utilisateur de revenir sur la page d'accueil du site.</i>
<i>Thèmes</i>	<i>Il permet à l'utilisateur d'afficher la page de gestion de l'entité Thème.</i>
<i>Examens</i>	<i>Permet à l'utilisateur d'afficher la page de gestion de l'entité Examen.</i>
<i>Questions</i>	<i>Permet à l'utilisateur d'afficher la page de gestion de l'entité Question.</i>
<i>Roles</i>	<i>Permet à l'utilisateur d'afficher la page de gestion de l'entité Rôle.</i>
<i>Users</i>	<i>Permet à l'utilisateur d'afficher la page de gestion de l'entité User.</i>
<i>Déconnexion</i>	<i>Permet à l'utilisateur de mettre fin à une session.</i>

DEVELOPPEMENT

DESCRIPTION TECHNIQUE DE L'APPLICATION

Les principales exigences techniques imposées à l'application ONLINETEST sont les suivantes :

- Utilisation du design Pattern MVC2 afin de simplifier la mise en œuvre des évolutions et la montée en puissance.
- Accès à l'application à travers une interface Web.
- Définition de rôles utilisateurs afin de distinguer les fonctions d'administration de celle du passage de test.
- Les informations sont hébergées dans une base de données relationnelle nommée « stage » et l'application y accède à travers une couche logicielle ORM.

Pour renforcer la structuration de l'application, la mise en œuvre des modules Spring dans chacune des couches architecturales de l'application est requise.

Les configurations de ces différents modules restent séparées les unes des autres pour conserver autant d'autonomie que possible à chacune des composantes de l'application.

MISE EN PLACE DU PROJET

Arborescence générale du projet

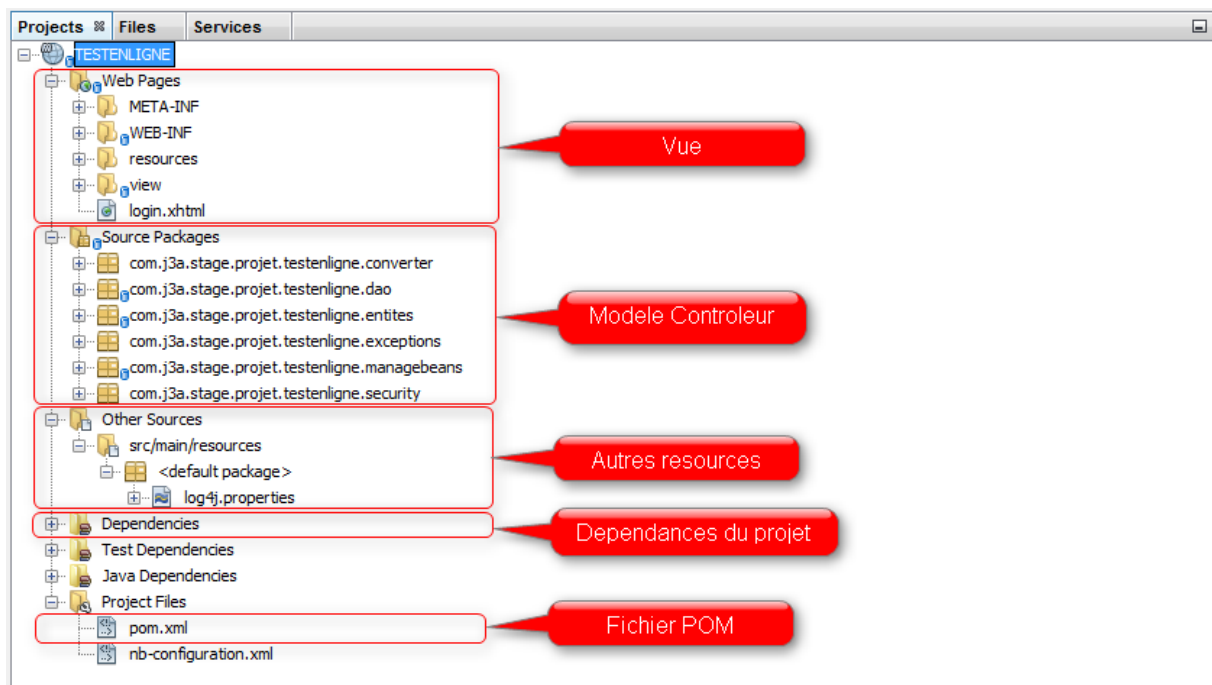


Figure 21 : Arborescence globale du projet

Arborescence des composants du modèle.

Les composants du modèle sont représentés par les classes entités du diagramme des classes mis en place lors de la modélisation. On distingue huit(8) entités développées dans le package « *com.j3a.stage.projet.testenligne.entites* ».

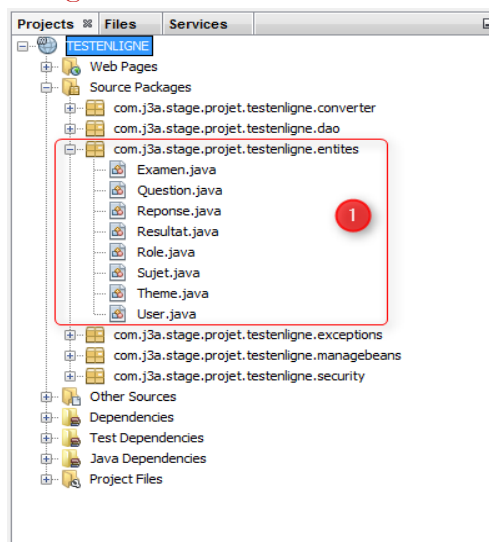


Figure 22 : Les classes entités de l'application

Les éléments du modèle sont mappés sur la base de données via les annotations JPA qui utilise Hibernate comme moteur de persistance (voir figure suivante) :

```

@Entity
@Table(name = "T_THEMES")
@NamedQueries({
    @NamedQuery(name = "Theme.findByNom", query = "SELECT t FROM Theme t WHERE t.nom LIKE :nom")
})

public class Theme implements Serializable {

    private static final long serialVersionUID = 1L;

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "THE_ID")
    private Integer id;

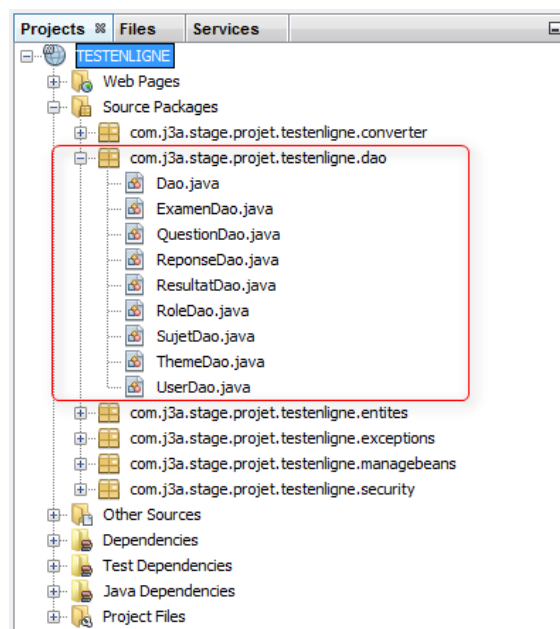
    @Column(name = "THE_NOM", nullable = false, unique = true)
    private String nom;

    @OneToMany(mappedBy = "theme", fetch = FetchType.LAZY)
    private List<Examen> examens;

```

Arborescence des composants DAO.

Les Dao exposent les services d'accès aux informations de la base de données : recherche et chargement des données, ajout, modification et suppression. Ces composants sont souvent associés aux traitements des données par l'acronyme CRUD (Create Read Update Delete). Ils sont développés dans le package « *com.j3a.stage.projet.testenligne.dao* ».



Nous exposons ici un extrait de la classe mère des composants DAO utilisés qui illustre de façon factorielle, les méthodes CRUD à spécialiser par les classes filles.

```
public abstract class Dao<T extends Serializable> {

    private Class<T> classe;

    protected abstract EntityManager getEm();

    public void setClasse(Class<T> classe) {
        this.classe = classe;
    }

    @Transactional
    public void create(T entity) {
        getEm().persist(entity);
    }

    @Transactional
    public void remove(T entity) {
        getEm().remove(getEm().merge(entity));
    }

    @Transactional
    public T update(T entity) {
        return getEm().merge(entity);
    }

    @Transactional(readOnly = true)
    public T getById(Integer id) {
        return getEm().find(classe, id);
    }

    @Transactional(readOnly = true)
    public List<T> getAll() {
        String requete = "SELECT t FROM " + classe.getSimpleName() + " t";
        Query query = getEm().createQuery(requete);
        return query.getResultList();
    }
}
```

Arborescence des composants « managebeans »

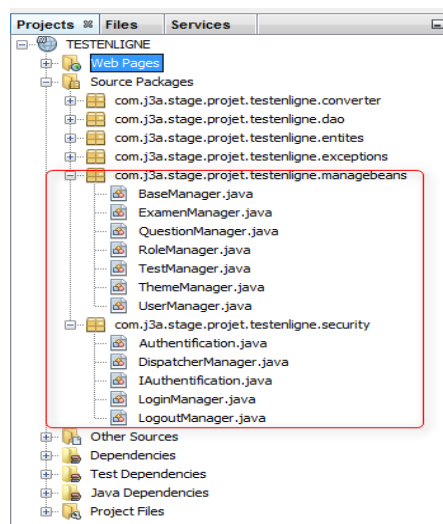


Figure 23 : Classes du contrôleur

Un extrait du manage Bean « ThemeManager » : il joue le rôle de colle entre la page JSF et le contrôleur FacesServlet.

The diagram illustrates the `ThemeManager` bean configuration. It features three red callout boxes with white text:

- injection dans le contenur IOC**: Points to the `@Named` annotation.
- Portée du Bean**: Points to the `@Scope(value = "session")` annotation.
- Injection des reference DAO**: Points to the `@Inject` annotations for `ThemeDao dao;` and `ExamenDao examenDao ;`.

```
@Named
@Scope(value = "session")
public class ThemeManager extends BaseManager implements Serializable {

    @Inject
    ThemeDao dao;

    @Inject
    ExamenDao examenDao ;

    public String motCle ;

    private Theme theme;
    private List<Theme> themes;

    private Examen examen ;
    private List<Examen> examens ;

    public ThemeDao getDao() {
        return dao;
    }

    public ExamenDao getExamenDao() {
        return examenDao;
    }

    @PostConstruct
    public void initDonnees() {
        theme = new Theme();
        themes = getDao().getAll();
    }
}
```

Arborescence des composants « Vue »

L'arborescence des composants « Vue » nous montre les différentes pages utilisées pour la conception des pages de l'application. Nous utilisons principalement des facelets pour la vue JSF.

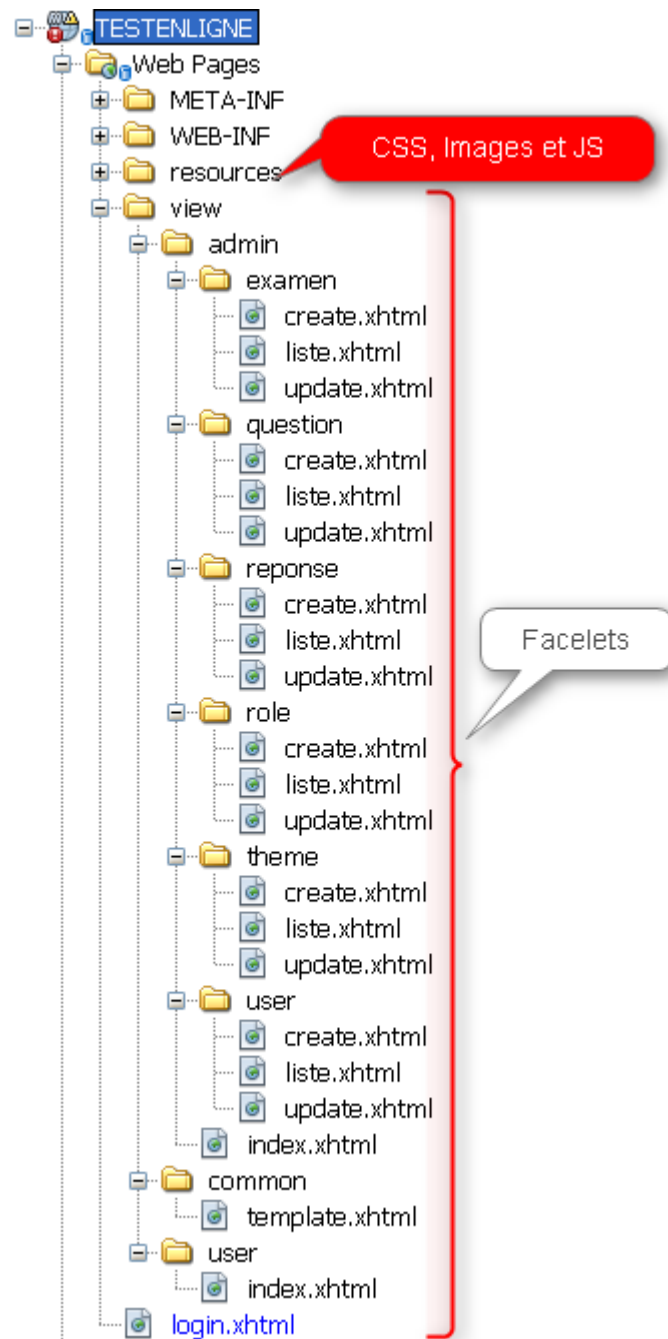


Figure 24 : Arborescence des composants Vue

Ci-dessous, un extrait du code la page « index.xhtml » de l'Admin

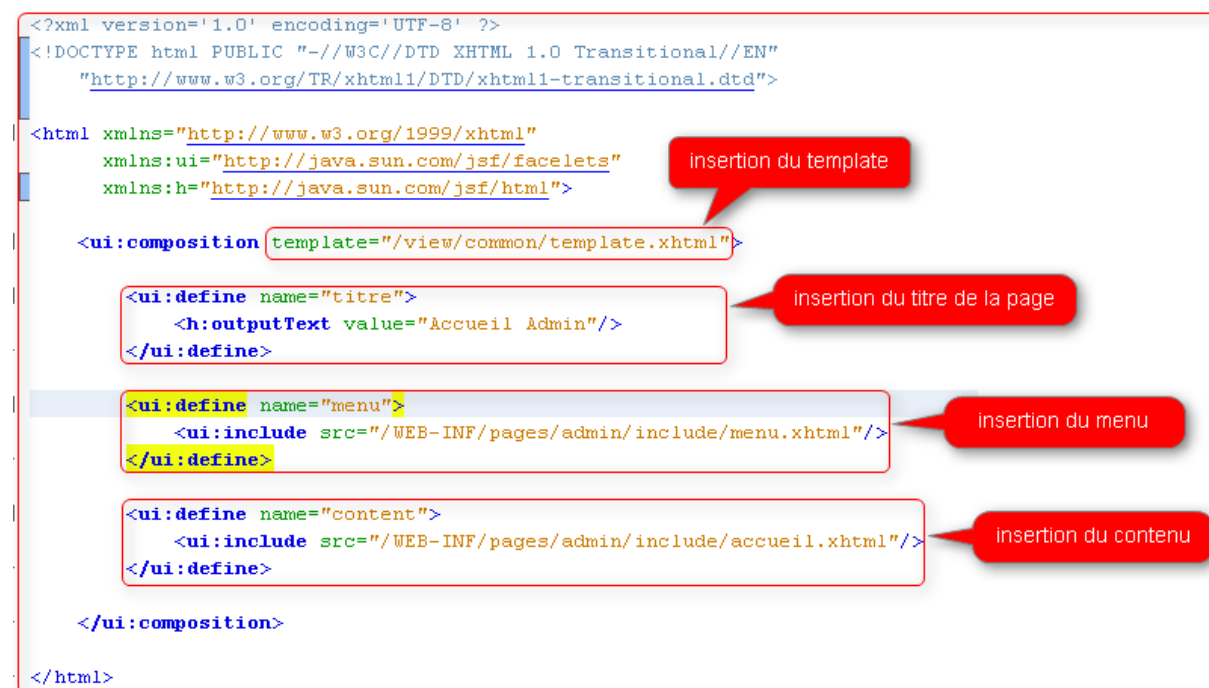


Figure 25 : Page accueil de l'administrateur

Intégration des composants avec Spring et Hibernate

L'intégration dans l'application de l'infrastructure Spring en plus de la solution Hibernate est de nature à simplifier le développement. Quelques-uns des apports à attendre de cette intégration est :

- Déléguer la gestion des sessions Hibernate à Spring
- Uniformiser la gestion des erreurs

Cette intégration se fait

1. dans un premier temps, par la configuration du fichier XML Spring nommé « beansConfig.xml »
2. dans un second temps, déclarer le fichier de configuration dans le web.xml

```

<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:tx="http://www.springframework.org/schema/tx"
       xmlns:context="http://www.springframework.org/schema/context"
       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd
                           http://www.springframework.org/schema/context http://www.springframework.org/schema/context/spring-context-3.2.xsd
                           http://www.springframework.org/schema/tx http://www.springframework.org/schema/tx/spring-tx-3.2.xsd">

    <context:annotation-config />
    <tx:annotation-driven transaction-manager="transactionManager" proxy-target-class="true"/>
    <context:component-scan base-package="com.j3a.stage.projet.testenligne" />

    <!-- Declaration de la source de données -->
    <bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource">
        <property name="driverClassName" value="com.mysql.jdbc.Driver" />
        <property name="url" value="jdbc:mysql://localhost:3306/stage" />
        <property name="username" value="gadi" />
        <property name="password" value="XXXXXXXXXX" />
        <property name="initialSize" value="2" />
        <property name="maxActive" value="5" />
    </bean>

    <!-- Declaration de la fabrique d'entity Manager -->
    <bean id="emf" class="org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean">
        <property name="dataSource" ref="dataSource" />
        <property name="packagesToScan" value="com.j3a.stage.projet.testenligne.entites" />
        <property name="jpaVendorAdapter">
            <bean class="org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter" />
        </property>
        <property name="jpaProperties">
            <props>
                <!--<prop key="hibernate.hbm2ddl.auto">create</prop>-->
                <prop key="hibernate.dialect">org.hibernate.dialect.MySQL5Dialect</prop>
            </props>
        </property>
    </bean>

    <!-- Declaration du gestion de transaction -->
    <bean id="transactionManager" class="org.springframework.orm.jpa.JpaTransactionManager">
        <property name="entityManagerFactory" ref="emf" />
    </bean>

    <!-- Declaration de la prise en charge de la gestion d'erreur par Spring -->
    <bean class="org.springframework.dao.annotation.PersistenceExceptionTranslationPostProcessor" />
</beans>

```

Figure 26 : fichier de configuration Spring

```

<context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>
        /WEB-INF/spring/beansConfig.xml
        /WEB-INF/spring/securityConfig.xml
    </param-value>
</context-param>
<listener>
    <listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>
</listener>
<listener>
    <listener-class>org.springframework.web.context.request.RequestContextListener</listener-class>
</listener>

```

Figure 27 : Déclaration de Spring dans le fichier web.xml

La configuration du projet a été complétée par la suite dans d'autres fichiers tel que le fichier « faces-config.xml » pour la gestion de la navigation entre les pages, et « securityConfig.xml » pour la configuration de la sécurité des ressources.

Extrait de codes java

Nous présentons ici quelques extraits de codes qui permettent la bonne compréhension du mécanisme de création d'un candidat.

Le code suivant est extrait de la classe « UserManager »

```
* Methode permettant de generer un nombre de Questions
*
* @param nombre le nombre de questions à generer
* @return La liste de Questions générées
*/
private List<Question> genererQuestion(Integer nombre) {
    List<Question> qs = new ArrayList<Question>();
    int taille = 0;

    while (taille < nombre) {
        int j = (int) Math.round((Math.random() * nombre));
        Question q = questions.get(j);
        if (!qs.contains(q)) {
            qs.add(q);
            taille++;
        }
    }

    return qs;
}
```

Figure 28 : Méthode permettant de générer un nombre de questions

Cette méthode privée sera utilisée lors de la création d'un candidat. Elle permettra de générer les questions du Test que passera le candidat. Autrement dit, lorsqu'un candidat est créé, son Sujet est automatiquement créé.


```

/**
 * Methode permettant de créer un candidat
 *
 * @return La page liste des resultats
 */
public String createUser() {

    //Role de d'utilisateur à créer
    role = getRoleDao().getByRole(user.getRole().getRole());

    //attribution du role à l'utilisateur
    user.setRole(role);

    //Rendre actif l'utilisateur afin de pouvoir se connecter au systeme
    user.setEnabled(true);

    //Recuperation de la liste de questions de l'examen concerné
    questions = getQuestionDao().getByExamen(examen.getNom());

    //recuperation du nbre de question saisi par l'admin
    int nbQuestion = questions.size();

    try {
        //On verifie que le non de question n'est pas superieur au nombre total de questions de l'examen
        if (nombreQuestion > nbQuestion) {
            addErrorMessage(null, "Erreur : Le nombre de question doit être inférieur à " + (nbQuestion - 1), null);
            return null;
        } else {

            //On genere les question du sujet à partir de la methode privée
            List<Question> qs = genererQuestion(nombreQuestion);

            //On attribut au sujet, les questions genereés
            sujet.setQuestions(qs);

            //on crée l'utilisateur
            getUserDao().create(user);

            //on recupere l'utilisateur avec tous ses attributs à partir de son email
            user = getUserDao().getUserByEmail(user.getEmail());

```

```

        //on envoi par mail, le Login et Mdp de l'utilisateur pour pouvoir se connecter
        envoiEmail(user);

        //On relie l'utilisateur au sujet
        sujet.setIdSujetUser(user.getLogin());

        //on créer le sujet
        getSujetDao().create(sujet);

        //on relie le sujet a son resultat
        resultat.setSujet(sujet);

        //on relie le candidat a son resultat
        resultat.setUser(user);

        //on attribut le score par default (0) au resultat
        resultat.setScore(score);

        //on crée le resultat
        getResultatDao().create(resultat);

        //on met à jour la liste des resultats
        resultats = getResultatDao().getAll();

        role = new Role();
        user = new User();
        sujet = new Sujet();
        resultat = new Resultat();

        return getPageListe();
    }

    } catch (Exception e) {
        logger.debug("Erreur creation User");
        addErrorMessage(null, "Erreur : " + e.getMessage(), null);
        return null;
    }
}

```

Le code suivant est extrait de la classe « TestManager »

```
/**
 * Methode permettant de terminer le test
 * @return La page de score
 */
public String terminerTest() {

    //on recupere le resultat lié au candidat
    resultat = getResultatDao().getResultatByUser(user).get(0);

    //desactive le candidat du syteme
    user.setEnabled(false);

    resultat.setUser(user);

    //On remplace le score par default pa le nouveau score
    resultat.setScore(score);

    //On met a jour le resultat du candidat
    resultat = getResultatDao().update(resultat);

    //On affiche le score
    afficherScore = true;
    commencer = false ;
    return null;
}
}
```

Cette méthode permet de terminer le test et met à jour le résultat du candidat.

CONCLUSION

Voici venue le terme de mon projet. Il m'a permis de mettre en œuvre les différents composants de ma formation, c'est-à-dire :

- L'analyse d'un problème à informatiser
- Sa conception et modélisation.
- Sa persistance en base de données
- La réalisation de son IHM.

Le stage dans sa globalité m'a permis de me rendre compte de la qualité du travail effectué en équipe dans le cadre du développement informatique de nos jours, de la capacité que doit avoir un développeur à réaliser sa veille technologique et enfin à mettre en pratique le langage Java/Java EE dans un cadre d'entreprises.

Tous mes remerciements à l'équipe des formateurs de l'ESIC Paris 12 qui, à travers une pédagogie bien adaptée, a su aller à l'essentiel durant cette formation, ainsi qu'à mon encadreur de stage qui a été un véritable aiguilleur dans l'apprentissage des bonnes pratiques que je dois avoir en entreprise.

ANNEXE

SCRIPT SQL DE LA BASE DE DONNEES « STAGE »

SGBDR : MySQL

```
--  
-- Base de données: `stage`  
--  
--  
--  
--  
-- Structure de la table `t_examens`  
--  
  
CREATE TABLE IF NOT EXISTS `t_examens` (  
  `EXA_ID` int(11) NOT NULL AUTO_INCREMENT,  
  `EXA_NOM` varchar(255) NOT NULL,  
  `THE_ID` int(11) DEFAULT NULL,  
  PRIMARY KEY (`EXA_ID`),  
  UNIQUE KEY `UK_ogvmqe8adho8yfqq01ww3o18wk` (`EXA_NOM`),  
  KEY `FK_lnrnk8go4bvvhru2fpan24bnm` (`THE_ID`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=2 ;  
  
--  
-- Contenu de la table `t_examens`  
--  
  
INSERT INTO `t_examens` (`EXA_ID`, `EXA_NOM`, `THE_ID`) VALUES  
(1, 'JAVA 01', 1);
```

```
--
-- Structure de la table `t_questions`
--

CREATE TABLE IF NOT EXISTS `t_questions` (
  `QUE_ID` int(11) NOT NULL AUTO_INCREMENT,
  `QUE_COEF` int(11) NOT NULL,
  `QUE_CONTENU` longtext NOT NULL,
  `QUE_IMG` longtext,
  `QUE_ISENABLE` tinyint(1) NOT NULL,
  `QUE_NIVEAU` int(11) DEFAULT NULL,
  `QUE_TYPE` varchar(255) DEFAULT NULL,
  `EXA_ID` int(11) DEFAULT NULL,
  PRIMARY KEY (`QUE_ID`),
  KEY `FK_xfp7uwxtldld1gxxw6y2qyefk` (`EXA_ID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=12 ;

--
-- Contenu de la table `t_questions`
--

INSERT INTO `t_questions` (`QUE_ID`, `QUE_COEF`, `QUE_CONTENU`, `QUE_IMG`, `QUE_ISENABLE`, `QUE_NIVEAU`,
  ➤ `QUE_TYPE`, `EXA_ID`) VALUES
(1, 1, 'La syntaxe de Java se rapproche du', 'pas_d_image.png', 1, 0, NULL, 1),
(2, 1, 'Quelle plate-forme Java est utilisée pour réaliser des application clientes ?',
  ➤ 'pas_d_image.png', 1, 0, NULL, 1),
(3, 1, 'Un objet représente :', 'pas_d_image.png', 1, 0, NULL, 1),
(4, 1, 'Une classe représente', 'pas_d_image.png', 1, 0, NULL, 1),
(5, 1, 'Java supporte-t-il l\'héritage multiple ?', 'pas_d_image.png', 1, 0, NULL, 1),
(6, 1, 'Une interface en Java représente :', 'pas_d_image.png', 1, 0, NULL, 1),
(7, 1, 'Le polymorphisme permet :', 'pas_d_image.png', 1, 0, NULL, 1),
(8, 1, 'Java supporte-t-il les types énumérés :', 'pas_d_image.png', 1, 0, NULL, 1),
(9, 1, 'L\'opérateur Java new', 'pas_d_image.png', 1, 0, NULL, 1),
(10, 1, 'Une méthode de classe', 'pas_d_image.png', 1, 0, NULL, 1),
(11, 1, 'Le modificateur d\'accès protected permet', 'pas_d_image.png', 1, 0, NULL, 1);
```

```
--
-- Structure de la table `t_reponses`
--

CREATE TABLE IF NOT EXISTS `t_reponses` (
  `REP_ID` int(11) NOT NULL AUTO_INCREMENT,
  `REP_CONTENU` varchar(255) NOT NULL,
  `REP_ISSELECTED` tinyint(1) DEFAULT NULL,
  `REP_ISTRUE` tinyint(1) NOT NULL,
  `QUE_ID` int(11) DEFAULT NULL,
  PRIMARY KEY (`REP_ID`),
  KEY `FK_3au0032i900qxgndlae88fdps` (`QUE_ID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=33 ;

--
-- Contenu de la table `t_reponses`
--

INSERT INTO `t_reponses` (`REP_ID`, `REP_CONTENU`, `REP_ISSELECTED`, `REP_ISTRUE`, `QUE_ID`) VALUES
(1, 'C', 0, 1, 1),
(2, 'Ada', 0, 0, 1),
(3, 'Pascal', 0, 0, 1),
(4, 'JME', 0, 0, 2),
(5, 'JSE', 0, 1, 2),
(6, 'JEE', 0, 0, 2),
(7, 'Un type de données', 0, 0, 3),
(8, 'Une entité créée à partir d’une classe', 0, 0, 3),
(9, 'Une concrétisation générée à partir d’un modèle', 0, 1, 3),
(10, 'Une abstraction d’un concept', 0, 1, 4),
(11, 'Un type de données évolué ( abstrait )', 0, 0, 4),
(12, 'Une instance particulière', 0, 0, 4),
```

```

(13, 'Oui', 0, 0, 5),
(14, 'Non', 0, 1, 5),
(15, 'Une classe ordinaire', 0, 0, 6),
(16, 'Une classe abstraite avec des données membres.', 0, 0, 6),
(18, 'Une classe abstraite sans données membres et dont les méthodes ne sont pas codées.', 0, 1, 6),
(19, 'D'éviter l'usage des interfaces', 0, 0, 7),
(20, 'De rendre abstraite une classe concrète', 0, 0, 7),
(21, 'D'unifier des appels de méthodes pour des objets issus de classes différentes', 0, 1, 7),
(22, 'Oui', 0, 1, 8),
(23, 'Non', 0, 0, 8),
(24, 'Ne s'utilise que pour les membres statiques', 0, 0, 9),
(25, 'Permet la création d'un objet cohérent grâce au concept de constructeur', 0, 1, 9),
(26, 'S'utilise pour créer une classe à partir d'une autre', 0, 0, 9),
(27, 'Est visible partout où la classe est accessible', 0, 0, 10),
(28, 'Est atteignable sans qu'une instance ne soit créée', 0, 1, 10),
(29, 'Est un traitement spécifique à une instance', 0, 0, 10),
(30, 'De donner un accès sans restriction à la propriété', 0, 0, 11),
(31, 'D'interdire l'accès de la propriété en dehors de la classe', 0, 0, 11),

```

```
--
```

```
-- Structure de la table `t_resultats`
```

```
--
```

```

CREATE TABLE IF NOT EXISTS `t_resultats` (
  `RES_ID` int(11) NOT NULL AUTO_INCREMENT,
  `RES_SCORE` int(11) DEFAULT NULL,
  `RES_TEMPS` int(11) DEFAULT NULL,
  `SUJ_ID` int(11) DEFAULT NULL,
  `USE_ID` int(11) NOT NULL,
  PRIMARY KEY (`RES_ID`),
  KEY `FK_n17aqmpg6g10btqc2qsmhcw26` (`SUJ_ID`),
  KEY `FK_ebu1av3yjjonnj11ea9p5mcud` (`USE_ID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=3 ;

```

```
--
```

```
-- Contenu de la table `t_resultats`
```

```
--
```

```

INSERT INTO `t_resultats` (`RES_ID`, `RES_SCORE`, `RES_TEMPS`, `SUJ_ID`, `USE_ID`) VALUES
(1, 0, 0, 1, 1),
(2, 4, 0, 2, 2);

```



```
--  
-- Structure de la table `t_roles`  
--  
  
CREATE TABLE IF NOT EXISTS `t_roles` (  
  `ROL_ID` int(11) NOT NULL AUTO_INCREMENT,  
  `ROL_ROLE` varchar(255) NOT NULL,  
  PRIMARY KEY (`ROL_ID`),  
  UNIQUE KEY `UK_20l08jsh0rtxwdyogvqdnets` (`ROL_ROLE`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=3 ;
```

```
--  
-- Contenu de la table `t_roles`  
--
```

```
INSERT INTO `t_roles` (`ROL_ID`, `ROL_ROLE`) VALUES  
(1, 'ROLE_ADMIN'),  
(2, 'ROLE_USER');
```

```
-- Structure de la table `t_sujets`
```

```
--
```

```
CREATE TABLE IF NOT EXISTS `t_sujets` (  
  `SUJ_ID` int(11) NOT NULL AUTO_INCREMENT,  
  `SUJ_DATE` date DEFAULT NULL,  
  `SUJ_DUREE` int(11) DEFAULT NULL,  
  `SUJ_NOM` varchar(255) NOT NULL,  
  PRIMARY KEY (`SUJ_ID`),  
  UNIQUE KEY `UK_sdwnxdnu1egj1bsvslymjc3gy` (`SUJ_NOM`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=3 ;
```

```
--
```

```
-- Contenu de la table `t_sujets`
```

```
--
```

```
INSERT INTO `t_sujets` (`SUJ_ID`, `SUJ_DATE`, `SUJ_DUREE`, `SUJ_NOM`) VALUES  
(1, NULL, 20, 'joachim'),  
(2, NULL, 20, 'arsene');
```

```
--
```

```
-- Structure de la table `t_sujets_t_questions`
```

```
--
```

```
CREATE TABLE IF NOT EXISTS `t_sujets_t_questions` (  
  `sujets_SUJ_ID` int(11) NOT NULL,  
  `questions_QUE_ID` int(11) NOT NULL,  
  KEY `FK_2k067if4usroiijkbxpt6330a` (`questions_QUE_ID`),  
  KEY `FK_5wggilf37dlotgwrm0bt5ra` (`sujets_SUJ_ID`)  
) ENGINE=InnoDB DEFAULT CHARSET=latin1;
```

```
--  
-- Contenu de la table `t_sujets_t_questions`  
--  
  
INSERT INTO `t_sujets_t_questions` (`sujets_SUJ_ID`, `questions_QUE_ID`) VALUES  
(1, 5),  
(1, 6),  
(1, 7),  
(1, 3),  
(1, 11),  
(1, 10),  
(1, 9),  
(1, 1),  
(1, 8),  
(1, 2),  
(2, 5),  
(2, 4),  
(2, 8),  
(2, 2),  
(2, 6),  
(2, 9),  
(2, 7),  
(2, 3);
```

```
--
-- Structure de la table `t_themes`
--

CREATE TABLE IF NOT EXISTS `t_themes` (
  `THE_ID` int(11) NOT NULL AUTO_INCREMENT,
  `THE_NOM` varchar(255) NOT NULL,
  PRIMARY KEY (`THE_ID`),
  UNIQUE KEY `UK_q0rk2a40y1vy0746umrv7pbp1` (`THE_NOM`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=2 ;
```

```
--
-- Contenu de la table `t_themes`
--

INSERT INTO `t_themes` (`THE_ID`, `THE_NOM`) VALUES
(1, 'JAVA');
```

```
-- Structure de la table `t_users`
--

CREATE TABLE IF NOT EXISTS `t_users` (
  `USE_ID` int(11) NOT NULL AUTO_INCREMENT,
  `USE_DDN` date NOT NULL,
  `USE_EMAIL` varchar(255) NOT NULL,
  `ENABLED` tinyint(1) NOT NULL,
  `USE_LOGIN` varchar(255) NOT NULL,
  `USE_MDP` varchar(255) NOT NULL,
  `USE_NOM` varchar(255) NOT NULL,
  `USE_PRENOM` varchar(255) NOT NULL,
  `ROL_ID` int(11) NOT NULL,
  PRIMARY KEY (`USE_ID`),
  UNIQUE KEY `UK_jf9omsaddcgash9r2baa2c19b` (`USE_EMAIL`),
  UNIQUE KEY `UK_jqc28j4bpya0syba6jdblabos` (`USE_LOGIN`),
  KEY `FK_4wxpm3cem4g2cye5u702s6r13` (`ROL_ID`)
) ENGINE=InnoDB DEFAULT CHARSET=latin1 AUTO_INCREMENT=3 ;
```

```
--

-- Contenu de la table `t_users`

--

INSERT INTO `t_users` (`USE_ID`, `USE_DDN`, `USE_EMAIL`, `ENABLED`, `USE_LOGIN`, `USE_MDP`, `USE_NOM`,
➤ `USE_PRENOM`, `ROL_ID`) VALUES

(1, '1969-03-20', 'goibazaro@yahoo.fr', 1, 'joachim', '123456789', 'Zadi', 'Joachim', 1),
(2, '1971-04-01', 'joachim.zadi@gmail.com', 1, 'arsene', '987654321', 'Yao', 'Emmanuel', 2);

--

-- Contraintes pour la table `t_examens`

--

ALTER TABLE `t_examens`

  ADD CONSTRAINT `FK_lnrnk8go4bvvhru2fpan24bnm` FOREIGN KEY (`THE_ID`) REFERENCES `t_themes` (`THE_ID`);

--

-- Contraintes pour la table `t_questions`

--

ALTER TABLE `t_questions`

  ADD CONSTRAINT `FK_xfp7uwxtldld1gxxw6y2qyefk` FOREIGN KEY (`EXA_ID`) REFERENCES `t_examens` (`EXA_ID`);

--

-- Contraintes pour la table `t_reponses`

--

ALTER TABLE `t_reponses`

  ADD CONSTRAINT `FK_3au0032i900qxgndlae88fdps` FOREIGN KEY (`QUE_ID`) REFERENCES `t_questions`
➤ (`QUE_ID`);
```

```

-- Contraintes pour la table `t_resultats`
--

ALTER TABLE `t_resultats`

  ADD CONSTRAINT `FK_ebu1av3yjjonnj11ea9p5mcud` FOREIGN KEY (`USE_ID`) REFERENCES `t_users` (`USE_ID`),

  ADD CONSTRAINT `FK_n17aqmpg6g10btqc2qsmhwcw26` FOREIGN KEY (`SUJ_ID`) REFERENCES `t_sujets` (`SUJ_ID`);

--

-- Contraintes pour la table `t_sujets_t_questions`
--

ALTER TABLE `t_sujets_t_questions`

  ADD CONSTRAINT `FK_2k067if4usroiijkbxpt6330a` FOREIGN KEY (`questions_QUE_ID`) REFERENCES
  ➤ `t_questions` (`QUE_ID`),

  ADD CONSTRAINT `FK_5wggilf37dlotgwvrm0bt5ra` FOREIGN KEY (`sujets_SUJ_ID`) REFERENCES `t_sujets`
  ➤ (`SUJ_ID`);

--

-- Contraintes pour la table `t_users`
--

ALTER TABLE `t_users`

  ADD CONSTRAINT `FK_4wxpm3cem4g2cye5u702s6r13` FOREIGN KEY (`ROL_ID`) REFERENCES `t_roles` (`ROL_ID`);

```