

Framework de modélisation d'un projet en langage UML

Par Stéphanie YOUNA, Armel CHETAH et Salomon MAYENGUE
- (Centre de Développements Projets forfaits, KOOSSERY
TECHNOLOGY) - Baké Jc. BAKENEGHE (Software Coordinator)

Date de publication : 19 octobre 2009

Cet article présente un cadre de travail pour la modélisation d'un projet en langage UML. Il s'adresse principalement aux concepteurs UML, aux développeurs emmenés à faire la modélisation UML ainsi qu'aux architectes et chefs de projets.
Vos réactions par l'intermédiaire de ce fil :

I - Introduction.....	4
II - Les principales parties d'un projet UML.....	4
III - Modèle des cas d'utilisation : Use Case Model.....	4
III-1 - Description des éléments de modélisation du Use case Model.....	5
III-1-1 - Acteur (Actor).....	5
III-1-1-1 - Multiplicité.....	5
III-1-1-2 - Convention de Nommage.....	5
III-1-1-3 - Utilisation d'un Actor.....	6
III-1-1-4 - Documentation à fournir pour chaque Actor.....	6
III-1-2 - Use Case.....	6
III-1-2-1 - Multiplicité.....	6
III-1-2-2 - Convention de Nommage.....	6
III-1-2-3 - Utilisation d'un Use Case.....	6
III-1-2-4 - Documentation à fournir pour chaque Use Case.....	6
III-1-3 - Diagramme de Cas d'Utilisation.....	7
III-1-3-1 - Multiplicité.....	7
III-1-3-2 - Convention de Nommage.....	7
III-1-3-3 - Utilisation d'un diagramme de Cas d'utilisation.....	8
III-1-4 - Diagramme de Séquence de niveau système.....	8
III-1-4-1 - Multiplicité.....	8
III-1-4-2 - Conventions de Nommage.....	8
III-1-4-3 - Utilisation d'un Diagramme de séquence.....	9
III-1-4-4 - Documentation à fournir pour un diagramme de séquence.....	9
III-2 - Contenu du Use Case Model.....	9
III-2-1 - Package Actor.....	10
III-2-2 - Package Context.....	10
III-2-3 - Package Uses Cases.....	10
IV - Modèle d'analyse.....	10
IV-1 - Description des éléments de modélisation du modèle d'analyse.....	11
IV-1-1 - Les Classes.....	11
IV-1-1-1 - Multiplicité.....	11
IV-1-1-2 - Utilisation d'une classe.....	11
IV-1-1-3 - Contenu.....	11
IV-1-1-4 - Documentation d'une classe.....	11
IV-1-2 - Composant.....	11
IV-1-2-1 - Multiplicité.....	12
IV-1-2-2 - Convention de nommage.....	12
IV-1-2-3 - Contenu.....	12
IV-1-3 - Diagramme de Classe.....	12
IV-1-3-1 - Multiplicité.....	12
IV-1-3-2 - Conventions de nommage.....	12
IV-1-3-3 - Utilisation.....	13
IV-1-3-4 - Documentation d'un diagramme de classe.....	13
IV-1-4 - Diagramme de Séquence.....	13
IV-1-4-1 - Multiplicité.....	13
IV-1-4-2 - Conventions de nommage.....	14
IV-1-4-3 - Utilisation d'un diagramme de séquence.....	14
IV-2 - Contenu du modèle d'analyse.....	15
IV-2-1 - Package des composants métiers.....	15
IV-2-1-1 - Multiplicité.....	15
IV-2-1-2 - Nommage.....	15
IV-2-1-3 - Documentation.....	15
IV-2-2 - Package des composants services.....	16
IV-2-2-1 - Multiplicité.....	16
IV-2-2-2 - Nommage.....	16
IV-2-3 - Package des médiations.....	16
IV-2-3-1 - Multiplicité.....	16
IV-2-3-2 - Nommage.....	17

IV-2-3-3 - Contenu.....	17
IV-2-4 - Package Use Case Realization.....	17
IV-2-4-1 - Package des Regroupements de réalisation des cas d'utilisation.....	17
IV-2-4-1-1 - Multiplicité.....	17
IV-2-4-1-2 - Convention de Nommage.....	17
IV-2-4-1-3 - Contenu.....	17
IV-2-4-2 - Package de réalisation de cas d'utilisation.....	18
IV-2-4-2-1 - Multiplicité.....	18
IV-2-4-2-2 - Convention de Nommage.....	18
IV-2-4-2-3 - Contenu.....	18
V - Cas pratique : Modélisation UML d'un projet simple.....	18
V-1 - Spécifications fonctionnelles du projet.....	18
V-2 - Modélisation de la partie serveur : GestClient.Server.....	19
V-2-1 - Use Case Model.....	19
V-2-1-1 - Package Actor.....	19
V-2-1-2 - Package Context.....	19
V-2-1-3 - Package Use Cases.....	19
V-2-2 - Modèle d'analyse.....	20
V-2-2-1 - Data Model.....	20
V-2-2-2 - Package des composants métiers.....	20
V-2-2-3 - Package des composants services.....	21
V-2-2-4 - Package des médiations.....	21
V-2-2-5 - Package des réalisations des cas d'utilisation.....	22
V-2-3 - Téléchargement du projet UML de la partie serveur de l'exemple.....	22
V-3 - Modélisation du Front End : GestClient.FrontEnd.....	23
V-3-1 - User Interface Model.....	23
V-3-2 - Use Case Model.....	23
V-3-2-1 - Package Actor.....	23
V-3-2-2 - Package Context.....	23
V-3-2-3 - Package Use Cases.....	23
V-3-2-3-1 - <<UC Package>>-Connexion.....	23
V-3-3 - Modèle d'analyse.....	25
V-3-3-1 - Data Model.....	25
V-3-3-2 - Package des composants métiers.....	25
V-3-3-3 - Package des composants services.....	25
V-3-3-4 - Package des médiations.....	27
V-3-3-5 - Package des réalisations des cas d'utilisation.....	27
V-3-3-5-1 - <<UCR Package>>-Connexion.....	27
V-3-4 - Téléchargement du projet UML de la partie front-end de l'exemple.....	28

I - Introduction

Dans le cycle de développement d'un projet logiciel, la modélisation du projet en langage UML revêt une grande importance. En effet c'est notamment à travers la modélisation des cas d'utilisation du projet UML que les équipes fonctionnelles et les équipes techniques s'accordent sur la bonne compréhension de l'expression des besoins décrits dans une spécification fonctionnelle. Le projet UML permet de décliner la conception technique du projet, permettant ainsi une meilleure implémentation par les développeurs. Le projet UML sert aussi d'outil de communication efficace entre les différents intervenants techniques du projet.

Il convient aussi de souligner qu'afin de gagner en productivité, de plus en plus d'entreprises basent leurs développements sur des moteurs MDA (Model Driven Architecture) qui prennent en entrée un modèle UML et génèrent du code source en sortie. Telle est l'option prise par Koossery Technology avec la mise en oeuvre de **OptimaDev**, moteur MDA qui fera l'objet d'une prochaine publication.

II - Les principales parties d'un projet UML

Un projet UML est structuré en plusieurs parties dont les principales sont :

- la partie de modélisation des cas d'utilisation : Use Case Model.
- la partie d'analyse qui est la réponse technique au Use Case Model : Analysis Model.
- la partie de modélisation des interfaces utilisateurs : User Interface Model. Cette partie n'intervient que dans le cas d'un projet front-end (winapp ou webapp).
- la partie de modélisation des workflow : Business Process Model.
- le modèle de déploiement : Deployment model. Le modèle de déploiement sert à représenter les éléments matériels (ordinateurs, périphériques, réseaux, systèmes de stockage...) et la manière dont les composants du système sont répartis sur ces éléments matériels et interagissent avec eux.
- le modèle des tests : Test Model.
- le modèle de maintenance : Maintenance Model.
- etc.

Le présent article porte uniquement le modèle des cases d'utilisation (Use Case Model) et le modèle d'analyse (Analysis Model).

III - Modèle des cas d'utilisation : Use Case Model

Un cas d'utilisation représente généralement un objectif complet que remplit le système de manière à répondre au besoin d'un acteur appelé initiateur pour ce cas d'utilisation.

Les cas d'utilisation d'un projet sont modélisés dans la partie Use Case Model. Cette dernière est constituée d'un ensemble d'éléments dit de modélisation et de diagrammes parmi lesquels :

- les acteurs,
- les cas d'utilisations,
- les diagrammes de séquence et de cas d'utilisation.

Nous traitons chacun des éléments ci-dessus dans la suite.

III-1 - Description des éléments de modélisation du Use case Model

III-1-1 - Acteur (Actor)

Un acteur représente un rôle d'une entité présente dans le contexte du système considéré vis-à-vis de ce système. L'entité doit être distincte du système lui-même. L'entité doit être extérieure au système considéré. En fonction de son rôle l'entité interagit avec le système.

III-1-1-1 - Multiplicité

1..n : on peut avoir un ou plusieurs acteurs.

III-1-1-2 - Convention de Nommage

Pour les acteurs représentant des rôles d'utilisateurs humains : **<<Actor>> <nom du rôle d'utilisateur>**.

Exemple : pour un projet front-end on pourrait avoir un acteur humain (un utilisateur de rôle Agent) et on note **<<Actor>>Agent**.

Pour les acteurs représentant des rôles de systèmes externes : **<<External System>> <nom du rôle de système>**.

Exemple : pour un projet serveur, on pourrait avoir un acteur externe (le front-end) et on note **<<ExternalSystem>>Front-End**.

Le nom du rôle doit être un groupe nominal.

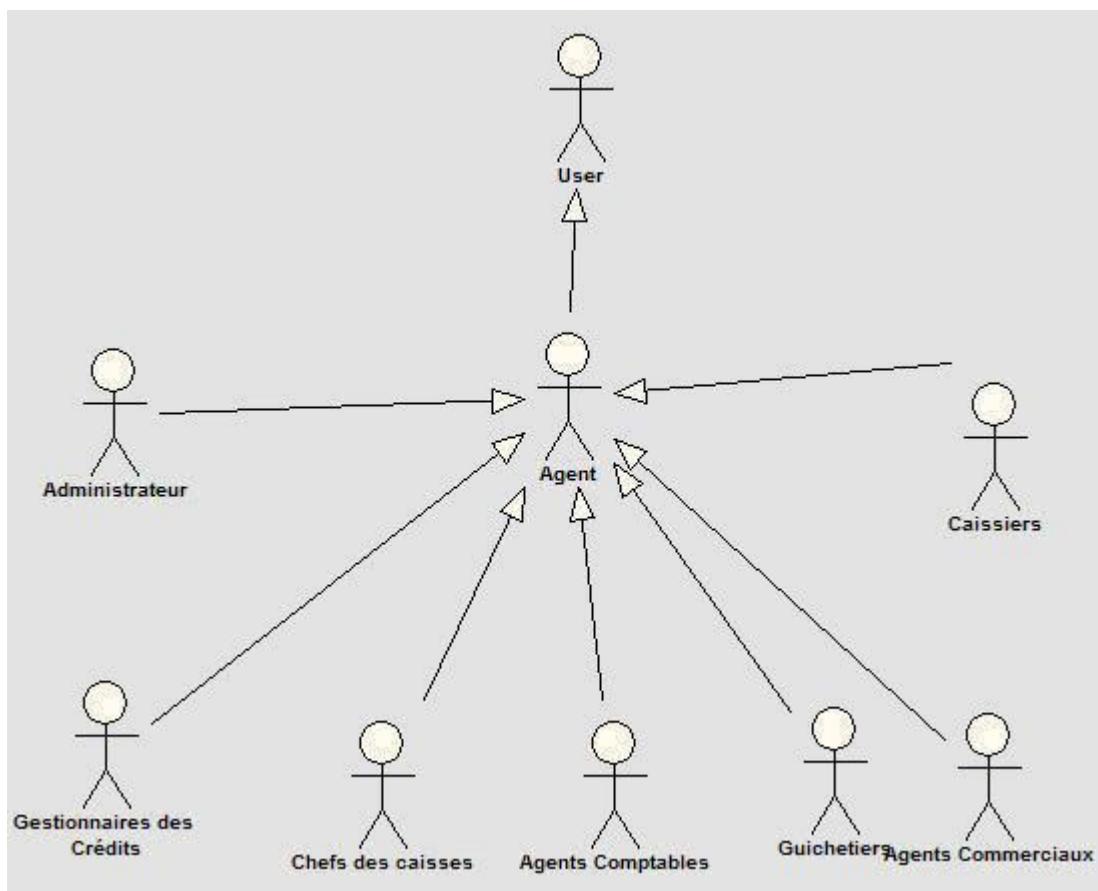


Figure 1 : Exemple d'acteurs humain

III-1-1-3 - Utilisation d'un Actor

Un acteur peut apparaître dans tout ou partie des diagrammes de cas d'utilisation du modèle.

Un acteur peut apparaître sous la forme d'une de ses instances dans tout ou partie des diagrammes de séquence du modèle.

III-1-1-4 - Documentation à fournir pour chaque Actor

La documentation à fournir pour chaque acteur doit contenir la définition complète de l'acteur. Par exemple pour un projet front-end un acteur humain de rôle Caissier pourra avoir comme documentation : *<<agent ayant pour rôle d'effectuer les opérations de dépôt et de retrait d'argent demandées par les usagers>>*.

III-1-2 - Use Case

Un cas d'utilisation peut impliquer dans le cadre de son déroulement un ou plusieurs autres acteurs appelés participants pour ce cas d'utilisation.

L'implication d'un acteur dans un cas d'utilisation doit être matérialisée par une association entre l'acteur et le cas d'utilisation. Optionnellement, pour l'acteur initiateur il est possible de faire porter le rôle "Initiateur" à l'extrémité de l'association liée à l'acteur.

III-1-2-1 - Multiplicité

1...n : on peut avoir un ou plusieurs Use Case.

III-1-2-2 - Convention de Nommage

La convention de nommage est :

<<Use Case>>UC-<Code du Package>-<Numéro d'ordre du UC>-<Nom du cas d'utilisation>

- le nom du cas d'utilisation doit être un groupe verbal commençant par un verbe à l'infinitif;
- le code du package est une chaîne de caractère qui représente l'ensemble des UC (UC pour Use Case) qui sont liés à une fonctionnalité;
- numéro d'ordre de l'UC de format 0xy, où x et y sont des chiffres allant de 0 à 9;


 **<<Use Case>>UC-CLTPHYS-001-Créer un client physique**

Figure 2 : Exemple de nommage d'un Use Case

III-1-2-3 - Utilisation d'un Use Case

Un cas d'utilisation doit apparaître avec tous les acteurs impliqués dans son déroulement dans le diagramme de cas d'utilisation du package de regroupement des cas d'utilisation auquel il appartient.

III-1-2-4 - Documentation à fournir pour chaque Use Case

La documentation d'un Use Case doit contenir :

- Événement déclencheur :

Description de l'événement matérialisant la présence d'un besoin de l'acteur initiateur, besoin qui va alors nécessiter la réalisation du cas d'utilisation.

Conventions : La description utilise la forme standardisée suivante : 'Le cas d'utilisation s'initie lorsque <l'acteur initiateur> <le groupe verbal décrivant l'action de l'acteur considérée comme l'événement déclencheur>.

- Pré-conditions :

Description des conditions pertinentes et significatives (non triviales) devant être vérifiées à l'apparition de l'événement déclencheur pour que le cas d'utilisation puisse être déclenché.

- Résumé :

Description synthétique du déroulement du cas d'utilisation dans son cas nominal, déroulement présentant les grandes étapes de cette interaction.

- Post-conditions :

Description des conditions globales et significatives devant être vérifiées lorsque le cas d'utilisation se termine quelle que soit la manière dont le cas d'utilisation s'est réalisé (scénario nominal, alternatif ou exceptionnel).

- Contraintes non fonctionnelles :

Descriptions des contraintes pesant sur le cas d'utilisation. Ces contraintes peuvent être de nature technique (exemple : contraintes transactionnelles), volumétrique, etc.

- Règles de gestion :

Descriptions des règles de gestion et autres exigences applicables au cas d'utilisation ou références vers leurs spécifications dans des documents extérieurs au modèle.

Exemple de documentation d'un cas d'utilisation	
Titre : Enregistrer un client physique	
Evènement Déclencheur: Le cas d'utilisation s'initie lorsque Front-End (FE) demande d'enregistrer un client physique.	
Post-condition(s): N/A	
Résumé: cas nominal	
1. Le FE invoque un service du serveur pour enregistrer un client physique	
2. Le FE communique au service ci-dessus les informations relatives à ce client physique tel que : son nom, prénom, la liste des documents d'identifications, les revenus de ce dernier, etc.	
3. Le serveur procède à l'enregistrement du client physique et renvoie le matricule généré	
Contraintes non fonctionnelles	
Type de contrainte	Descriptif
Temps de réponse	Les différentes requêtes doivent prendre moins de 1 seconde
Fréquence Volumétrie	On peut compter une centaine d'enregistrement de ce type en une heure Le nombre de clients physiques est grand (l'ordre peut être 1000 par jour)
Disponibilité	Cette fonction doit être disponible à tout instant, lorsque le système est disponible.
Concurrence	Plusieurs clients peuvent être enregistrés en simultané en même temps
Intégrité	Non applicable.
Confidentialité	L'enregistrement d'un client physique est uniquement possible pour un agent ayant un rôle de chargé de clientèle. Le service du serveur doit donc être sécurisé en conséquence.

Figure3 : Exemple de documentation d'un cas d'utilisation

III-1-3 - Diagramme de Cas d'Utilisation

Un diagramme de cas d'utilisation réunit sur un même diagramme un ensemble de cas d'utilisation, un ensemble d'acteurs et les relations qui unissent ces acteurs aux cas d'utilisation lors du déroulement des fonctionnalités prises en charge par les cas d'utilisation.

III-1-3-1 - Multiplicité

1..n : on peut avoir un ou plusieurs cas d'utilisation.

III-1-3-2 - Convention de Nommage

Pas de convention de nommage particulière.

III-1-3-3 - Utilisation d'un diagramme de Cas d'utilisation

Un diagramme de cas d'utilisation doit être fourni pour chaque package de regroupement de cas d'utilisation.

Un diagramme de cas d'utilisation présente :

- l'ensemble des cas d'utilisation contenus dans le package de regroupement considéré avec leurs acteurs.
- les dépendances (include, extend, généralisation) qui unissent les cas d'utilisation entre eux.

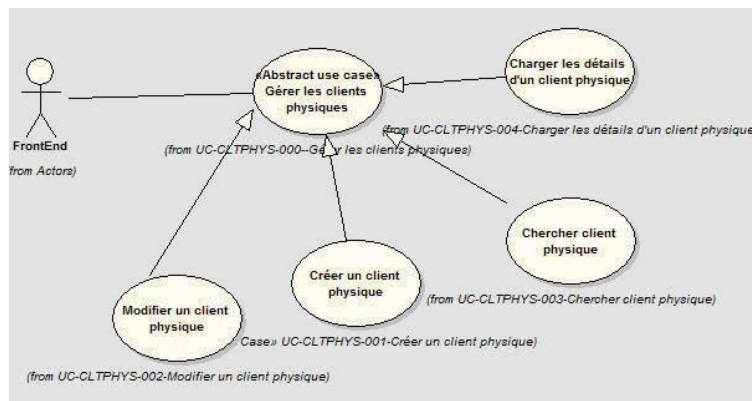


Figure 4 : Exemple de diagramme de cas d'utilisation

III-1-4 - Diagramme de Séquence de niveau système

Un diagramme de séquence met en jeu un ensemble d'objets (instances de classes ou acteurs) s'échangeant des messages représentant les demandes de services et les réponses à ces demandes.

Dans le Use Case Model, les diagrammes de séquence spécifiés sont exprimés "au niveau du système". Seuls sont représentés les acteurs extérieurs et un objet unique symbolisant le système dans sa globalité. Le diagramme de séquence système ne présente donc que les interactions entre le système et l'extérieur, les interactions internes au système n'étant révélées que plus tard dans des réalisations de cas d'utilisation.

III-1-4-1 - Multiplicité

1..n : on peut avoir un ou plusieurs diagramme de séquence.

III-1-4-2 - Conventions de Nommage

SC<numéro d'ordre du scénario>-<type du scénario>-<nom du scénario>

- Les types possibles de scénarios sont : Nominal, Alternatif et Exception.
- Le numéro d'ordre doit être exprimé sur deux positions. Les numéros d'ordre de format x0 doivent être réservés aux scénarios nominaux. Les scénarios alternatifs et exceptionnels doivent être numérotés selon le format xy où x correspond au x du scénario nominal à partir duquel ils sont exprimés et où y est un chiffre séquentiel différent de 0.
- Le nom du scénario doit être un groupe verbal commençant par un verbe à l'infinitif.
- Le nom du scénario est optionnel pour un scénario de type Nominal s'il s'agit de l'unique scénario de ce type pour le cas d'utilisation considéré.

III-1-4-3 - Utilisation d'un Diagramme de séquence

Chaque diagramme de séquence de niveau système doit représenter un scénario de résolution d'un cas d'utilisation. Chacun de ces diagrammes doit être créé dans le package contenant le cas d'utilisation pour lequel il représente un scénario de résolution.

Pour chaque cas d'utilisation :

- Un (1) de ces diagrammes présente le scénario de type Nominal c'est-à-dire représentant des interactions normales permettant d'atteindre la satisfaction de l'objectif de l'acteur initiateur.
- Plusieurs (0..n) de ces diagrammes présentent les scénarios de type Alternatif c'est-à-dire représentant des interactions inhabituelles suite à des choix, circonstances et conditions particulières au cours de la résolution du cas d'utilisation, mais permettant d'atteindre tout de même l'objectif initial de l'acteur initiateur.
- Plusieurs (0..n) de ces diagrammes présentent les scénarios de type Exception c'est-à-dire représentant des interactions anormales suite à des erreurs ou des conditions particulières au cours de la résolution du cas d'utilisation ne permettant pas d'atteindre l'objectif initial de l'acteur initiateur.

Chacun de ces diagrammes présente :

- Les acteurs et le système impliqués dans le déroulement du scénario considéré sous la forme d'instances de ces acteurs et de l'objet représentant artificiellement le système.
- L'enchaînement des interactions, des actions et réactions entre les acteurs et le système sous la forme de messages ordonnancés temporellement entre les instances des acteurs et du système.

III-1-4-4 - Documentation à fournir pour un diagramme de séquence

Elle doit contenir :

- Spécification textuelle :

Description de l'interaction entre les acteurs et le système pour le scénario considéré. Cette description doit donner étape par étape l'enchaînement des demandes et des réponses entre acteurs et système en précisant plus finement les éléments d'informations échangés.

Des références aux exigences et règles métiers justifiant les étapes de déroulement du scénario doivent être intégrées dans cette spécification sous la forme suivante : [<référence de l'exigence>].

- Post-conditions :

Description des conditions spécifiques et significatives devant être vérifiées lorsque le scénario se termine. Ces post-conditions sont optionnelles et complémentaires aux post-conditions globales spécifiées dans le cas d'utilisation.

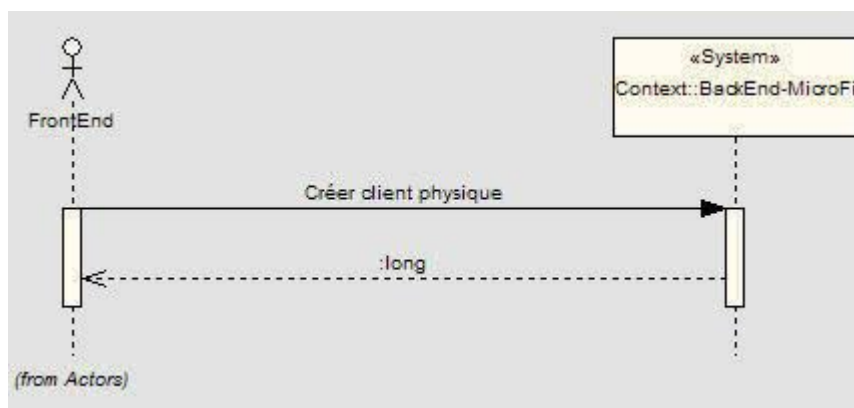


Figure 5 : SC10-Nominal-Créer un client physique.

III-2 - Contenu du Use Case Model

Le Use case Model est composé de 3 principaux packages :

- le package des acteurs,

- le package de contexte,
- et le package des cas d'utilisation.

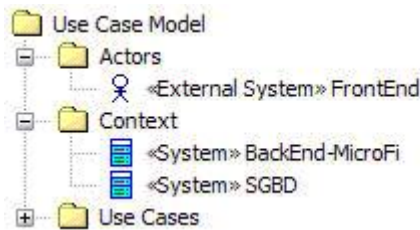


Figure 6 : Exemple d'organisation d'un Use Case Model

III-2-1 - Package Actor

C'est dans ce package qu'on range tous les acteurs du projet.

On y trouve aussi bien les acteurs humains que les systèmes extérieurs. On peut éventuellement faire figurer les interfaces de bibliothèques extérieures utilisées par le système modélisé. Ce regroupement dans un package unique permet une gestion centralisée de l'ensemble des acteurs du système.

III-2-2 - Package Context

On y signale toutes les entités systèmes du projet. On entend ici par entité un bloc. Par exemple pour un projet donné, les entités pourraient être : le front-end, le serveur, la base de donnée, l'ERP, etc. (cf figure 6).

III-2-3 - Package Uses Cases

C'est dans ce package qu'on range tous les uses cases du projet. Pour une meilleure lecture les use cases d'un projet sont organisés par module fonctionnel : chaque bloc fonctionnel constitue alors un package de Uses Cases.

Par exemple pour un logiciel de gestion banque de détail on pourrait avoir les modules fonctionnels suivants :

- Gestion des clients physiques.
- Gestion des comptes d'épargne.

Ce qui donnerait alors par exemple pour le module de gestion des clients physiques le regroupement <<UC Package>>CLTPHYS- Gestion Clients Physiques comme montre la figure 7 extraite d'un projet UML :

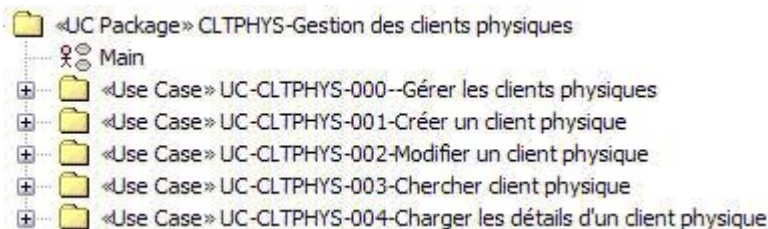


Figure 7 : Package des Use Case de Gestion des clients physiques

IV - Modèle d'analyse

Cette partie est la réponse technique au Use Case Model.

Le modèle d'analyse est construit à partir d'un ensemble d'éléments de modélisation (les classes, les composants, les diagrammes de classes, les diagrammes de séquence) et structuré en plusieurs parties.

IV-1 - Description des éléments de modélisation du modèle d'analyse

IV-1-1 - Les Classes

IV-1-1-1 - Multiplicité

1..n : on peut avoir une ou plusieurs classes.

IV-1-1-2 - Utilisation d'une classe

A l'instar du Use Case Model, en analyse les classes modélisées ne représentent toujours pas des objets logiciels implémentables mais résultent de la conceptualisation objet du système attendu.

Certaines de ces classes, les entités, sont directement issues de l'expression des besoins. Leur spécification se poursuit et se précise en analyse.

Les autres classes, services et médiations, sont identifiées au cours de l'analyse essentiellement en fonction des nécessités liées à la modélisation des scénarios.

IV-1-1-3 - Contenu

Une classe peut contenir :

- 1..n attributs.
- 1..n opérations.

IV-1-1-4 - Documentation d'une classe

Elle doit contenir la définition de la classe et l'ensemble des commentaires et contraintes pertinents portant sur la classe dans son ensemble.

Il est également nécessaire de valoriser certains métas-attributs comme la volumétrie. La volumétrie indique le nombre d'instances possibles de la classe dans le système (il s'agit généralement plus d'un ordre de grandeur que d'une valeur précise).

IV-1-2 - Composant

On distingue 2 types de composants : les composants métiers et les composants services.

Les composants métiers modélisent la partie la plus stable du système d'information c'est-à-dire celle susceptible d'évoluer le moins dans le temps. Elle correspond en général aux entités stables du Système d'Information.

A la différence des composants métiers, les composants services modélisent la partie la plus instable du système d'information c'est-à-dire celle susceptible d'évoluer le plus dans le temps. Elle correspond en général aux processus métiers dont le rôle est d'orchestrer les composants métiers.

Les services rendus par un composant service doivent être à terme exprimés à travers des interfaces présentées par le composant mais réalisées par les entités du composant de manière à renforcer l'encapsulation des objets et donc le faible couplage des composants entre eux.

Exemple : pour un projet Serveur un composant métier peut être constitué de :

- Classes pour les services simples (SISV pour service simple).
- Classes pour la DAO (Data Access Object) : ce sont les classes pour les accès aux données.
- Classes d'objets du domaine : DTO (Data Transfert Object), Criteria.



Figure 8 : Exemple de Composant métier Users

IV-1-2-1 - Multiplicité

1..n : on peut avoir un ou plusieurs composants.

IV-1-2-2 - Convention de nommage

Pour les composants métiers : **<<composant>> <nom du composant métier>**

Pour les composants Services : **<<service>> <nom du composant service>**

IV-1-2-3 - Contenu

Pour un composant métier :

- 1..n entités représentant les objets métiers du composant.
- 1..n diagrammes de classes manifestant ces entités et les associations qu'elles partagent avec des entités du même composant ou des entités d'autres composants métiers.

Pour un composant service :

- 1..n contrôles représentant les processus métiers du système.
- 1..n diagrammes de classes manifestant ces control et les associations qu'ils partagent avec des médiateurs donnant accès aux composants métiers.

IV-1-3 - Diagramme de Classe

IV-1-3-1 - Multiplicité

1..n : on peut avoir un ou plusieurs diagramme de classe.

IV-1-3-2 - Conventions de nommage

Si le diagramme de classes est le seul présent dans le package considéré, son nom doit être **<Structure>** .

S'il doit y avoir plusieurs diagrammes de classes dans un même package, les noms possibles sont les suivants :

- pour un diagramme présentant une partie de la structuration des classes généralement autour d'une classe donnée, le nom devra être : **Structure - <nom de la classe centrale >** .
- si autour d'une même classe plusieurs diagrammes doivent être présentés, le nom de chacun devra être : **Structure - <nom de la classe centrale > - <nom qualifiant l'objectif du diagramme>** .
- pour un diagramme présentant un arbre de généralisation-spécialisation particulier, le nom devra être :

Classification - <nom de la super-classe> .

IV-1-3-3 - Utilisation

Un ou plusieurs diagrammes de classes doivent être fournis pour chacun des packages (cf **IV-2**) présents dans les différentes couches du modèle statique d'analyse. Chacun présente un ensemble cohérent des classes du package éventuellement accompagnées d'un ensemble de classes d'autres packages du modèle avec lesquelles les premières partagent des associations (ou mieux, les interfaces d'autres composants), en fonction de l'objectif exact du diagramme tel qu'il est présenté par son nom et par sa documentation.

Par ailleurs des diagrammes de classes doivent également être utilisés pour représenter graphiquement les packages formant l'architecture logique en couches du système à ses différents niveaux et les relations de dépendance qui les structurent.

IV-1-3-4 - Documentation d'un diagramme de classe

Elle doit contenir une description de l'objectif du diagramme c'est-à-dire la raison pour laquelle le diagramme a été créé, de manière à manifester un ensemble particulier de classes avec un ensemble particulier de relations (associations, arbres de généralisation ou de spécialisation) qu'elles partagent.

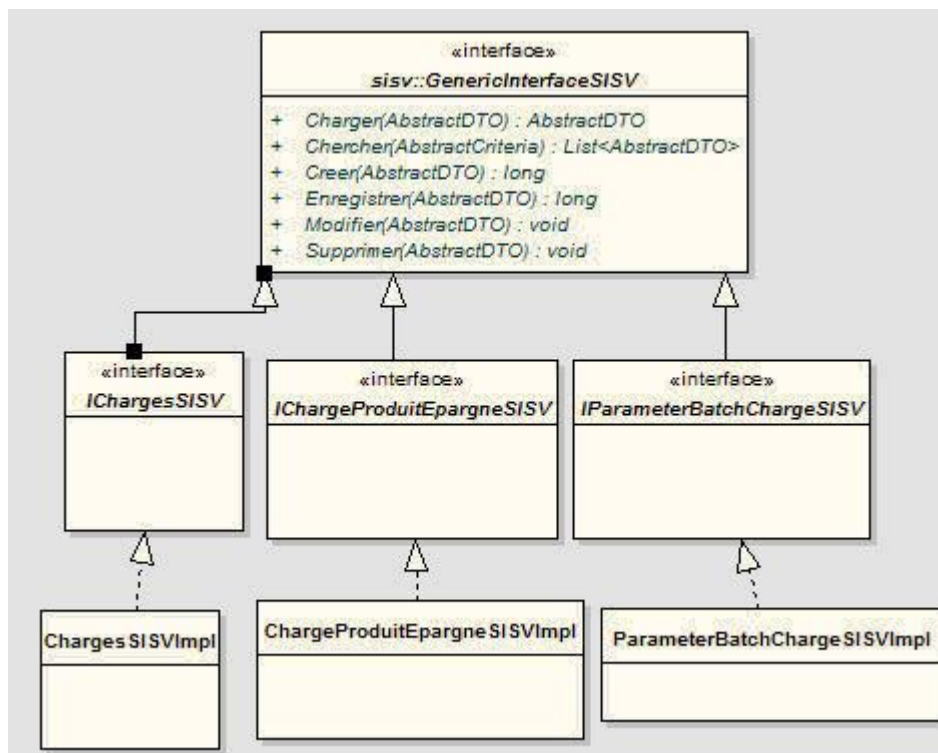


Figure 9 : Exemple de diagramme de classe de la couche SISV

IV-1-4 - Diagramme de Séquence

IV-1-4-1 - Multiplicité

1..n : On peut avoir un ou plusieurs diagrammes de séquence.

IV-1-4-2 - Conventions de nommage

SC<numéro d'ordre du scénario>-<type du scénario>-<nom du scénario>

Les différents paramètres du nommage ci-dessus sont identiques à ceux définies pour le diagramme de séquence de niveau système du modèle des cas d'utilisation (voir [III-1-4](#))

IV-1-4-3 - Utilisation d'un diagramme de séquence

En analyse les diagrammes de séquence spécifiés présentent les interactions entre instances des classes. Là où les diagrammes de séquence de niveau système exprimaient les scénarios de déroulement des cas d'utilisation avec une vue <boite noire> sur le système, les diagrammes de séquence du modèle dynamique d'analyse expriment ces mêmes scénarios en s'appuyant sur les objets du système tels qu'ils ont été définis. L'objet système est <explosé> pour faire apparaître ses constituants.

Chacun de ces diagrammes de séquence d'analyse doit être créé dans le package contenant la réalisation de cas d'utilisation correspondant au package de cas d'utilisation qui contient le diagramme de séquence de niveau système auquel il est lié.

Chacun diagramme de séquence présente :

- Les acteurs sous la forme d'une ou plusieurs de leurs instances.
- Les objets pertinents du système dans le cadre du scénario c'est-à-dire intervenant effectivement dans son déroulement sous la forme d'instances de classes du modèle statique d'analyse, quelle que soit la nature de ces classes (frontières, services ou entités). Globalement on trouvera dans chacun de ces diagrammes :

- Une instance de l'acteur initiateur du cas d'utilisation dont la réalisation est spécifiée à travers le diagramme de séquence considéré.
- Une ou plusieurs instances de classes frontières, notamment celle de la frontière assurant la médiation avec l'acteur initiateur mais également toutes celles des frontières représentant les médiateurs avec tous les autres acteurs impliqués dans le scénario considéré.
- Une ou plusieurs instances des classes contrôles responsables des services métier utilisés dans le cadre de la réalisation du scénario considéré.
- Une ou plusieurs instances des classes entités utilisées ou manipulées utilisés dans le cadre de la réalisation du scénario considéré.

- L'enchaînement des interactions, des actions et réactions, entre les acteurs et le système sous la forme de messages ordonnancés temporellement entre les instances des acteurs et les constituants du système.

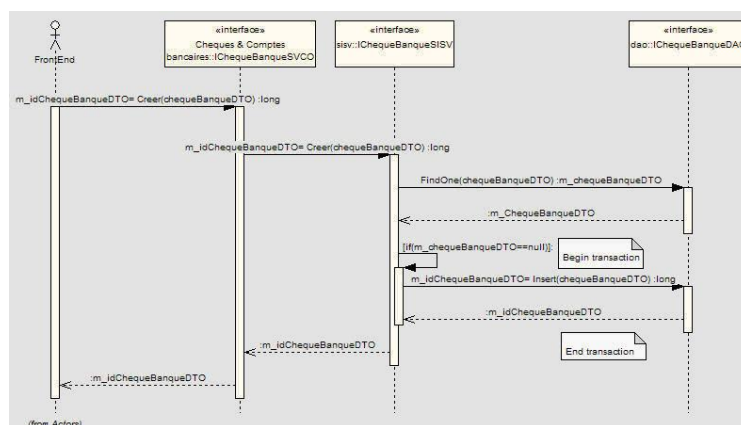


Figure 10 : Diagramme de séquence de SC10-Nominal-Enregistrer un chèque Banque

```

Notes
Enregistrement d'un chèque bancaire:
On doit pouvoir vérifier s'il n'existe pas déjà un chèque ayant le mm numéro provenant de la mm succursale.
Par défaut lors de la création d'un chèque on change son status à 'EN_ATTENTE'.
1/Le FE fait appel à la méthode à la méthode Créer(ChequeBanqueDTO) de IChequeBanqueSVCO
2/Au niveau de la SISV
On récupère la DTO entrée 'chequeBanqueDTO', on fait ceci:chequeBanqueDTO.Status=STATUS.EN_ATTENTE;
On fait ensuite appel à la méthode Créer(chequeBanqueDTO) de IChequeBanqueSISV
3/
//Vérification de l'unicité de ce chèque dans notre BD
On construit la DTO chequeBanqueDTO1 coe suit:
chequeBanqueDTO1.numero=chequeBanqueDTO.Numero;
chequeBanqueDTO1.CodeCompteBanque=chequeBanqueDTO.CodeCompteBanque
On fait appel à la méthode FindOne(chequeBanqueDTO) de IChequeBanqueDAO qui renvoi la DTO
m_ChequeBanqueDTO
4/Si m_ChequeBanqueDTO==null alors on fait appel à la méthode Insert(chequeBanqueDTO) de IChequeBanqueDAO
5/Sinon on throw le msg d'erreur M
M="Erreur lors de l'enregistrement du chèque:Ce chèque est déjà présent en base"

```

Figure 11 : Notes de l'UCR SC10-Nominal-Enregistrer un chèque Banque

IV-2 - Contenu du modèle d'analyse

Le modèle d'analyse est composé de quatre packages principaux :

- Le package des composants métiers
- Le package des médiations
- Le package des composants services
- Le package des Uses Case Realization

Chacun de ces packages représente une unité de regroupement d'entités fortement cohérentes autour d'un concept central dont on envisage qu'elles rendent collectivement des services

IV-2-1 - Package des composants métiers

C'est dans ce package qu'on regroupe tous les composants métiers du système. Rappelons que les composants métiers représentent la partie la plus stable du système.

IV-2-1-1 - Multiplicité

1..n : on peut avoir un ou plusieurs package de composants.

IV-2-1-2 - Nommage

Ce package porte le nom **<<layer>>BusinessComponent**. A l'intérieur de <<layer>>BusinessComponent on trouve tous nos composants métiers (cf)

IV-2-1-3 - Documentation

Elle doit contenir une description de l'intention du package c'est-à-dire des composants regroupés dans le package.

Pour un projet Serveur on peut par exemple avoir comme l'illustre la figure 12, un package de composants métiers :

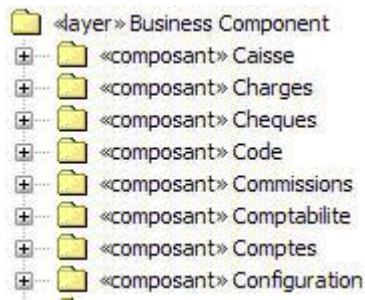


Figure 12 : Exemple d'un package de composant métiers pour un projet serveur

IV-2-2 - Package des composants services

C'est dans ce package qu'on regroupe tous les composants service du système. Rappelons que les composants service représentent la partie la plus instable du système c'est-à-dire celle en charge d'orchestrer les services proposés par la couche de composants métiers.

IV-2-2-1 - Multiplicité

1..n : on peut avoir un ou plusieurs packages services.

IV-2-2-2 - Nommage

Pour un projet serveur le package des services est symbolisé par **<<layer>>Services**. A l'intérieur on retrouve tous nos composants services (cf IV-1-2).

Pour un projet front-end ce pourrait être les classes d'actions qui sont en charge d'orchestrer les services offerts par le serveur. Les classes d'actions peuvent dans ce cas être regroupées dans **<<layer>><Front-End Action>**.

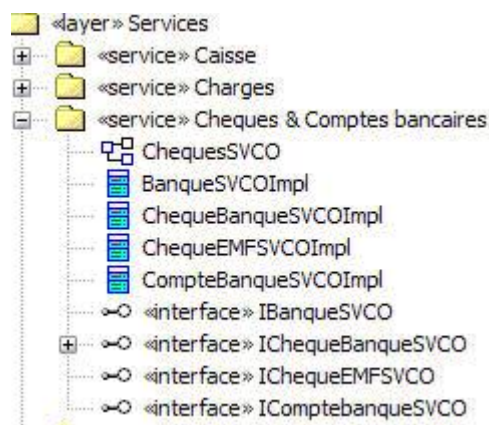


Figure 13 : Exemple de Services d'un projet Serveur.

IV-2-3 - Package des médiations

On y met les médiations entre les différents composants du système ainsi qu'entre le système et les acteurs extérieurs.

IV-2-3-1 - Multiplicité

1..n : on peut avoir un ou plusieurs package de médiation.

IV-2-3-2 - Nommage

<<interface>><nom du regroupement des médiations>

IV-2-3-3 - Contenu

1..n frontières (classes stéréotypées avec le stéréotype standard <boundary>) représentant les objets médiateurs des interactions entre les composants ainsi qu'entre le système et ses acteurs.

Pour un projet Serveur on peut avoir un médiateur pour l'accès aux composants service : SVCOFinder. Il peut aussi exister un médiateur pour l'accès aux composants métiers : SISVFinder. Il peut enfin exister un médiateur pour l'accès aux entités d'accès aux données : DAOController.

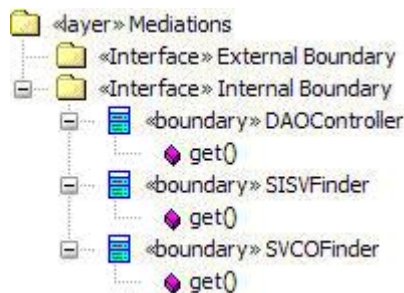


Figure 14 : package médiation d'un projet Serveur

IV-2-4 - Package Use Case Realization

Le contenu de ce package forme le modèle dynamique d'analyse.

IV-2-4-1 - Package des Regroupements de réalisation des cas d'utilisation

Un tel package regroupe un sous-ensemble des réalisations de cas d'utilisation correspondant aux sous-ensembles de cas d'utilisation identifiés dans le modèle des cas d'utilisation.

IV-2-4-1-1 - Multiplicité

1..n : on peut avoir un ou plusieurs packages de regroupement des réalisations de cas d'utilisation.

IV-2-4-1-2 - Convention de Nommage

<<UCR Package>> <nom du regroupement de réalisations de cas d'utilisation>

Un package de regroupement des réalisations de cas d'utilisation doit être créé par package de regroupement de cas d'utilisation du modèle des cas d'utilisation en portant le même nom que ce dernier.

IV-2-4-1-3 - Contenu

- 1..n packages de réalisation de cas d'utilisation (voir figure 15).
- 1 diagramme de cas d'utilisation manifestant la traçabilité des cas d'utilisation par leurs réalisations respectives.



Figure 15 : Package d'UCR

IV-2-4-2 - Package de réalisation de cas d'utilisation

Chacun de ces packages représente une unité de modélisation cohérente détenant la spécification de la réalisation d'un cas d'utilisation sur la base des objets identifiés dans le système. La réalisation d'un cas d'utilisation consiste à <éclater> l'entité <Système> présente sur les diagrammes de séquence du cas d'utilisation origine afin de faire apparaître les classes boundary, service et métier qui constituent le système.

IV-2-4-2-1 - Multiplicité

1..n : on peut avoir un ou plusieurs packages de réalisation de cas d'utilisation.

IV-2-4-2-2 - Convention de Nommage

<<Use Case Realization>> <nom de la réalisation de cas d'utilisation>

Un package de réalisation de cas d'utilisation doit être créé par package de cas d'utilisation du modèle des cas d'utilisation en portant le même nom que ce dernier (à l'exception du préfixe qui doit être <UCR-> et non <UC-x->).

IV-2-4-2-3 - Contenu

- 1 réalisation de cas d'utilisation (cas d'utilisation stéréotypé)
- 1..n diagrammes de séquence manifestant chacun un scénario de réalisation du cas d'utilisation (nominal, alternatif ou d'erreur) faisant intervenir des instances des classes identifiées dans le modèle d'analyse.

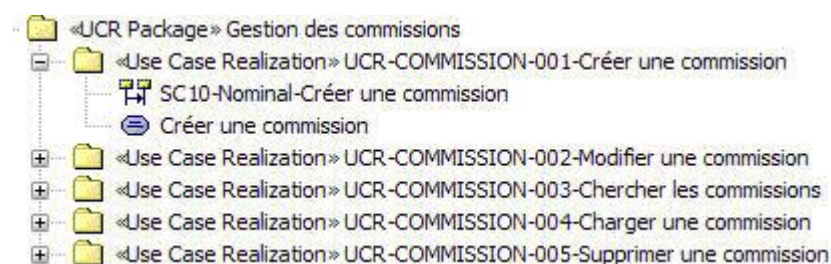


Figure 16 : Contenu d'un package de réalisation de cas d'utilisation

V - Cas pratique : Modélisation UML d'un projet simple

V-1 - Spécifications fonctionnelles du projet

GestClient est une application de gestion des clients. L'application doit permettre à un utilisateur de rôle Administrateur de :

- créer un client
- rechercher un ou des clients selon certains critères.

- modifier un client
- désactiver un client

Ce projet va comporter une partie serveur et une partie front-end. La partie serveur va exposer ses services au front-end, ce dernier pouvant par exemple être une winapp ou une webapp.

V-2 - Modélisation de la partie serveur : GestClient.Server

V-2-1 - Use Case Model

V-2-1-1 - Package Actor

On distingue un acteur externe qui est ici le Front-end.

V-2-1-2 - Package Context

On distingue l'application serveur et la base de données.

V-2-1-3 - Package Use Cases

Nous avons un seul package de gestion des clients : <<UC Package>>CLIENT-Gestion des clients. Ce package contient 5 use cases tels que le présente la figure ci-dessous :

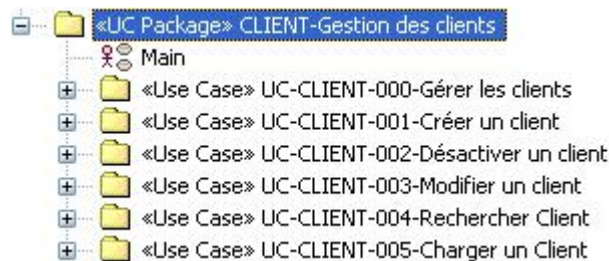


Figure 32 : UC Package Gestion des clients

Ci-dessous le diagramme de cas d'utilisation du <<UC Package>>CLIENT-Gestion des clients

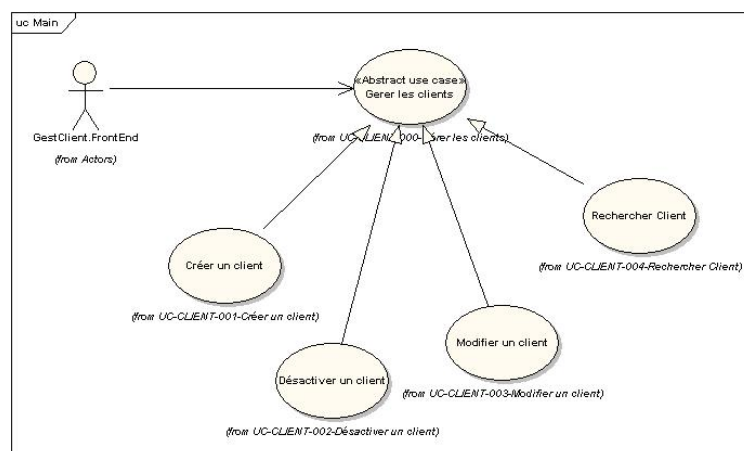


Figure 33 : Diagramme de cas d'utilisation du UC Package CLIENT-Gestion des clients

Pour chaque UC on distingue un diagramme de séquence pour le cas nominal, comme le montre la figure ci-dessous :

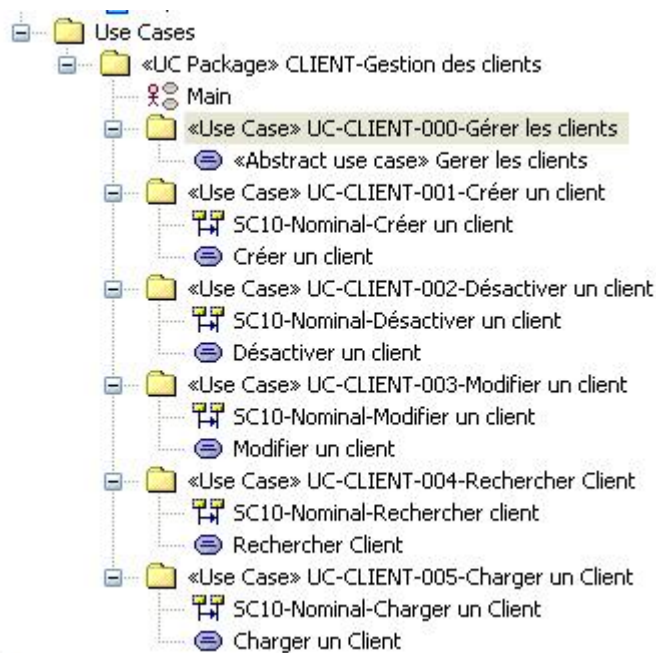


Figure 34 : Diagramme de séquence Nominal du UC Package CLIENT-Gestion des clients

La figure ci-dessous montre le diagramme de cas nominal du use case UC-CLIENT-004-Rechercher Client

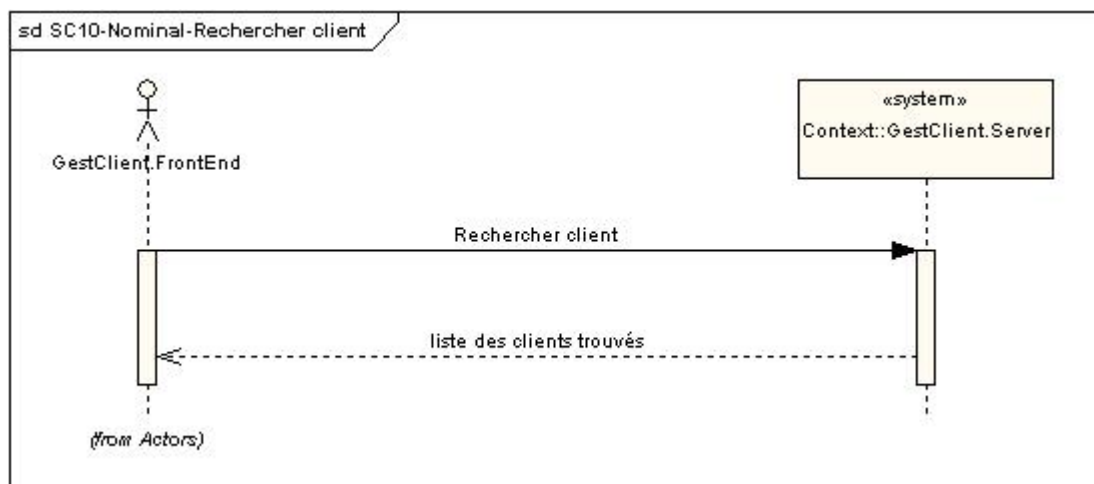


Figure 35 : diagramme de cas nominal du use case UC-CLIENT-004-Rechercher Client

V-2-2 - Modèle d'analyse

V-2-2-1 - Data Model

C'est dans cette partie qu'on conçoit le modèle de donnée du projet.
Le Data Model fera l'objet d'un article ultérieur.

V-2-2-2 - Package des composants métiers

Nous avons un seul composant métier : <<composant>>Client. Ce composant comprend un ensemble d'éléments présentés par la figure ci-dessous :

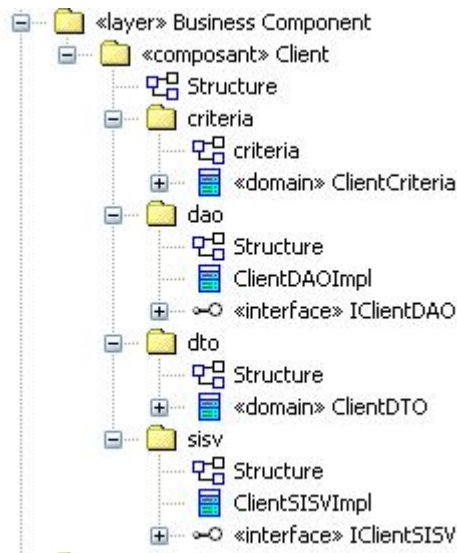


Figure 36 : composant Client

- <<domain>>ClientCriteria : contient les critères de recherche d'un client
- <<domain>>ClientDTO : encapsule des données d'un client
- <<interface>>IClientDAO : pour l'accès à la base de données
- <<interface>>IClientSISV : service simple implémentant la logique métier.

V-2-2-3 - Package des composants services

Nous disposons d'un composant de services :

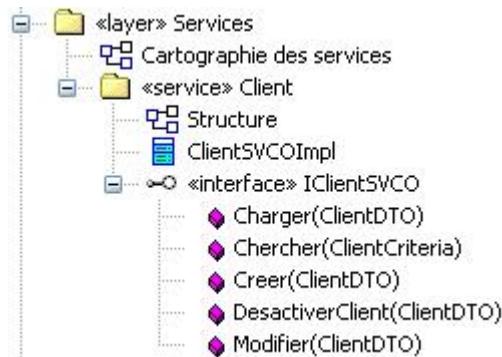


Figure 37 : service Client

V-2-2-4 - Package des médiations

On distingue trois médiateurs permettant de mettre en relation des différentes couches de l'application :

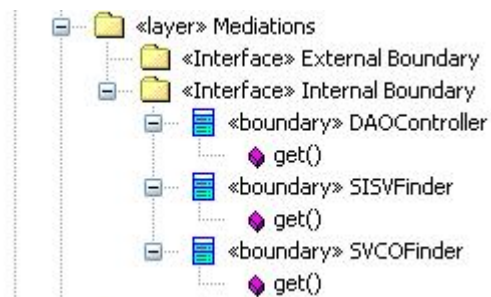


Figure 38 : layer Médiations

- DAOController : médiateur pour l'accès aux services de la couche DAO
- SISVFinder : médiateur pour l'accès aux services de la couche SISV
- SVCOFinder : médiateur pour l'accès aux services de la couche SVCO

V-2-2-5 - Package des réalisations des cas d'utilisation

Pour chaque use case on a un "Use Case Realization" correspondant tel que le présente la figure ci-dessous :

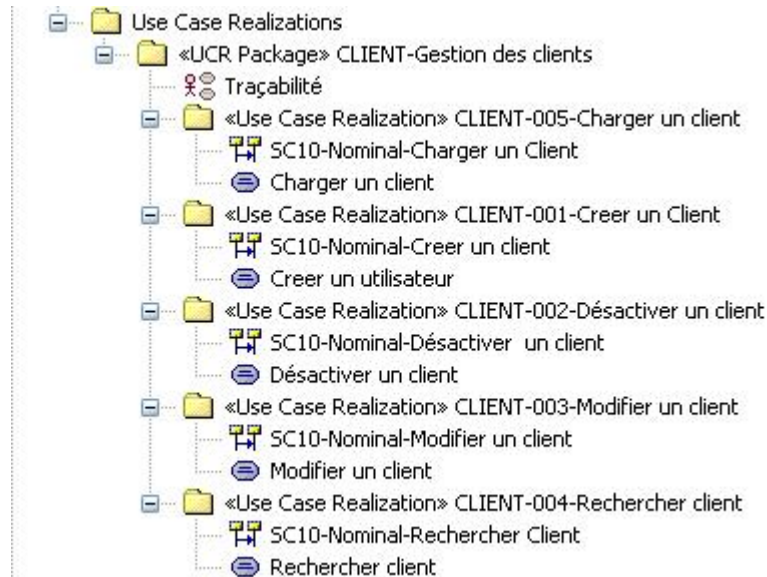


Figure 39 : Use Case Realization gestion des clients

La figure ci-dessous présente le diagramme de cas réalisation nominal de création d'un client :

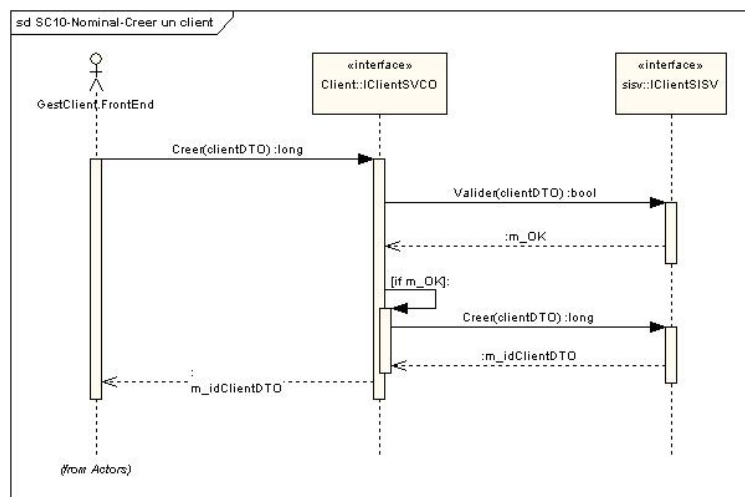


Figure 40 : SC10-Nominal-Creer un client

V-2-3 - Téléchargement du projet UML de la partie serveur de l'exemple

Le projet UML de la partie serveur de l'exemple est disponible sur le lien suivant : **projet UML du serveur**
Il a été fait à l'aide de l'outil Enterprise Architect dont une version d'évaluation est disponible sur le lien suivant : **Enterprise Architect 30 days trial**

V-3 - Modélisation du Front End : GestClient.FrontEnd

V-3-1 - User Interface Model

Cette partie est réservée à la modélisation des interfaces utilisateurs d'un projet front-end. Le User Interface Model fera l'objet d'un article ultérieur.

V-3-2 - Use Case Model

V-3-2-1 - Package Actor

On distingue un acteur humain qui est ici l'administrateur.

V-3-2-2 - Package Context

C'est le bloc constitué par la winform.

V-3-2-3 - Package Use Cases

On distingue 3 regroupements de cas d'utilisation :

- <<UC Package>>Connexion : traite les cas d'utilisations liés au module de Connexion
- <<UC Package>>Gestion des clients : traite des cas d'utilisations liés au module de Clients
- <<UC Package>>Welcome : traite des cas d'utilisations liés au module de page d'accueil (ou Welcome)

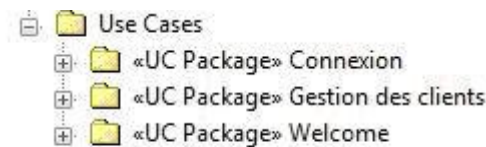


Figure 17 : UC packages du projet GestClient.FrontEnd

V-3-2-3-1 - <<UC Package>>-Connexion

Il contient à son tour le regroupement <<UC Package>>LOGIN-Connexion comme le montre la figure 18.

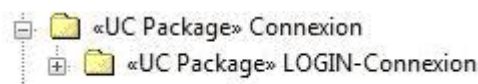


Figure 18 : UC Package - Connexion

<<UC Package>>LOGIN-Connexion contient les UC suivants :

- UC-LOGIN-001-Initialiser et afficher la page de login
- UC-LOGIN-002-S'authentifier via le login et le mot de passe
- UC-LOGIN-003-Fermer la page de login

Tous ces UC étendent le 'UC-LOGIN-000-Se connecter' qui est quant à lui un UC abstrait.

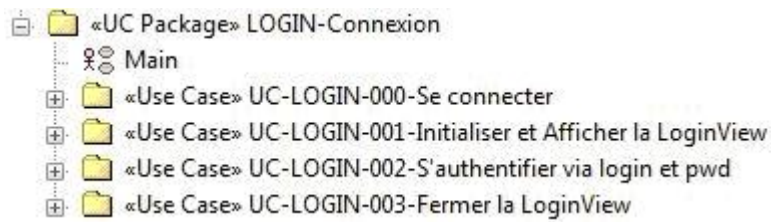


Figure 19 : UC Package LOGIN-Connexion

La figure 20 nous montre le diagramme de cas d'utilisation pour <<UC package>>LOGIN-Connexion

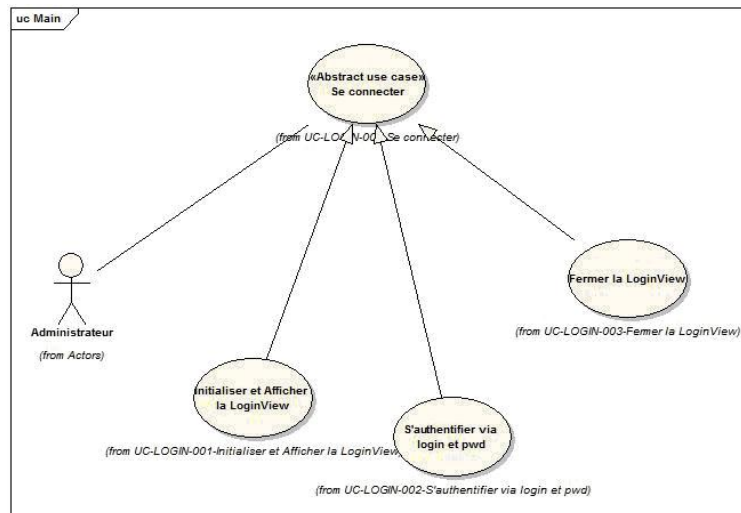


Figure 20 : diagramme de cas d'utilisation pour <<UC package>>LOGIN-Connexion

Dans chaque UC on distingue un diagramme de séquence pour le cas nominal, comme le montrent la figure 21 et la figure 22

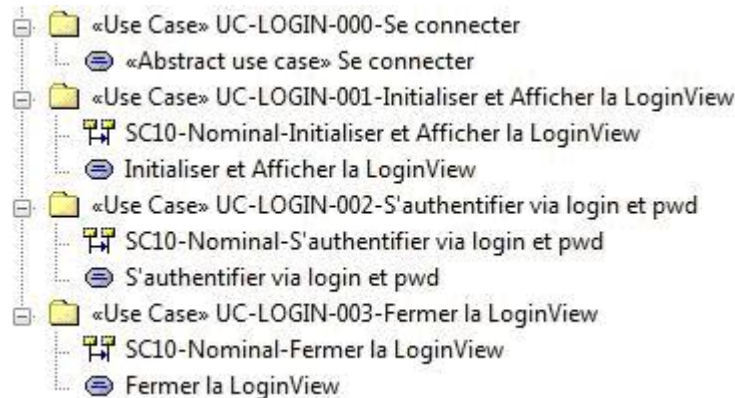


Figure 21 : Diagrammes de séquence Nominal du UC Package LOGIN-Connexion

La figure 22 montre le diagramme de séquence "Nominal SC10-Nominal-S'authentifier via login et pwd".



Figure 22 : SC10-Nominal-Connexion : Use Case S'authentifier via login et pwd

V-3-3 - Modèle d'analyse

V-3-3-1 - Data Model

C'est dans cette partie qu'on conçoit le modèle de donnée du projet. Le Data Model fera l'objet d'un article ultérieur.

V-3-3-2 - Package des composants métiers

Le package des composants métiers <<Layer>>BusinessComponent contient tous les composants métiers du projet. Dans le cas du projet front-end nous avons le <<Composant>>Client qui contient une <<interface>>IClientSVCO : il s'agit d'un service fourni par le serveur.

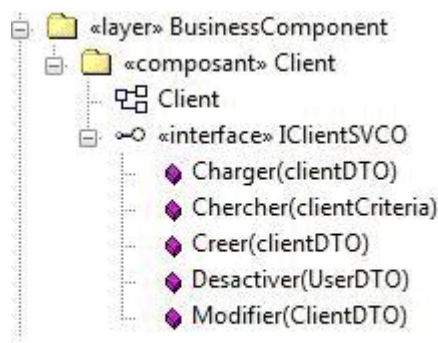


Figure 23 : <<Layer>>BusinessComponent

V-3-3-3 - Package des composants services

Comme c'est une application front-end, ici on parle de "<<Layer>>Front-End Controller" et on y présente tous les contrôleurs de l'application. Les contrôleurs sont regroupés par modules :

- <contrôleur>Client dans le module Client
- <contrôleur>Login dans le module Connexion
- <contrôleur>Welcome dans le module Welcome

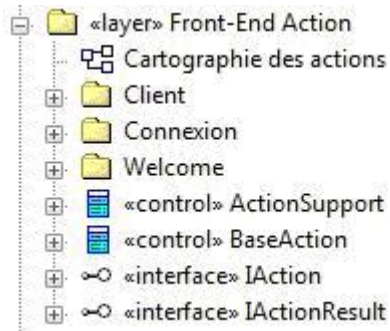


Figure 24 : <<layer>>Front-End Controller

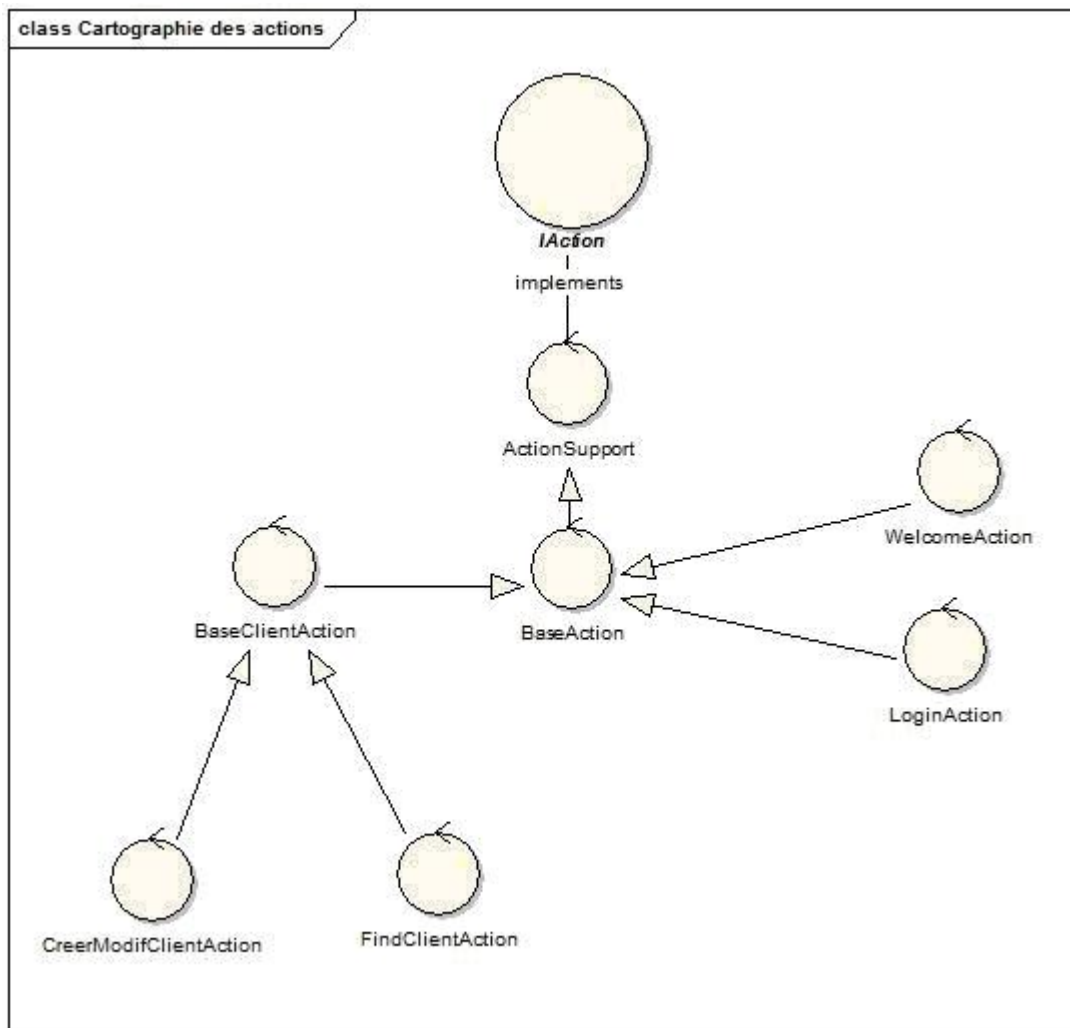


Figure 25 : Diagramme de classes des contrôleurs

<controller>Login contient <<control>>LoginController. Ce contrôleur contient les méthodes suivantes :

- EnterFromLoginPwd() : méthode publique qui effectue l'authentification sur la base du login et mot de passe.
- Fermer() : méthode publique pour fermer la page de login et quitter l'application.

V-3-3-4 - Package des médiations

Ici sont représentées les médiations entre les différents composants du système ainsi qu'entre le système et les acteurs extérieurs.

On distingue :

- les écrans de l'application : médiation entre l'utilisateur et l'application
- la médiation entre le FrontEnd et le BackEnd.
- la médiation entre les actions des utilisateurs et leurs traitements.

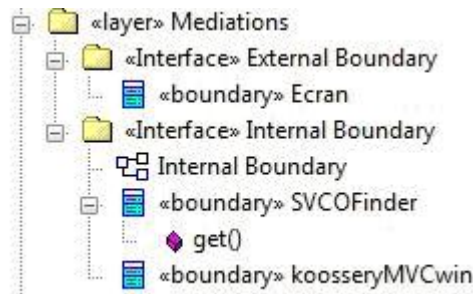


Figure 26 : Package des médiations

V-3-3-5 - Package des réalisations des cas d'utilisation

Les différents packages recensés pour ce projet sont :

- <<UCR Package>>-Connexion : traite les cas d'utilisations liés au module de Connexion.
- <<UCR Package>>-Gestion des clients : traite des cas d'utilisations liés au module de Clients.
- <<UCR Package>>-Welcome : traite des cas d'utilisations liés au module de page d'accueil.

Ces packages sont organisés comme le présente la figure 27 ci-dessous :

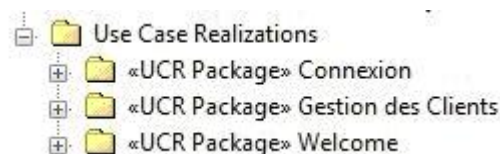


Figure 27 : UCR packages du projet GestClient.FrontEnd

V-3-3-5-1 - <<UCR Package>>-Connexion

Il est organisé comme présenté dans les figures ci-dessous :

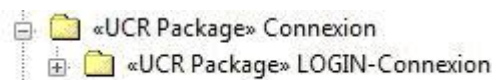


Figure 28 : UCR Package Connexion

<<UCR package>>LOGIN-Connexion contient les Use Case de réalisation ci-dessous :

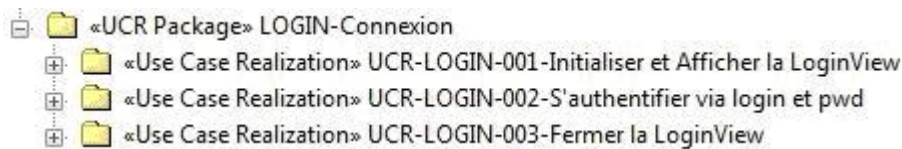


Figure 29 : UCR Package LOGIN-Connexion

Chaque UCR contient un diagramme de séquence représentant le cas Nominal.

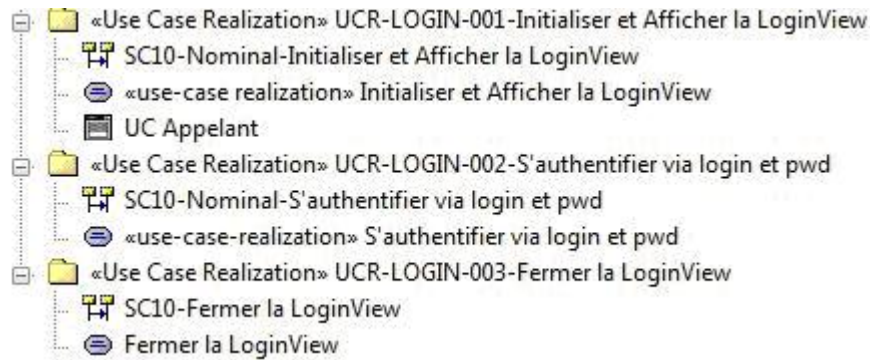


Figure 30 : UCR package LOGIN-Connexion

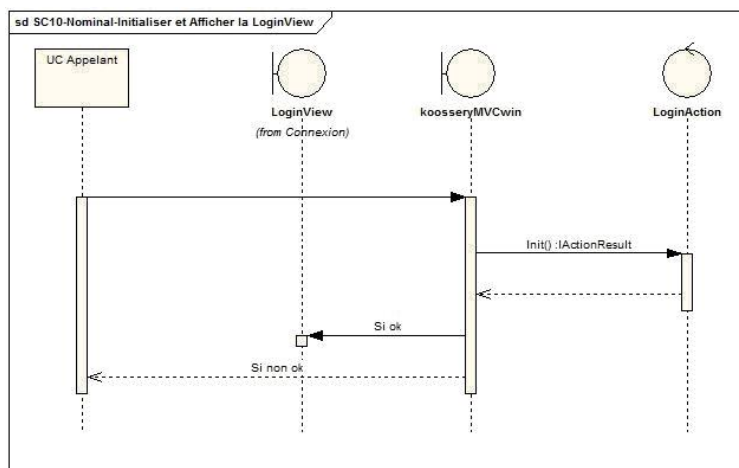


Figure 31 : SC10-Nominal-Connexion : use case S'authentifier via login et pwd

V-3-4 - Téléchargement du projet UML de la partie front-end de l'exemple

Le projet UML de la partie front-end de l'exemple est disponible sur le lien suivant : **projet UML du front-end**
Il a été fait à l'aide de l'outil Enterprise Architect dont une version d'évaluation est disponible sur le lien suivant : **Enterprise Architect 30 days trial**

Koossery Technology est une entreprise spécialisée dans les développements de projets au forfait (.NET & Java J2ee) et dispose pour cela d'une cellule Architecture Technique et Outillages ainsi que d'une Usine Logicielle.

Koossery Technology dispose aussi d'un Centre de Service GED-ECM-BPM Alfresco, jBPM pour adresser les projets GED-ECM-BPM sur la base des plate-formes Alfresco et jBoss BPM.

Tous nos remerciements au comité de relecture **developpez.com**

Vos réactions par l'intermédiaire de ce fil :