Académie de Montpellier

Université Montpellier II
Sciences et Techniques du Languedoc

MÉMOIRE DE STAGE DE MASTER 2

effectué au laboratoire

Agrotechnology & Food Science Group,

Wageningen University & Research Center

Spécialité :
Professionnelle et Recherche unifiée en Informatique

Une application Smartphone pour un système de recommandations alimentaires personnalisées

par **Anne Tireau**

Soutenu le **13 Septembre 2010**

Sous la direction de : Nicole Koenderinck

Devant le jury composé de :

Président	:	Rodolphe Giroudeau
Rapporteurs	:	Isabelle Mougenot
	:	Abdelkader Gouaich

Remerciements

Je tiens à remercier, dans un premier temps, toute l'équipe pédagogique de l'université Montpellier II et les intervenants professionnels responsables du MASTER Informatique IFPRU, pour avoir assuré la partie théorique de celle-ci.

Je remercie également Madame Nicole Koenderinck ma tutrice au sein du département de recherche *Food & Biobased Research* à l'Université de Wageningen (Pays-Bas) pour son accueil et son encadrement tout au long de ce stage.

Je tiens à remercier tout particulièrement et à témoigner toute ma reconnaissance à Madame Isabelle Mougenot pour son intérêt pour mon travail, ses conseils, son aide et sa disponibilité.

Un grand merci à toute l'équipe *Information Management* pour m'avoir fait découvrir un nouveau laboratoire de recherche, une autre façon de travailler et aussi une autre culture. Plus particulièrement, je remercie Hajo pour avoir facilité mon intégration et pour les discussions variées.

Pour finir, je n'oublie pas mes collègues de l'UMR MISTEA qui m'ont accueillie il y a 5 ans déjà dans leur équipe et qui m'ont encouragée à faire ce Master. Je les remercie pour tout leur soutien et leur aide précieuse. Je pense particulièrement à Brigitte, Nadine et Pascal.

Table des matières

Introduction

Ce stage de Master 2 s'est déroulé dans le département de recherche *Food & Biobased Research* à l'Université de Wageningen (Pays-Bas). L'objectif appliqué était de proposer aux clients d'un restaurant - self service expérimental , le Restaurant du Futur, un prototype de système de diffusion de conseils alimentaires qui convienne à leur mode de vie, leurs caractéristiques physiques et physiologiques ainsi qu'à leurs goûts alimentaires. Pour cela, il s'est avéré nécessaire d'avoir un environnement de connaissances qui représente les différentes connaissances sur les menus (produits ou aliments associés, leurs caractéristiques et valeurs nutritives) et sur les consommateurs référencés (caractéristiques physiques, préférences, mode de vie, objectifs diététiques, etc...), et qui permette de délivrer des conseils diététiques personnalisés. Pour aider le consommateur dans sa démarche, il est important que lorsqu'il se déplace dans le restaurant pour faire ses choix, il dispose de l'information pertinente. La création de cet environnement a constitué la première étape du stage. L'objectif à long terme est la prise en compte de la localisation des produits et du consommateur pour lui donner des conseils en temps réel, adaptés à la gestion des tentations possibles et respectant ses objectifs diététiques. Une des originalités de l'approche est de proposer une application sur un environnement mobile, tel qu'un *Smartphone*.
Pour ce faire, une ontologie sera construite pour chaque pan de connaissance, des données seront élicitées et des ontologies locales définies à partir de ces données spécifiques. Dans ce but, les technologies et langages du Web Sémantique sont mises à profit. L'ensemble de ces ontologies est exploité au sein de l'environnement de connaissance. L'attention est portée sur la définition d'une architecture appropriée à leur exploitation par des usagers finaux aux travers de plateformes mobiles.

La démarche générale sera testée et validée au travers de la réalisation d'un prototype. Ce prototype est développé à l'aide de différents outils : l'éditeur d'ontologie TopBraid, le système de gestion de connaissances Sesame, les langages du Web Sémantique RDF/S, OWL et SPARQL, ainsi qu'un Web Service RestFul basé sur le framework Restlet et pour la plateforme mobile Windows Phone 7 Silverlight, XAML et C#.

Le rapport est structuré de la façon suivante : le contexte du stage et les objectifs informatiques dégagés font l'objet du premier chapitre. Les principaux outils utilisés sont

introduits dans l'état de l'art au chapitre 2. Les modèles de connaissance que nous avons développés sont décrits au chapitre 3. Toute la partie mise en œuvre est détaillée dans le chapitre 4, de l'architecture du prototype du système de diffusion des conseils personnalisés à l'application sur Smartphone. Des perspectives sont ensuite dégagées autour de plusieurs questions de recherche liées aux ontologies. La conclusion, chapitre 5, clôt la partie principale du mémoire. Des chapitres annexes sont donnés pour compléter l'information, tant technique que associée à l'application (diététique, préférences).

Chapitre 1

Contexte et objectifs

1.1 Contexte du domaine applicatif

Le contexte dans lequel ce stage a été réalisé est présenté ici. Il s'agit d'en expliquer les grandes lignes afin de justifier les choix conceptuels ou technologiques qui seront faits par la suite. Nous introduisons aussi le projet FOVEA sans lequel ce stage n'aurait pas eu lieu.

Les gouvernements des pays occidentaux ont mis en place de vastes campagnes d'information pour sensibiliser les populations à leur mode alimentaire. L'idée est donc de poser les bases d'une vie plus saine sans rien imposer. Ces campagnes, qui passent par exemple par la diffusion de spots publicitaires, coûtent cher et ne récoltent que des résultats plutôt médiocres. Or il est postulé que l'efficacité du message serait plus grande si ce message était personnalisé. En effet, les conseils donnés jusqu'alors sont des conseils généraux portant soit sur l'alimentation, soit sur la santé et rencontrent un accueil assez limité de la part de la population. Les principaux reproches qui sont faits à ces conseils, par trop généraux, sont les suivants :

- ils ne tiennent compte ni des préférences ou aversions alimentaires, ni des intolérances ou allergie alimentaires éventuelles ;
- ils ne considèrent, en aucun cas, les dépenses physiques journalières de la personne ;
- ils sont en arrière plan au moment du choix ;
- ils ne se doublent pas d'un travail de prévention portant sur une évaluation des mauvaises habitudes alimentaires de tout un chacun.

Dans ce contexte, le projet *Food Valley Eating Advisor (FOVEA)*[1] a été mis en place et a pour un de ses objectifs, la création de la base d'**un système de diffusion d'informations** axé sur la personne. Ce système est attendu de fournir à toute personne **des**

1. http://www.fovea-project.org/UK/

conseils personnalisés pour **une alimentation plus saine** et une activité physique plus adaptée.

Le département de recherche *Food & Biobased Research* est l'un des partenaires du projet FOVEA. Ce département fait partie du centre *Wageningen University & Research (WUR)*[2] aux Pays-Bas. Ses domaines d'expertise portent sur les sciences sensorielles et le comportement alimentaire d'une part et sur les systèmes dit "intelligents" et la recherche appliquée en vision d'autre part. Cette double compétence en fait un département de recherche de choix pour la bonne conduite du projet FOVEA. J'ai été plus spécifiquement accueillie dans l'équipe ***Information Management***[3] dont les activités sont ancrées dans le second axe "systèmes intelligents". Un système dit intelligent est vu, ici, comme un système automatisé enrichi d'une couche logicielle traitant de la connaissance afin d'en élargir le périmètre de fonctionnalités. Dans cette optique et de concert avec trois entreprises privées, le département de recherche *Food & Biobased Research*, a créé en 2007, le *Restaurant du Futur*[4], pour mener à bien ses recherches sur le comportement des consommateurs et les moteurs qui influent sur leurs choix alimentaires. Dans le projet FOVEA, *Food & Biobased Research* se positionne à l'échelle du restaurant pour répondre de manière appliquée et proposer un cas d'étude bien réel à la problématique du sujet. A partir des études menées dans ce cadre, les chercheurs de *Food & Biobased Research* développent des modèles conceptuels orientés vers de la personnalisation de conseils alimentaires en vertu de différents facteurs : physiologiques, ethniques, financiers, éducationnels, liés aux modes de vie, etc. Le restaurant sert alors de domaine d'étude et vient instancier les modèles construits. Il est alors plus facile de valider les modèles construits.

Le Restaurant du Futur est un restaurant d'entreprise, en libre-service, situé sur le centre de l'Université de Wageningen (cf. figure 1.1). Ce restaurant est, avant tout, une zone d'expériences et d'études où les expérimentations en laboratoire portent sur le domaine sensoriel des aliments : odeur, couleur, goût, etc.. La salle principale du restaurant, quant à elle, permet d'étudier le comportement des consommateurs et leurs choix alimentaires durant un repas. Cela peut inclure l'étude de l'influence des paramètres liés aux produits : composition, goût, emballage, présentation ou informations associées (prix, nom, etc.). L'environnement est aussi pris en compte : agencement de la salle, couleur des buffets, des tables et des chaises, circulation dans le restaurant, influence de la luminosité, de la température ambiante... Ces informations sont collectées grâce à des capteurs, notamment des caméras dont les séquences vidéos sont analysées et annotées. A cela s'ajoutent des caisses enregistreuses, reliées à des systèmes de gestion

2. http://www.wur.nl/UK/
3. http://www.fbresearch.nl/InformationManagement/
4. http://www.restaurantvandetoekomst.wur.nl/UK/

de données, qui consignent les achats des consommateurs. En outre, chaque nouveau client peut s'inscrire et répondre à un questionnaire (cf. annexe A). Celui-ci permettra aux chercheurs d'avoir une idée du mode de vie des usagers ou des relations entretenues avec les aliments (détection de néophobie [5], par exemple). Cette démarche d'acquisition et de valorisation d'information est importante et difficile, car l'étude du comportement implique que la collecte des informations sur les consommateurs soit la moins intrusive possible pour ne pas influencer leurs comportements. Ceci va aussi nécessiter de l'expertise dans le domaine des nouvelles technologies de l'information et de la communication, afin d'optimiser les traitements et leurs exploitations. C'est dans ce domaine qu'intervient l'équipe *Information Management*. Dans ce contexte, les projets actuellement en cours de réalisation portent sur des systèmes de reconnaissance visuelle de personnes, de comportements et de produits ou sur le système d'intégration de données.

FIGURE 1.1 – Restaurant du Futur : buffet de la salle principale, caisse enregistreuse, salle de contrôle des caméras

La personnalisation des conseils va aussi dépendre des connaissances disponibles sur le consommateur, dans le cadre du restaurant.

Pour le consommateur, il pourra s'agir de :

- ses caractéristiques physiques et physiologiques,
- ses préférences alimentaires (goût / aversion),
- l'historique de consommations des derniers jours,
- ses habitudes alimentaires,

5. Néophobie : crainte de tout ce qui est nouveau.

– ses objectifs diététiques (régime végétarien ou hypocalorique par exemple),
– son mode de vie (célibataire, déplacements fréquents, etc.),
– son activité sportive,
– son comportement face à la nourriture (importance des repas dans sa vie, etc.).

Pour le Restaurant du Futur, la personnalisation va dépendre de l'environnement mais surtout des connaissances possédées sur les produits disponibles :

– la composition des menus,
– les produits avec leurs caractéristiques et leurs valeurs nutritives. Notons que les caractéristiques peuvent être assez fines, car elles concernent également tout ce qui fait qu'un produit peut attirer ou rebuter un consommateur (goût amer, odeur de vanille...).

1.2 Objectifs informatiques

Modèles de connaissances : l'objectif du projet est de proposer les premières briques du système de conseils alimentaires personnalisés, pour le Restaurant du Futur. L'une des exigences du système, et l'un des enjeux informatiques ici, est **la formalisation des connaissances** des différents domaines nécessaires. Cette formalisation va permettre une exploitation plus fine et poussée des informations et des données à disposition. Par exemple, elle pourra faciliter l'accès à l'information et à l'inférence de nouveaux faits. La particularité de ces modèles, ou **composants ontologiques** dans notre cas, est qu'ils devront être capables d'être combinés de manière intelligente, afin d'être exploitables par le modèle de conseils. De plus, ils devront pouvoir évoluer et être enrichis pour s'orienter vers un système de plus en plus complet.

Dans cette problématique, mon stage a tout d'abord consisté à proposer les premiers modèles de connaissances, en se basant pour commencer sur l'existant, c'est-à-dire les données et les informations déjà disponibles. Dans un second temps, l'objectif du stage était de mettre en œuvre ces composants ontologiques et de concevoir un premier prototype du système de diffusion des conseils personnalisés ou **PEA** pour ***Personal Eating Advisor*** en anglais.

Les exigences du système PEA sont triples : le système doit personnaliser les messages, donner des retours à l'utilisateur sur ses choix et être mobile afin d'être présent au moment du choix. L'idée est de proposer un début de solution aux limitations des conseils généraux diffusés actuellement. Le but ici est avant tout pédagogique.

La personnalisation du message diffusé peut influencer son efficacité. Elle va être guidée par les recommandations du modèle de conseils, en fonction des préférences de l'utilisateur. Il s'agit de la forme et du contenu du message :

1.2 Objectifs informatiques

Forme : données textuelles, images, diagrammes, couleurs... Cette idée provient des conclusions de l'étude de l'influence des informations nutritives de Stubenitsky et al. (2000), et suppose que le type de format d'information utilisé pourrait influencer l'efficacité du message. Par exemple, *le nombre* de calories (information textuelle) d'un aliment correspondant à un plat allégé, n'aura pas forcément de portée sur la personne qui n'a pas la notion de la ration qui lui est nécessaire.

Contenu : c'est l'information pertinente pour le consommateur. Il s'agit ici d'être en accord avec le régime diététique du consommateur. Par exemple, donner la teneur en sel d'un produit reste surtout pertinent pour l'utilisateur qui suit un régime dit "sans sel".

Il en résulte, que cette personnalisation servira de guide pour la partie IHM [6] du système de diffusion de l'information.

Les réactions du système au choix du consommateur sont aussi des fonctionnalités attendues. Elles peuvent être vues comme une première solution au manque de retour à l'utilisateur sur ses mauvais choix. Comme par exemple, la proposition d'une alternative s'avérant plus pertinente que le choix opéré par le client. Une autre solution est la mise en place du concept de ***plateau virtuel***, ou *Tray* en anglais. Avant même de se trouver dans la cantine, l'utilisateur pourra ajouter ou supprimer des aliments de son plateau virtuel et consulter une évaluation globale de ses choix : résumé de l'apport calorique de son repas pré-sélectionné, par exemple, ou réception d'une alerte au dépassement d'un seuil fixé et personnalisé. Ainsi, il pourra ajuster ses choix en fonction d'un plateau virtuel et de ses objectifs.

La mobilité de la solution est aussi une exigence importante pour aider la personne, et qui va influencer le choix de l'architecture du système. En effet, dans le cas de l'utilisation du plateau virtuel, vu précédemment, la mobilité permettrait à l'utilisateur d'optimiser la composition de son plateau pendant ses choix. Cela est vrai aussi de façon plus générale, pour aider la personne à améliorer son comportement. Un des phénomènes à prendre en compte est celui de la tentation. Il influence énormément les choix du consommateur et dépend souvent du contact avec les produits. Ce qu'aimeraient les chercheurs, c'est de pouvoir, dans une prochaine version, coupler le conseiller à un système de géolocalisation des produits et du consommateur afin de le prévenir s'il s'approche de produits estimés comme "dangereux" ou au contraire de lui proposer un chemin optimal vers les produits bénéfiques. De ce fait, nous avons choisi une plate-forme mobile, type ordiphone ou *Smartphone* en anglais, pour restituer les conseils. En effet, ils allient la capacité et les fonctionnalités nécessaires.

6. IHM : Interface Homme-Machine

L'objectif de mon stage est donc double. Premièrement, il s'agit de modéliser les composants ontologiques des domaines concernant une vaste préoccupation humaine : à savoir l'alimentation. Deuxièmement, il est aussi question de proposer un prototype d'architecture et d'application sur Smartphone capable de restituer au consommateur le menu avec les conseils personnalisés et s'appuyant sur ces composants ontologiques.

La solution proposée est schématisée par la figure 1.2 où les principaux acteurs du prototype sont représentés avec leurs rôles. Un environnement de connaissances associé aux données collectées sera exploité par un modèle de conseils alimentaires personnalisés que nous avons considéré comme une boîte noire. Sa tâche est de fournir les recommandations nécessaires à la constitution d'un menu issu de notre environnement et personnalisé, c'est-à-dire dont les informations seront restituées de manière pertinente pour le consommateur. La plateforme choisie pour l'utilisateur est le Smartphone afin de l'aider lors de ses choix au sein du Restaurant du Futur.

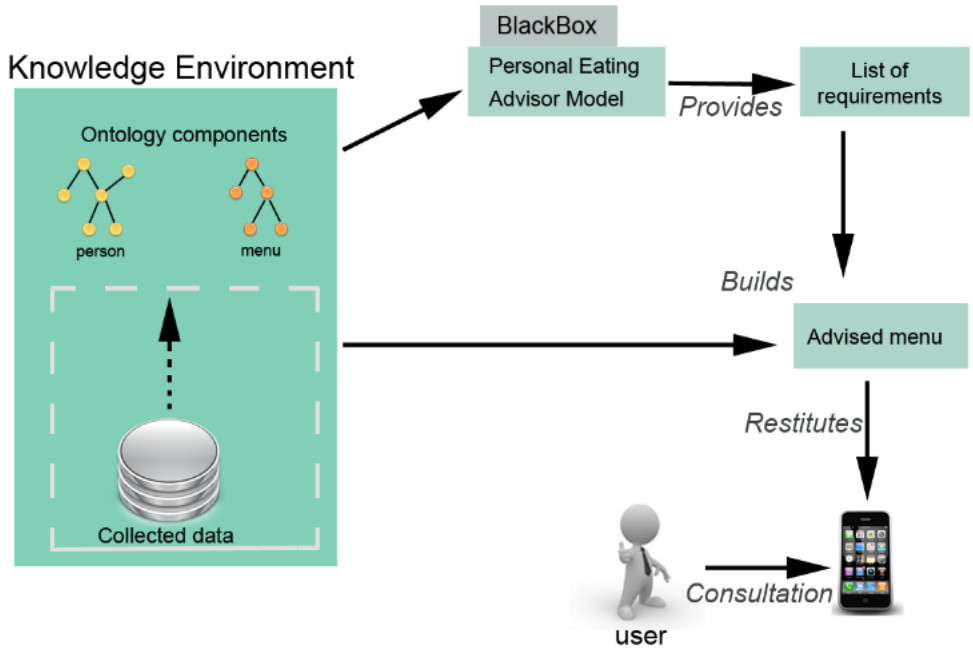


FIGURE 1.2 – Architecture générale du prototype

Trois profils de consommateurs établis avec les chercheurs et une diététicienne ont guidé notre étude. En effet, le domaine est très vaste et il fallait réduire dans un premier temps l'espace de complexité. Par rapport au modèle de conseils et aux données disponibles, le prototype intègre des fonctionnalités restreintes. Il s'agissait cependant

de valider la globalité de l'approche. Les profils définis apportent individuellement une problématique bien identifiée pour laquelle il est attendu que le système apporte des réponses. Le profil principal et prioritaire est celui de ***John***, détaillé ci-après. Le second profil est celui de ***Wim*** un sportif qui désire se préparer pour une compétition et le dernier profil est celui d'***Esther***, une environnementaliste (qui veut manger biologique) qui a une intolérance à la tomate. Vous trouverez en Annexe B, chapitre 1, le détail de nos profils.

John *est un homme de 50 ans et mesure 1 mètre 80. Il fait partie de l'équipe* Information Management*, travaille devant son ordinateur toute la journée et a une activité sédentaire. Il est un consommateur régulier du Restaurant du Futur, puisqu'il y mange quatre fois par semaine. Il n'apporte jamais d'aliment extérieur au restaurant. Il a rarement de repas de travail, mais déjeune souvent avec les autres membres de l'équipe. John n'est pas sportif, habite trop loin de son travail pour venir en vélo, et vient donc en voiture. Son IMC*[7] *est d'environ 30, donc il doit perdre du poids et c'est pourquoi il a un régime alimentaire basses calories particulier. Le problème est que John est gourmand et habitué à choisir des plats sucrés, il prend toujours un dessert pour finir son repas.*

Recommandations diététiques pour John : *dans le cadre d'un projet pour le Restaurant du Futur, John s'est rendu chez une diététicienne qui lui a listé les recommandations suivantes, en concordance avec le projet du restaurant :*

Rec. 1 : le repas est divisé en catégories pour chacunes desquelles un produit doit être choisi ;

Rec. 2 : Soupe, prendre une soupe de type bouillon ou basse calorie ;

Rec. 3 : Pain, prendre maximum deux tranches de pain complet ;

Rec. 4 : Assortiment pour sandwich, prendre maximum deux tranches de fromage :

Rec. 5 : Salade, prendre une salade naturelle ;

Rec. 6 : la somme des calories des produits choisis doit être inférieure à 600 kcal.

Exemple simple de scénario d'utilisation du Smartphone : *John quitte son laboratoire vers 12h30, et se dirige avec les autres membres de l'équipe vers le Restaurant du Futur pour aller déjeuner, son Smartphone en poche. Juste avant d'arriver, John sort son Smartphone et lance l'application* Lunch@RoF*, le premier écran d'accueil souhaite la bienvenue à John sur l'application et lui indique que le menu du jour est disponible. Arrivé au restaurant, John prend un plateau et ses couverts, puis consulte la liste des*

7. IMC : Indice de Masse Corporelle

catégories d'aliments disponibles aujourd'hui, il sait en regardant la liste qu'il pourra trouver des soupes adaptées à son régime (point vert devant la catégorie des soupes). Arrivé devant le buffet des soupes, il se rend compte que toutes lui donnent envie. Il joue le jeu et consulte la liste des soupes sur son Smartphone et voit qu'une des soupes est signalée comme "à éviter" par une marque rouge, mais que l'autre est conseillée (marque verte). Il se sert de la soupe conseillée et juste avant de repartir pour choisir la suite, il ajoute dans le plateau virtuel de son Smartphone la soupe choisie. Il continue ainsi de suite. Avant de quitter le self, il passe devant le buffet des desserts et a envie de se laisser tenter par une crème au chocolat, il consulte le résumé du plateau virtuel et voit qu'il n'a pas dépassé les 600 kcal conseillées. Il ajoute la crème au plateau virtuel et là, une alerte rouge lui indique que le total des calories de son repas va dépasser les 700 kcal. Un peu déçu, John supprime la crème de son plateau virtuel et cherche un dessert alternatif parmi les desserts conseillés, il trouve une mousse de fruits qui convient juste, il voit que cette mousse est située sur le buffet des fruits, il s'y rend tout content, car il adore aussi cette sorte de produits, il la met sur son plateau physique et va payer ses produits à la caisse.

Chapitre 2

Etat de l'art

Pour mettre en place le prototype de diffusion d'informations alimentaires personnalisées pour les consommateurs du Restaurant du Futur, nous avons eu besoin d'un large spectre de notions et de technologies. La démarche est de baser ce système sur des connaissances que nous avons choisies de représenter sous forme d'ontologies et en utilisant les langages de représentation du Web Sémantique. Le système en lui même comprend un système permettant d'exploiter ces connaissances. L'architecture mise en place est distribuée et le moyen de consultation des conseils alimentaires est une plateforme de type ordiphone.

2.1 Ontologies et outils du Web Semantic

Ontologies

Dans le contexte de l'informatique et des sciences de l'information, une ontologie peut être définie comme un ensemble de concepts et de relations permettant de modéliser un domaine de connaissances (Gruber (2009)). Dans concepts sont incluses la signification et les contraintes logiques d'application d'une représentation.

Nous pouvons distinguer plusieurs types d'ontologie, regroupés et commentés par Psyché et al. (2003). Cette classification va dépendre du but de l'ontologie, de l'étendue et de la précision du domaine modélisé :

- les ontologies de représentation des connaissances : formalisation des connaissances ;
- les ontologies supérieures ou de haut niveau : ontologies universelles qui ne dépendent pas de domaine, ni de problème particulier. Elles concernent des concepts très généraux comme le temps, les évènements, l'espace, les actions... Ces concepts doivent être consensuels entre une grande communauté ;

– les ontologies génériques ou méta-ontologies : moins génériques que les précédentes, mais assez pour être réutilisables à travers plusieurs domaines (exemple OBOE [1]) ;
– les ontologies de domaine : ensemble de vocabulaires et de concepts qui décrit un domaine d'application, basé sur la connaissance du domaine où la tâche est réalisée ;
– les ontologies de tâches : pour conceptualiser des tâches (diagnostic, planification...) spécifiques dans les systèmes ;
– les ontologies d'application : les plus spécifiques, domaine restreint, destinées à l'exécution d'une tâche. Un concept correspond souvent à un rôle joué par une entité dans le système.

Les ontologies peuvent exploiter le fonctionnement du Web de plusieurs manières, Berners-Lee et al. (2001) :

– pour améliorer la pertinence des recherches, par concept,
– pour associer à une page des structures de connaissance et à des règles d'inférence.

Web Sémantique

L'approche actuelle du Web Sémantique, ou Web des données, est basée sur l'idée de travailler, au niveau du Web, à partir de concepts plutôt que de documents, qui contiendraient alors des informations, ou métadonnées, formalisées pour être traitées automatiquement par des agents logiciel. Le Web de données est fondé sur une pile de langages et protocoles issus du Web et propres au Web Sémantique (illustration figure 2.1), conçus pour être compris sémantiquement et utilisables par les programmes. Le W3C [2] supervise le développement des standards, propre aux Web Sémantique, comme RDF [3], RDFS [4], OWL [5] ou SPARQL [6] que nous allons décrire.

RDF : pour *Resource Description Framework*, Manola & Miller (2004), un langage pour représenter des informations à propos de ressources sur le Web. RDF est destiné aux situations où ces informations ont besoin d'être traitées par des applications au lieu d'être seulement affichées pour les personnes. Les informations peuvent être échangées entre les applications sans perte de sens. RDF est fondé sur l'idée d'identifier les choses en utilisant des URI [7], et en décrivant les ressources en fonction de propriétés et de

1. OBOE : OBOE : Extensible Observation Ontology
2. W3C : World Wide Web Consortium
3. RDF : Resource Description Framework
4. RDFS : Resource Description Framework Schema
5. OWL : Web Ontology Language
6. SPARQL : SPARQL Protocol and RDF Query Language
7. URI : *Uniform Resource Identifier*, identificateurs de ressource Web

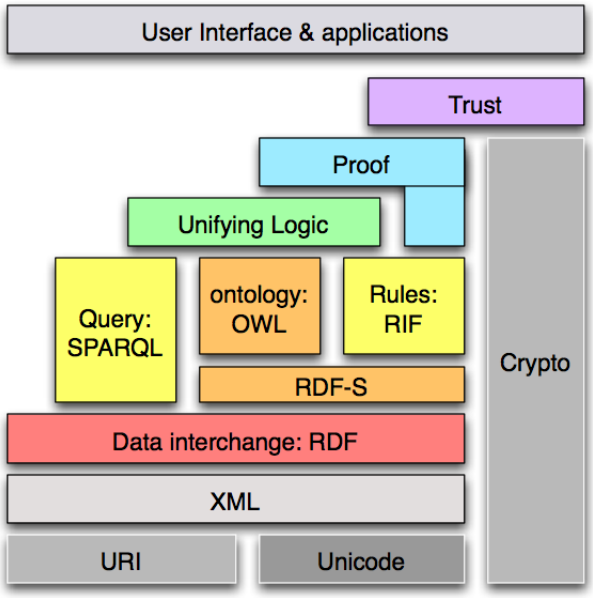


FIGURE 2.1 – Pyramide des technologies du Web Sémantique

valeurs de propriétés. De ce fait, on pourra traduire des informations simples sur les ressources sous forme de graphe, où les nœuds représentent des ressources et les arcs les propriétés. RDF emploie une terminologie particulière, illustrée par la figure 2.2, pour indiquer les diverses parties des déclarations :

- le sujet concerne ce qui est décrit,
- le prédicat correspond à la propriété,
- l'objet est la valeur de la propriété.

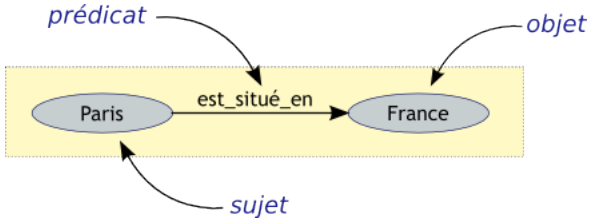


FIGURE 2.2 – Représentation graphique d'un triplet RDF

La sérialisation des déclarations, ou triplets, peut avoir différents formats, Segaran et al. (2009) : notation *N-Triples* [8] (syntaxe verbeuse), *N3* [9] (syntaxe condensée) et *RDF/XML* [10]

8. http://www.w3.org/2001/sw/RDFCore/ntriples/
9. http ://www.w3.org/DesignIssues/Notation3
10. http ://www.w3.org/TR/REC-rdf-syntax/

(syntaxe officielle).

RDFS : pour *RDF Schema*, permet de décrire les ressources du vocabulaire RDF. Il défini la notion de classe et de propriété. Il permet de définir des hiérarchies de classes et de propriétés.

OWL : pour *Web Ontology Language*, Mcguinness & van Harmelen (2004) et Segaran et al. (2009), est un langage RDF pour la définition de classes et de propriétés (RDFS), mais aussi pour permettre plus de raisonnements et des inférences basées sur les relations. Ce langage est divisé en trois sous-langages dont la complexité et l'expressivité augmentent. *OWL-Lite*, le plus simple, OWL-DL [11], basé sur les logiques de description et OWL-Full, le plus expressif mais dont la calculabilité n'est pas garantie.

SPARQL : pour *SPARQL Protocol and RDF Query Language*, Prud'hommeaux & Seaborne (2008), SPARQL est un langage d'interrogation standard pour les graphes RDF. Il est capable de rechercher des motifs de graphe (graph patterns) obligatoires et optionnels ainsi que leurs conjonctions et leurs disjonctions. SPARQL gère également le test extensible des valeurs et la contrainte des interrogations par un graphe RDF source. Les résultats des interrogations SPARQL peuvent être des ensembles de résultats ou des graphes RDF. Ce langage d'interrogation ne permet pas, comme le fait SQL [12] pour les bases de données relationnelles, d'insertion, de modification ou de suppression de triples.

Outils d'exploitations

Sesame est un logiciel libre. Ce "framework" Java, développé au départ par la société hollandaise Aduna dans le cadre d'un projet européen, évolue actuellement grâce à un projet communautaire et hébergé par OpenRDF. Sesame [13] va fournir les fonctionnalités nécessaires au stockage de triplets RDF ainsi que des librairies d'interrogation et de manipulation de ces triplets. Sesame va également fournir le raisonneur RDFS, subsumption de classes et de propriétés par exemple et OWL DLP avec l'extension OWLIM. Il est facilement extensible grâce au développement de la communauté, citons par exemple des adapteurs Jena pour faciliter le passage entre les deux API [14], Elmo qui fournit une API, basée sur les annotations Java, permettant un mapping entre structure métier et ontologie.

11. OWL-DL : Ontology Web Language Description Logics)
12. SQL : Structured Query Language
13. http://www.openrdf.org/
14. API : Application Programming Interface

Jena est aussi un outil libre et similaire à Sesame. Jena [15] a été développé au départ par la société Hewlett-Packard Labs Semantic Web Programme. L'API est plus complète car elle inclut nativement le support de OWL. De plus, ce "framework" Java fournit également plusieurs raisonneurs internes et permet l'utilisation de Pellet, raisonneur OWL-DL.

2.2 Web Service

Ils vont permettre de mettre en place l'architecture distribuée pour l'utilisation d'une plateforme mobile. Il existe deux grands types de Web Service : les architectures REST [16] et l'utilisation des standards liés à XML [17].

Les Web Services RESTful sont basés sur un type d'architectures fondées sur les concepts de ressources et d'agents et développées dans la thèse de Roy Fielding, Fielding (2000). Le principe est qu'un composant logiciel puisse lire ou modifier une ressource en utilisant une représentation (XML ou JSON [18] par exemple) de cette ressource. Comme dans le domaine du Web Sémantique, l'URI a un rôle important et doit suffire pour identifier une ressource. Les services basés sur rest utilise le protocole HTTP, défini par le norme RFC2616, voir Fielding et al. (1999). Ce protocole fournit les opérations nécessaires à la manipulation des ressources : GET, POST, PUT, DELETE, etc.. L'inconvénient de REST est que le client doit conserver localement les données, puis que c'est un service sans état, ce qui peut poser des problèmes de charge.

Les Web Services basés sur les technologies du W3C reposent sur un ensemble de protocoles et de standards utilisés au départ pour l'échange de données entre environnements hétérogènes. Les trois éléments principaux sont SOAP (Simple Object Access Protocol) pour l'échange des messages pour l'appel et réponse de méthodes entre les deux environnements, WSDL (Web Service Description Language) pour la description des messages échangés (type de données, opérations...) et enfin les annuaires UDDI (Universal Description Discovery and Integration) pour le référencement et la découverte des Services Web. L'inconvénient dû à la multitude des technologies est la performance qui est moins importante.

15. http://openjena.org/
16. REST : Representational State Transfer
17. XML : eXtensible Markup Language
18. JSON : JavaScript Object Notation

2.3 La plateforme *Smartphone*

Un smartphone ou ordiphone peut être différencié d'un téléphone portable ou d'un assistant électronique de poche (ADP) [19] par ses fonctionnalités. Un smartphone intègre actuellement un client de messagerie, un navigateur web, des fonctionnalités GPS [20], de synchronisation avec des outils d'ordinateur de bureau type "organiseur" comme agenda, carnet d'adresse, dictaphone (Charlesworth (2009)).

Le système d'exploitation Windows Phone 7 a été choisi pour notre prototype. Ce système sera disponible en octobre 2010 mais dès à présent Microsoft met à disposition un ensemble d'outils de développement gratuits, à savoir :

- un émulateur Windows Phone 7,
- l'EDI [21] Visual Studio 2010 Express pour Windows Phone CTP [22],
- la plateforme de développement Silverlight pour Windows Phone CTP,
- le framework XNA 4.0 Game Studio CTP (destiné au développement de jeux, il n'a pas été utilisé pour notre projet).

Le fait que soit livré un ensemble d'outils complets et faciles à installer a contribué au choix de la plate-forme Windows Phone 7. De plus, nous avons choisi Silverlight car il permet de migrer facilement le prototype Smartphone vers un site web. Enfin, nous avons particulièrement tenu compte des compétences des membres l'équipe dans les techniques liées à Silverlight et C# et par-là même de la future maintenance et évolution du produit.

L'émulateur *Windows Phone 7* est une application [23] de type "ordinateur de bureau" permettant de simuler le système d'exploitation Windows Phone 7. Il fournit un environnement de virtualisation pour déployer, déboguer et tester les applications développées. Il a été conçu pour fournir la même ergonomie et les mêmes performances qu'un réel ordiphone. De plus, il requiert les mêmes exigences de développement. Toutefois, il ne fournit pas pour l'instant tous les programmes natifs de Windows Phone 7 (e.g. la fonction de téléphonie, de camera ou de GPS). Ceci n'a pas été préjudiciable à cette version du prototype car seules les fonctionnalités de réseau (pour la communication avec le Web Service) et la présence d'une zone de stockage persistante (pour l'enregistrement de la configuration des paramètres utilisateur) ont été utiles.

19. PDA : Personal digital assistant
20. Géolocalisation par satellites ou *Global Positioning System*
21. Environnement de développement intégré
22. CTP : Community Technology Preview indique que les outils sont en pré-version)
23. Windows Phone Emulator : http ://msdn.microsoft.com/en-us/library/ff402563%28v=VS.92%29.aspx

Microsoft Visual Studio 2010 Express pour Windows Phone CTP permet de créer des applications Silverlight pour Windows Phone 7. Il permet de déployer l'application directement sur l'émulateur ou sur un appareil Windows Phone 7.

Silverlight est une implémentation du Framework Microsoft .Net, destinée à créer des applications interactives riches (ou RIA [24] en anglais) sur le Web, multiplateformes et navigateurs Web. Les bibliothèques de classes fournies par Silverlight pour Windows Phone donnent des classes, composants, contrôles et éléments d'interface utilisateur spécialement adaptés aux applications sur ordiphone Windows Phone 7. Ils correspondent à une version améliorée de Sivlerlight 3 classique sans offrir toutes les fonctionnalités de Silverlight 4 (la version actuelle).
Silverlight peut être décrit comme un sous-ensemble de la technologie *Microsoft Windows Presentation Foundation* [25] (WPF) et va notamment donner accès aux fonctionnalités de liaison de données (*data binding* en anglais).

L'infrastructure de liaison de données est importante pour cette technologie et pour le modèle de programmation d'applications Silverlight. Elle permet de lier l'interface utilisateur (e.g. une *ListBox*) à un objet de données (e.g. tableau d'objet). Il gère le flux entre les deux : lorsque l'un des deux change, les modifications sont répercutées automatiquement sur l'autre élément par un moteur de liaison. La figure 2.3 montre les éléments qui entrent en jeu dans la liaison [26].

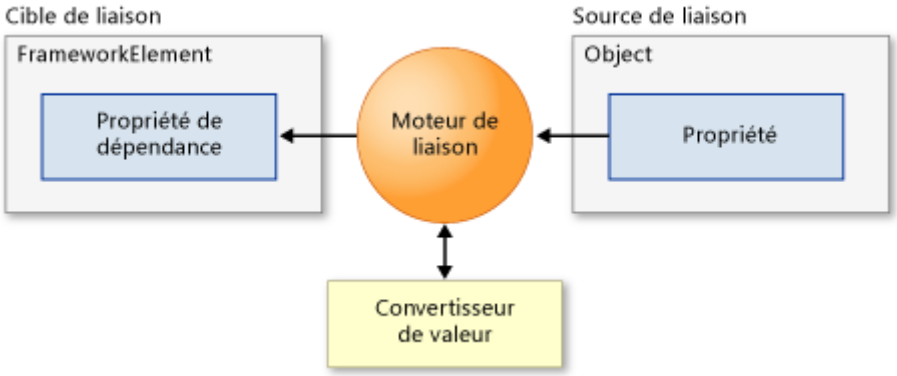


FIGURE 2.3 – Concepts de base d'une liaison de données

24. Rich Internet Application, dont la première définition et description des fonctionnalités sont décrites par Allaire (2002)
25. http://msdn.microsoft.com/fr-fr/library/cc903925%28v=VS.95%29.aspx
26. http://msdn.microsoft.com/fr-fr/library/cc278072%28v=VS.95%29.aspx

XAML *XAML* (eXtensible Application Markup Language) est un langage de balisage basé sur XML (eXtensible Markup Language). Dans Silverlight, le rôle principal de XAML est de décrire l'apparence visuelle de l'interface utilisateur. La logique est définie dans un fichier associé appelé *code-behind* ou code joint. L'avantage de son utilisation est la séparation des rôles des personnes : les concepteurs graphistes et développeurs de l'application.
Le *code-behind* est un objet d'un langage du *Common Language Runtime* (CLR), composant du Framework Microsoft .Net, comme C#. Lorsque l'application est compilée, la page correspondant au code XAML est partielle et complétée par le code joint. Ce code va permettre notamment la gestion de la logique comme la gestion des évènements de l'interface.

Le patron de conception *Modèle-Vue-VueModèle* (MVVM), [27] est un patron de conception architectural basé sur le design pattern Modèle-Vue-Controleur (MVC) bien connu. Ce modèle est utilisé massivement et fait partie des bonnes pratiques de conception pour les applications Silverlight. C'est donc ce patron de conception qui structure l'application ordiphone du projet avec trois types d'objets : des vues écrites en XAML et C#, des classes ModelVue en C# et des objets modèles présentant les données, de deux types C# et XML. La figure 2.4 [28] représente les interactions entre les vues, les données et le modèle de présentation. Ici on voit le rôle de la liaison de données qui va permettre de lier la vue et le modèle de présentation sans qu'il soit nécessaire de faire référence à la vue et donc de coder la mise à jour de la vue et tester les interactions. On va donc avoir une liaison dynamique entre ces deux composants.

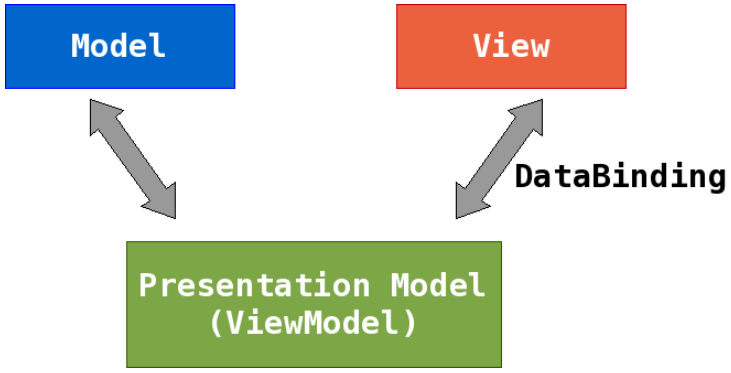


FIGURE 2.4 – Schéma du modèle Modèle-Vue-VueModèle

Dans ce patron de conception, le modèle de présentation va faire le lien entre le modèle (données) et les vues. Dans notre projet, nous avons mis à disposition les données par l'intermédiaire d'un Web Service RESTFull sous forme de données XML, voir

27. MVVM : *Model-View-ViewModel* en anglais
28. Présentation du modèle disponible à l'adresse http://live.visitmix.com/MIX10/Sessions/EX14

section 4.4. L'exploitation de XML avec Silverlight est possible avec l'interface de programmation *LINK to XML* [29].

LINK to XML est proche du modèle DOM [30] car le document XML est chargé en mémoire. Le document peut être interrogé ou modifié. *LINK* [31] est un langage de requêtes proche de SQL [32] ou XPath (non disponible sous Silverlight pour Windows Phone).

29. Complément disponible à l'adresse : http://msdn.microsoft.com/library/bb308960.asp
30. DOM : Document Object Model
31. LINK : Language Integrated Query
32. SQL : Structured Query Language

Chapitre 3

Modèles de connaissances

Introduction

Les modèles de connaissances, au sein du système PEA (*Personal Eating Advisor*), jouent un rôle prépondérant, dans le sens où ils vont permettre une consultation adaptée de l'information et par là même, guider l'interface homme-machine de diffusion. De plus, ils vont être un support à la résolution des problèmes posés et être exploités par le modèle de conseils alimentaires.

Deux champs principaux de connaissances sont impliqués dans le système PEA et sont à la base de la grande majorité des modules ontologiques, ces champs couvrent le Restaurant du Futur et les consommateurs. Les deux champs sont reliés par un bon nombre de passerelles et vont permettre au système de répondre aux problématiques du projet, à savoir, fournir des conseils alimentaires pertinents et personnalisés à un consommateur. A cette fin, plusieurs relations de dépendance ou de réutilisation vont exister entre nos modules ontologiques. Par ailleurs, étant dans un contexte complexe et de recherche, ils sont amenés à s'étendre et à inclure des concepts de domaines complémentaires, comme par exemple, celui de la perception sensorielle ou de la diététique. Ainsi, nous avons conçu nos modèles comme une collection de composants ontologiques mis en relation et réutilisés, afin de prévoir les futures extensions.

Nous présenterons, en premier lieu, le processus de conception et l'organisation générale des composants avec leurs interactions, section 3.1, puis nous détaillerons leur formalisation dans les sections suivantes.

3.1 Processus de conception et interaction entre les composants ontologiques

Processus de développement

Nos ontologies ont été développées en se basant sur la méthodologie "Ontology Development 101" décrite par Noy & Mcguinness (2001). Les première étapes de détermination du domaine et de la portée des ontologies, ainsi que, la liste des termes importants sont joints en annexe B, chapitres 2 et 3.

Pour les définir, nous nous sommes, en premier, appuyés sur l'existant, c'est-à-dire les données disponibles : les produits, l'identification des employés, les enquêtes, etc.. Par la suite, certains pans de connaissances ont été mis en exergue en interrogeant un expert du comportement des consommateurs et une diététicienne. Enfin, nous avons pris en compte l'expertise de l'institut au travers d'articles scientifiques publiés par l'institut dans le domaine.

Nous avons formalisé nos modèles de connaissances avec le langage du Web Sémantique OWL, standards du W3C. En effet, il apporte plus d'expressivité que les langages RDF et RDFS, sur lesquels, il s'appuie. Et plus précisément, nous avons utilisé les caractéristiques de OWL DL pour pouvoir raisonner dans un espace qui reste calculable et décidable.

Chacun des composants a été défini à l'aide de l'éditeur TopBraid. Dans la suite de ce document, le format de restitution des définitions des ontologies peut varier : pseudo notation UML fournie par l'éditeur, code Turtle [1] ou format XML/RDF.

Le cas du modèle de conseils alimentaires personnalisés

Le modèle de conseils est développé par les chercheurs en sciences du comportement des consommateurs. Son rôle est de fournir des conseils pertinents. Durant mon stage, nous l'avons considéré comme une *boîte noire*. Afin que le système puisse tenir compte des recommandations du modèle, nous avons créé une ontologie spécifique, destinée à l'expression des recommandations. La figure 3.1 schématise le principe de l'intégration allant du modèle au système. Les modules ontologiques des consommateurs, des menus et produits sont en entrée. Pour la sortie, nous utilisons l'**ontologie des recommandations** ou *Advise* dans notre projet.

Stratégie de combinaison des composants ontologiques

Un composant, en génie logiciel, est une unité de composition, avec un contexte de dépendances explicites. Szyperski complète la définition dans Broy et al. (1998) : un

1. Turtle - Terse RDF Triple Language, submission standard W3C référencée à l'adresse http://www.w3.org/TeamSubmission/turtle. Chaque triplet est exprimé textuellement de manière compacte.

FIGURE 3.1 – Interaction entre le modèle de conseils et les ontologies

composant logiciel peut être déployé indépendamment et peut être sujet de composition d'un tiers.

De la même façon, nos composants ontologiques ont des propriétés similaires. Ils sont réutilisables, composés d'autres composants, composables et possède une autonomie relative. La figure 3.2 représente, à l'aide de la notation UML, les trois composants majeurs, à savoir :

- une ontologie du contexte du restaurant avec la définition des menus et de l'environnement matériel,
- une ontologie pour décrire les produits,
- une ontologie pour définir les consommateurs.

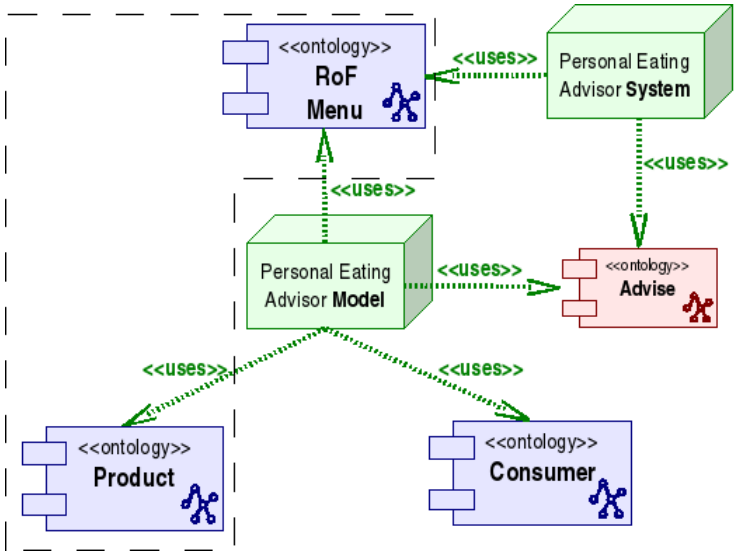


FIGURE 3.2 – Représentation des composants ontologiques majeurs, au travers d'un diagramme de composants UML

Nous avons aussi représenté l'ontologie des recommandations, *Advise*, car elle joue en effet un rôle particulier dans notre système.

Une zone en pointillés a été ajoutée afin de représenter le lien fort qui existe entre l'ontologie du contexte du restaurant et l'ontologie des produits. Ceci nous pousse, souvent tout au long de ce document, à ne considérer qu'une seule ontologie, celle des produits. En pratique cela est dû au fait que nous retrouvons un lien de composition entre les menus et les produits, et que l'ontologie de description de l'environnement (table, chaise, luminosité...) du restaurant reste encore à construire.

Sur ce schéma, les ontologies sont exploitées par deux éléments (noeuds) : le modèle et le système de diffusion. Dans les deux cas, des ontologies différentes sont actives, ou utilisées, par les acteurs : le contexte, les produits, les consommateurs et les recommandations par le modèle de conseils et le contexte/menu et les recommandations par le système de diffusion. Ceci correspond au premier type de modularité entre nos ontologies.

Une autre caractéristique de nos ontologies est la sous-composition et le partage de composants. Ceci est lié à la propriété de réutilisation des ontologies. Par exemple, la figure 3.3 schématise une partie des relations des ontologies *Measure* et *Diet*, que nous avons spécialement définies pour le projet, et dont le but est respectivement de décrire les mesures (e.g. masse, prix, etc.) et les régimes alimentaires des consommateurs.

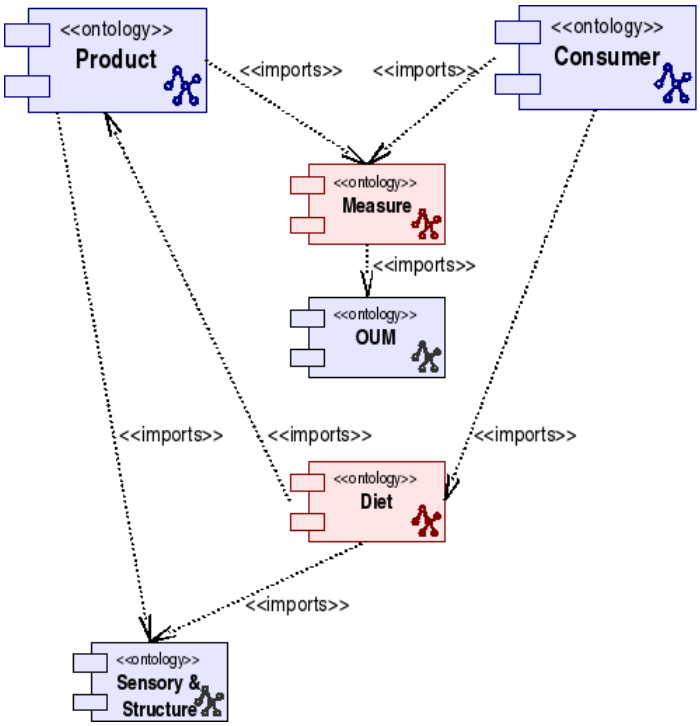


FIGURE 3.3 – Représentation des composants *Measure* et *Diet*, avec la notation UML

L'ontologie locale des mesures va réutiliser l'ontologie OUM (cf. section 3.2). *Measure* est utilisée pour annoter toutes les caractéristiques mesurables de nos modèles. Ainsi, elle est utilisée pour décrire les caractéristiques physiques (taille, poids...) d'un consommateur, ou encore les caractéristiques nutritives d'un produit. Au besoin, nous pouvons connecter le composant des *Measure* aux ontologies, pour ajouter des types de connaissances mesurées. La définition des régimes alimentaires, *Diet* sur la figure 3.3, et des produits sont d'autres exemples d'utilisation de composants "partagés". Ainsi l'ontologie *Sensory&Strucure* est utilisée par le consommateur pour décrire ses préférences gustatives, tandis que dans le cadre de l'ontologie des produits on décrit les goûts.

Dans les sections suivantes, nous détaillerons la modélisation des modules ontologiques des produits du restaurant, section 3.2, des consommateurs, section 3.3 et enfin la définition des recommandations du modèle de conseils, section 3.4.

3.2 Ontologie des produits du Restaurant du Futur

Le but de l'ontologie du Restaurant du Futur (RoF) est de modéliser le contexte d'un restaurant : matériel (disposition des buffets, des meubles, des plats...) et le contenu (le menu, catégories...). Une ontologie des produits est aussi nécessaire pour décrire les produits impliqués dans les menus. Ces modèles sont spécifiques à notre prototype et utilisables pour la restitution du menu à l'utilisateur. Pour cette modélisation, nous nous sommes appuyés sur les données acquises concernant les produits et les menus.

Les données disponibles sont de plusieurs types :
- les catégories et les buffets existants au restaurant ;
- la définition des menus par saison de trois mois ;
- les produits : nom, prix, masse, caractéristiques nutritives, végétarien ou non, catégories du restaurant d'appartenance, buffet où ils sont disposés, disponibilité (quotidienne ou variable), etc.

Product est le concept principal de notre modèle. Il correspond à un élément d'un menu, un aliment solide ou liquide simple (e.g. une pomme) ou plus sophistiqué composé d'ingrédients (e.g. tagliatelles au saumon). Certains produits sont cuisinés sur place par le chef, d'autres élaborés au-dehors. La figure 3.4, générée avec l'éditeur TopBraid, correspond à la définition d'un produit et montre ses relations avec les autres concepts du restaurant.

Graphiquement, les propriétés d'objets sont précédées d'un rectangle bleu, les propriétés de types de données d'un rectangle vert. Chaque nom de concept ou de propriété est précédé d'un identificateur d'espace de nommage défini par l'utilisateur, ici,

prod correspond à l'espace de nommage *http ://www.wurvoc.org/rof/product#* et *meas* à *http ://www.wurvoc.org/rof/measurement#*. L'espace de nommage, dans les standards du W3C, correspond à un vocabulaire spécifique. Différents espaces de nommage permettent ainsi de tirer parti de différents vocabulaires et structures dans une ontologie, l'utilisation d'identificateurs permet de rendre la présentation plus lisible et d'éviter toute ambiguïté entre les structures exploitées. Ainsi, l'entité *Product* fait référence à deux vocabulaires qui correspondent à la définition du module ontologique des produits et au composant des mesures, détaillé section 3.2.

L'entité *Product* est caractérisée par plusieurs types de propriétés qui vont porter sur : ses caractéristiques mesurables ou non et sa place dans le contexte du restaurant.

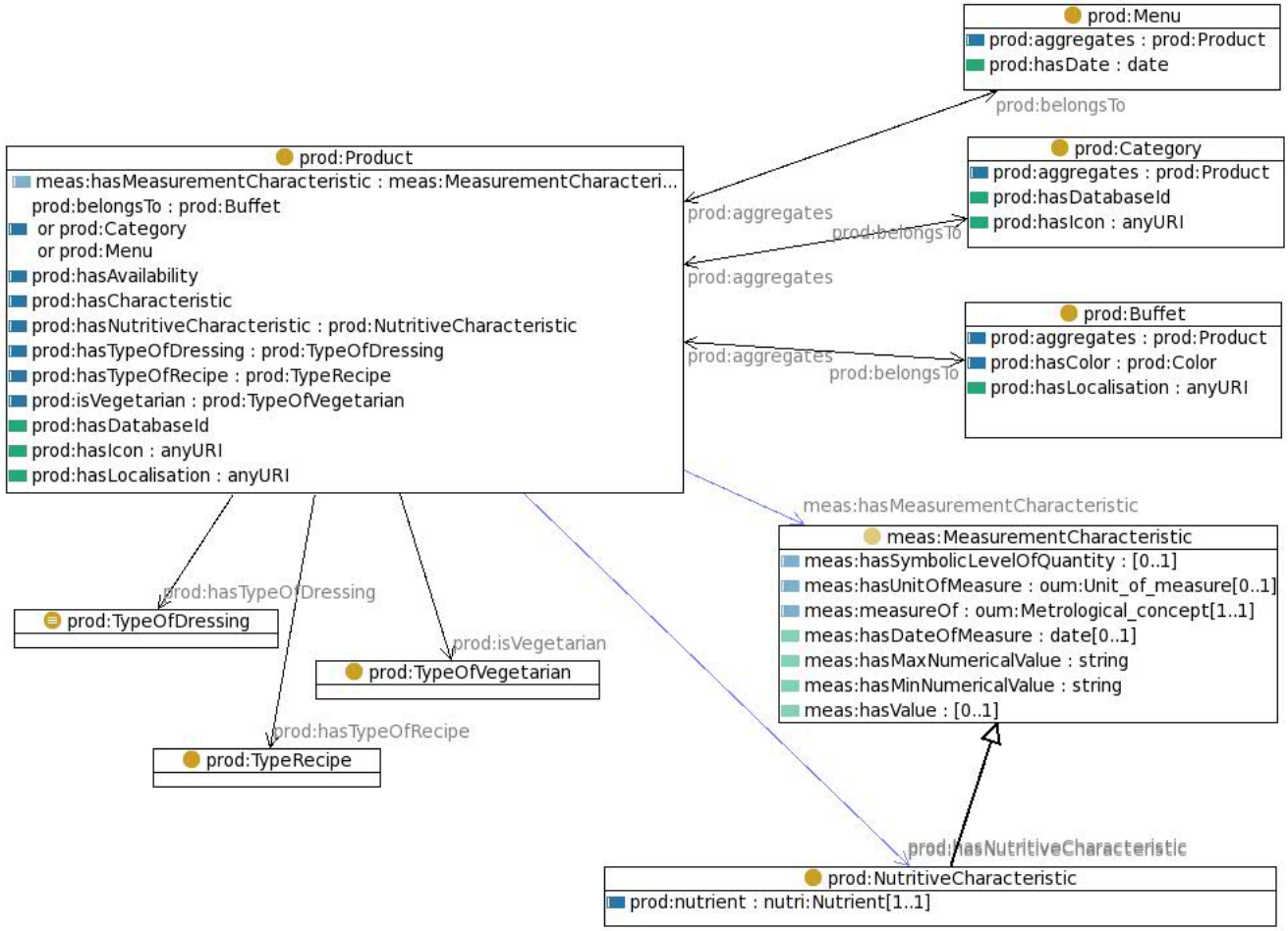


FIGURE 3.4 – Ontologie pour décrire les produits disponibles au RoF

L'illustration d'un produit est donnée par la propriété *hasIcon* de cible un nom du fichier. La gestion de l'affichage de l'image se fera ensuite par l'application. C'est une propriété importante au niveau du domaine d'application, car l'aspect des plats entre en jeu lors de du choix de certains consommateurs. Cette propriété est aussi utilisée par le concept *Category*. Un produit qui n'a pas d'illustration, utilise celle de sa catégorie.

La localisation des produits et des buffets dans le restaurant va permettre de mettre en place, l'aide à l'accès aux produits en évitant le phénomène de tentation : alerte visuelle, tactile ou chemin optimal (cf. section 1). Pour répondre à ce besoin, la propriété *hasLocalisation* de domaine *Product* ou *Category* a été ajoutée. Dans cette version, la localisation est représentée pour l'application par un nom de fichier de carte prédéfinie du restaurant avec un buffet mis en surbrillance (cf. fig. 3.5). Si nous n'avons pas la localisation d'un produit, sa localisation est celle du buffet.

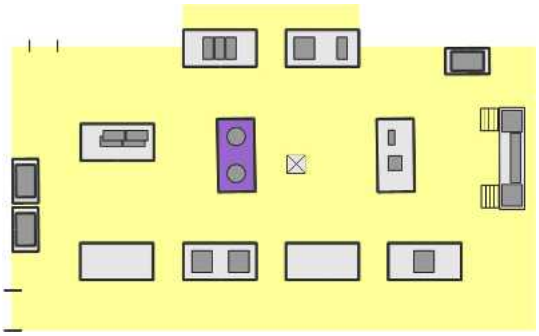


FIGURE 3.5 – Exemple de localisation du buffet des soupes

La représentation de l'agrégation de produits dans des catégories, sur des buffets ou pour les menus est établie au travers des propriétés d'objets *belongsTo* et *aggregates* (cf. figure 3.4). La figure 3.6, générée avec TopBraid, représente leur définition. Ces propriétés sont inverses, la propriété *belongsTo* a pour domaine *Product* et d'image formée par l'union des concepts *Buffet*, *Category* et *Menu* ce qui signifie qu'un produit peut appartenir à des menus, des catégorie ou des buffets. La propriété *aggregates* va permettre d'accéder aux produits en partant de l'élément regroupant : catégories, menu ou même localisation. Cette définition facilite la navigation entre les éléments et l'accès aux produits.

Dans notre modélisation, nous avons favorisé l'utilisation de relations entre nos produits d'autres concepts car cette structuration est souple et va permettre de faire évoluer la catégorisation des produits. Par exemple, les produits laitiers sont actuellement répartis dans deux catégories du restaurant : "beurre et sauce" et "dessert". Or, pour identifier les produits à base de lait de vache (en cas d'intolérance par exemple), nous allons avoir

besoin de nous appuyer sur un autre type de connaissances et utiliser l'ontologie des produits laitier, déjà existante.

L'idée des propriété *belongsTo* et *aggregates* est de représenter le concept de composition. Odell (1998) définit la composition comme un mécanisme servant à former un objet entier en utilisant d'autres objets comme parties. La représentation de la composition est un sujet en développement dans le cadre du W3C, voir Rector & Welty (2005) sur la représentation de la composition. En effet, elle n'est pas native au formalisme comme peut l'être la spécialisation et nécessite un modélisation manuelle.

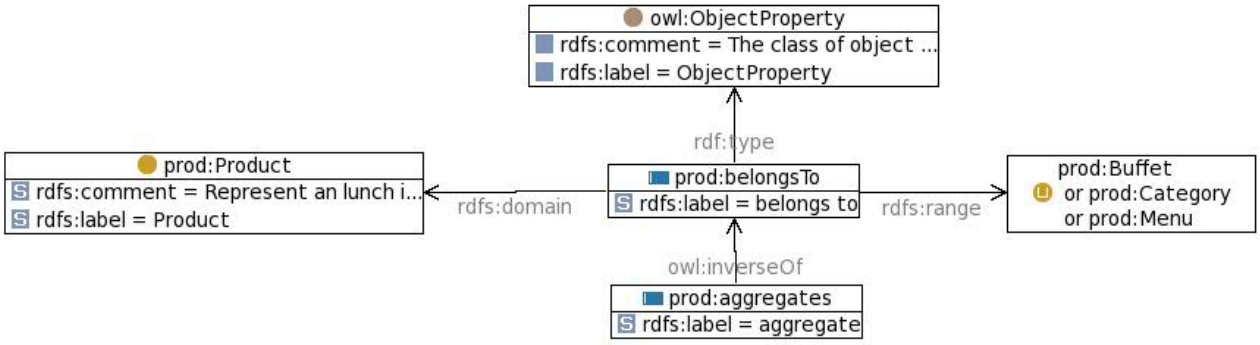


FIGURE 3.6 – Définition de la propriété *belongsTo*

Les caractéristiques mesurables des produits sont définies à l'aide du vocabulaire d'un composant que nous avons défini *Measure*. Nous avons choisi de modéliser ce concept par une relation n-aire entre :

- une variable ou *Metrological_concept* ;
- une unité *Unit_of_measure* ;
- une valeur qui peut être numérique, booléenne ou symbolique.

Le but est de pouvoir représenter, par exemple, qu'"une portion de tiramisu a une masse de 85 grammes".

La difficulté ici a été de représenter une relation n-aire, entre plusieurs éléments, alors qu'en général les propriétés dans le Web Sémantique sont des relations binaires entre deux instances. La solution choisie est celle qui est aussi explicitée dans le document du W3C de Noy & Rector (2006), c'est-à-dire l'introduction d'une classe pour une relation, ici le concept de *MeasurementCharacteristic*. Le listing 3.1, représente la définition, en Turtle, du concept que nous avons appelé *MeasurementCharacteristic*.

Une propriété *hasValue* avec comme sous-classe *hasNumericalValue* va nous permettre de renseigner la valeur de la mesure. Deux autres propriétés d'objet vont venir compléter l'information de la valeur à savoir *measureOf* et *hasUnitOfMeasure* pour la variable mesurée et l'unité de mesure. Ces trois propriétés sont fonctionnelles (cf.

annexe E, listings E.1 et E.2), c'est-à-dire que pour une instance de *MeasurementCharacteristic* ces propriétés auront une unique instance.

Listing 3.1 – Code Turtle de la définition de *MeasurementCharacteristic*

```
meas:MeasurementCharacteristic
      a       owl:Class ;
      rdfs:label "Measurement characteristic"@en ;
      rdfs:subClassOf owl:Thing ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:maxCardinality "1"^^xsd:nonNegativeInteger ;
                owl:onProperty meas:hasDateOfMeasure
              ] ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:maxCardinality "1"^^xsd:nonNegativeInteger ;
                owl:onProperty meas:hasValue
              ] ;
      rdfs:subClassOf
              [ a       owl:Class ;
                owl:intersectionOf ([ a       owl:Restriction ;
                            owl:allValuesFrom oum:Unit_of_measure ;
                            owl:onProperty meas:hasUnitOfMeasure
                          ] [ a       owl:Restriction ;
                            owl:allValuesFrom oum:Metrological_concept ;
                            owl:onProperty meas:measureOf
                          ])
              ] ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:maxCardinality "1"^^xsd:nonNegativeInteger ;
                owl:onProperty meas:hasUnitOfMeasure
              ] ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:cardinality "1"^^xsd:nonNegativeInteger ;
                owl:onProperty meas:measureOf
              ] .
```

L'instanciation des propriétés *measureOf* et *hasUnitOfMeasure* est rendue obligatoire par l'indication de cardinalités avec le langage OWL (restriction *cardinality*). De plus, l'intersection des restrictions sur l'unité et la variable mesurées va permettre, par inférence d'un raisonneur, de définir par restriction les éléments de la classe *MeasurementCharacteristic* directement par les propriétés de l'instance. La figure 3.7 montre l'organisation des concepts, notamment la réutilisation de l'ontologie OUM [2] (Ontology of Units of Measure) développée par mon équipe d'accueil et décrite par Rijgersberg et al. (2010). L'utilisation de cette ontologie permet de déléguer le lien et la sémantique de l'unité et de la variable et leur lien à l'ontologie importée. Seule la notion de devise, que nous avons ajoutée, manquait pour décrire les prix des produits. Enfin, l'utilisation de OUM permettra, dans le futur, de lier les autres applications basées sur OUM et notamment d'accéder facilement au Web Service de conversion d'unités [3]. La figure C.1

2. Disponible à l'adresse http://www.wurvoc.org/vocabularies/om-1.6/
3. Disponible à l'adresse http://www.wurvoc.org/services/oum.jsp

de l'annexe E représente les instanciations de l'exemple précédent grâce au concept de *MeasurementCharacteristic*.

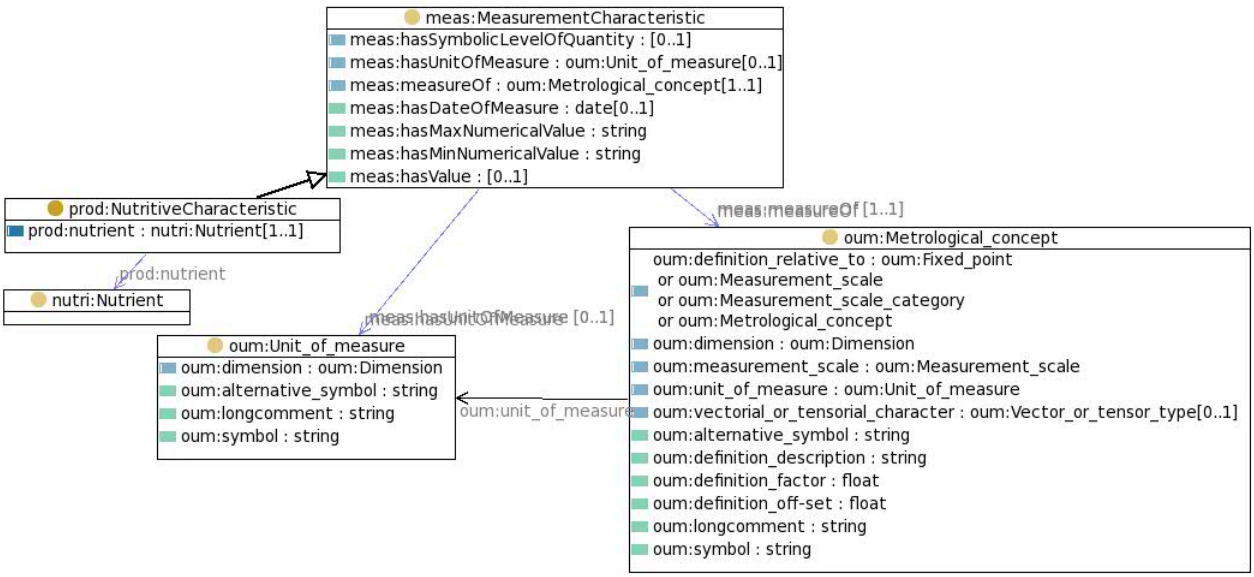


FIGURE 3.7 – Concepts des caractéristiques mesurables

La sous-classe *NutritiveCharacteristic* de *MeasurementCharacteristic* possède également la propriété *nutrient* qui va nous permettre de modéliser les caractéristiques nutritives des produits. Le principe est le même que précédemment, sauf que la variable est dépendante du nutriment. Par exemple, la dose de sucre contenue dans la pomme est en fait une masse de sucre, l'annexe E, figure C.2, montre sa représentation. Pour gérer la sémantique du nutriment, nous avons réutilisé l'ontologie développée par mon équipe d'accueil, qui est en fait une taxonomie des nutriments (cf. fig. 3.8).

L'ontologie développée ici, pour décrire les produits et leurs rôles dans le restaurant, est faite en fonction des connaissances déjà disponibles dans le projet FOVEA et au RoF. Elle reste suffisante tant que le modèle n'est pas stabilisé mais devra évoluer assez rapidement dans un premier temps avec l'introduction de la notion d'ingrédients, puis dans un second temps avec l'annotation des produits par rapport à leurs origines et leurs sensations et structures gustatives.

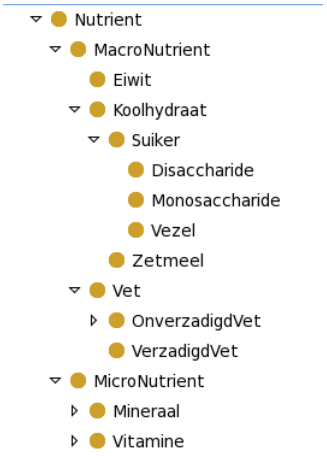


FIGURE 3.8 – Extrait de la taxonomie des nutriments

3.3 Ontologies pour les préférences des consommateurs

L'objectif du projet est de proposer des conseils personnalisés aux consommateurs et de les aider à faire les bons choix pour une alimentation adaptée. Ces conseils ou recommandations sont personnalisées en fonction des caractéristiques du consommateur et de ses préférences. Dans le domaine applicatif de mon projet, une des originalités est de tenir compte de ces préférences alimentaires. Ces connaissances alimenteront le modèle de conseils. Les recommandations produites porteront sur le contenu d'un menu mais aussi sur la forme du message.

La définition du modèle s'est faite en deux phases : la première assez générale, grâce à la littérature scientifique du domaine et la seconde plus "appliquée". Cette dernière est basée sur nos trois profils et sur les données disponibles, à savoir :

- identité de la personne, accessible grâce à l'administration de l'institut,
- résultat d'une enquête (mode de vie, interaction avec la nourriture, etc.),
- goûts alimentaires au restaurant (historique des précédents repas).

Ainsi, plusieurs composants ontologiques ont été produits. Ils pourront être assemblés et complétés en fonction des besoins du modèle. La version intégrée dans la mise en oeuvre du prototype est plus légère et ne contient que l'extension de l'entité de la personne et les résultats de l'enquête.

La définition d'un consommateur est actuellement liée à son identité au sein de l'établissement. Les données seront issues directement des bases de données de l'institut. Par exemple, la figure 3.9 montre une instanciation du concept *Consumer*. John est défini par son identité : nom, prénom, année de naissance, numéro de carte d'employé. Ce sont des propriétés de type de données. Il est à noter que la classe Consommateur

est définie avec des restrictions sur la cardinalité exprimant le caractère obligatoire de ces propriétés. Notamment le *WURCardNumber*, identifiant de l'employé à l'institut, est sous forme XML de type :

```
<owl:Restriction>
        <owl:cardinality rdf:datatype="http://www.w3.org/2001/XMLSchema#
            nonNegativeInteger">1</owl:cardinality>
        <owl:onProperty rdf:resource="http://www.wurvoc.org/rof/consumer#
            hasWURCardNumber"/>
</owl:Restriction>
```

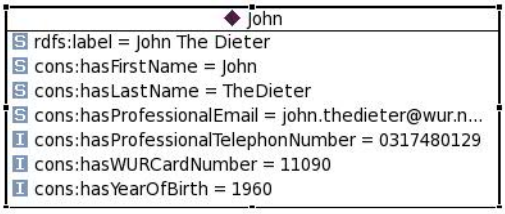


FIGURE 3.9 – Description de John

Le choix alimentaire des consommateurs est un comportement complexe et son étude est pluridisciplinaire (science du comportement, biologie et physiologie, économie, mercatique, sociologie et psychologie). Köster (2009) reprend, dans son article, l'approche de Jos Mojet, experte en science des consommateurs à l'Institut de recherche de Wageningen, pour exprimer la complexité du domaine et les différents aspects. Elle décrit en 2001 (cf. annexe C) les différents facteurs qui influencent le comportement du choix des consommateurs concernant l'alimentation. Nous avons adapté cette proposition pour certains de nos composants ontologiques, puisqu'une partie correspond bien au domaine étudié dans le cadre des recherches menées au Restaurant du Futur.

Les aspects biologiques et physiologiques correspondent aux caractéristiques de la personne : physiques, génétiques, etc.. Au niveau modélisation, nous avons différencié les caractéristiques mesurables (taille, poids par exemple) des autres caractéristiques (sexe de la personne). De plus, la propriété d'objet ***hasPhysicalProperty***, dérivée par la suite en fonction des différentes caractéristiques physiques de l'utilisateur permet d'une part, de pouvoir naviguer directement jusqu'aux caractéristiques physiques du consommateur, d'autre part, de contrôler la signature de la relation (domaine et image) de chacune des sous-propriétés. Par exemple, listing 3.2, la propriété *hasWeight* possède pour domaine un objet de type *Consumer* et d'image une instance de concept *Weight*. Enfin, ce choix de modélisation nous permettra de récupérer l'ensemble des propriétés physiques par l'utilisation directe de la propriété mère, ce qui est utile dans le cas de génération d'interface.

Listing 3.2 – Code Turtle de la définition de *hasWeight*

```
cons:hasWeight
      a       owl:ObjectProperty ;
      rdfs:domain cons:Consumer ;
      rdfs:label "has weight"^^xsd:string ;
      rdfs:range cons:Weight ;
      rdfs:subPropertyOf cons:hasPhysicalProperty .
```

Pour définir les **mesures** nous avons réutilisé la définition de *MeasurementCharacteristic*, décrite dans le listing 3.1, qui va nous permettre de lier une variable, une unité et une valeur. Pour l'instant, les principales variables physiques sont définies, à savoir : taille, poids et IMC. Nous les avons définies en plusieurs étapes. La définition en code Turtle du listing 3.3 du concept de poids montre que nous avons créé, dans un premier temps, une sous-classe du concept de mesure ; puis, nous avons affiné sa définition par restriction à la mesure du poids *oum :height* et en rendant la date de mesure obligatoire. L'objectif ici était de faciliter l'acquisition des données puisque seule la déclaration d'une mesure de type *Height* est suffisante pour le raisonneur pour avoir le type de variable mesurée. De plus, il permet un contrôle plus fin des informations nécessaires comme dans le cas de la date et du poids.

Listing 3.3 – Code Turtle de la définition de *Weight*

```
cons:Weight
      a       owl:Class ;
      rdfs:label "Weight@en"^^xsd:string ;
      rdfs:subClassOf meas:MeasurementCharacteristic ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:hasValue oum:weight ;
                owl:onProperty meas:measureOf
              ] ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:cardinality "1"^^xsd:nonNegativeInteger ;
                owl:onProperty meas:hasDateOfMeasure
              ] ;
      meas:measureOf oum:weight .
```

Les propriétés physiques de la personne dans notre cas sont les plus pertinentes, car elles suffisent aux diététiciens pour donner des premières recommandations. De ce fait, nous avons fait évoluer le concept de *Consumer* pour y ajouter des cardinalités plus précises (cf. annexe F).

Les facteurs psychologiques et liés au contexte entrent aussi en compte lors du choix des aliments. Il s'agit, par exemple, de la motivation, des habitudes alimentaires ou de vie. Pour modéliser ces derniers, je me suis appuyée sur l'enquête faite auprès des consommateurs du restaurant. Dans l'état actuel du projet, c'est l'une des sources importantes pour les chercheurs, puisque l'acquisition doit se faire le moins intrusive possible pour ne pas influencer le comportement des consommateurs. L'enquête est donnée

en annexe A. Elle couvre également le domaine des **facteurs contextuels**, comme le cadre de vie. Au niveau modèle, nous avons défini des concepts, comme *Household* qui représente le foyer de la personne (cf. annexe D, fig. D.1), lequel va être caractérisé grâce à des propriétés par le nombre d'enfants, le statut marital, la localisation ainsi que par les habitudes d'achats du foyer. Il s'agit du concept de *ShoppingHabit* qui permet de connaître le type d'aliments achetés (produits du commerce équitable par exemple), le type de magasin et la fréquence d'achat.
Sur le même principe que précédemment, nous avons modélisé le concept de l'activité sportive d'une personne (cf. annexe D, fig. D.2) avec des propriétés d'objet pour le type de sport, la fréquence et le niveau.
La modélisation de ces notions est faite sur le même principe : un élément central (*ShoppingHabit* ou *SportiveActivity*) lié par des propriétés (*inShopOfType* ou *concernedSport*) à des concepts de type (par exemple : les magasins d'alimentation *FoodProductShopType* ou les types de sports disponibles *Sport*). L'avantage ici est que, si le besoin apparaît de définir plus finement les concepts des types de magasins ou les différents sports, nous pourrons faire évoluer facilement notre ontologie de base en créant ou important de nouvelles connaissances produites par d'autres organismes comme la FAO par exemple pour les magasins ou le lien vers une ontologie des sports.

La notion de régime alimentaire est un concept que nous avons commencé à représenter ici. D'autres ontologies et structures de connaissances existent mais restent bien souvent trop orientées vers le domaine de la santé par rapport au domaine du projet et trop complexes par rapport à nos besoins en matière de prototypage. L'idée ici était plutôt d'essayer de représenter le type de régime alimentaire souhaité par un consommateur, ainsi que **ses objectifs**. Pour débuter, nous nous sommes appuyés sur nos trois profils. Ce composant ontologique minimal fait intervenir plusieurs types de facteurs décrits et illustrés par Köster (2009). La figure 3.10 illustre cette modélisation. Un consommateur peut avoir un ou plusieurs régimes alimentaires : *DietPreference*. Ce concept intègre la représentation d'une relation n-aire de propriétés d'objets. *adviser* représente celui qui est à l'origine de cette préférence. En effet, à la base, le consommateur lui-même veut suivre un régime particulier, par exemple "manger biologique". Le système/diététicien aura également un objectif pour le consommateur, par exemple un régime avec peu d'hydrates de carbone. Le modèle devra ensuite gérer ces souhaits et ou ces conflits et produire des conseils. La propriété *reason* et son image *ReasonType* correspondent à la notion de facteur de **motivation** et de facteur **contextuel** socio-culturel (la religion ou le désir de s'alimenter à partir des produits issus du commerce équitable par exemple). Ce concept est associé au concept de niveau d'importance défini pour l'instant simplement par ses instances :

Listing 3.4 – Code Turtle de la définition d'*ImportanceLevel*

```
:ImportanceLevel
```

```
    a       owl:Class ;
    rdfs:subClassOf owl:Thing ;
    owl:oneOf (:AbsoluteImportance :HighImportance :LowImportance :MediumImportance)
        .
```

Un régime alimentaire préférentiel est aussi défini par un type de régime : *DietType*. Ce concept peut nous permettre de définir des régimes et est schématisé figure 3.11. Cette figure illustre qu'il est possible d'indiquer l'importance donnée au prix ou à l'environnement dans ce type de régime. De plus, la propriété *involved*, définie listing 3.5, est spécialisée en sous-propriété *avoided*, *favored*, *needed* et *prohibited*. Ces propriétés vont permettre de définir les implications entre le régime et :

- un nutriment (de l'ontologie sur les nutriments), par exemple le régime "low_calory_diet" *avoided energy* ;
- un produit (du restaurant), par exemple le régime "without_tiramisu_diet" *prohibited tiramisu* ;
- un allergène (de l'ontologie *Allergens*[4])), par exemple le régime "peanut_intolerant_diet" *prohibited Peanut* ;
- une perception (de l'ontologie *Sensory_and_structure*[5])), par exemple le régime "chocolateFlavour_favored_diet" *favored ChocolateFlavour*.

Listing 3.5 – Code Turtle de la définition d'*involved*

```
:involved
    a       owl:ObjectProperty ;
    rdfs:comment "involved product or charcateristic of a product"@en ;
    rdfs:domain :DietType ;
    rdfs:label "involved"@en ;
    rdfs:range
            [ a       owl:Class ;
              owl:unionOf (:Characteristic allergens:Allergen nutri:Nutrient :Product
                  )
            ] .
```

Ces composants ontologiques sur les consommateurs et les régimes alimentaires vont donc nous permettre de formaliser :

- l'identification du consommateur ;
- ses caractéristiques biologiques et physiologiques ;
- les aspects psychologiques liés à ses habitudes de vie ;
- le régime alimentaire souhaité / nécessaire à la personne.

3.4 Ontologies de recommandations

Dans notre projet, le *Personal Eating Advisor Model*, ou modèle de conseils, fournira des "recommandations" (ou *Requirements*) offrant les fonctionnalités suivantes :

4. Ontologie développée par l'équipe IM-WUR : http ://www.wurvoc.org/allergens
5. Ontologie développée par l'équipe IM-WUR : http ://www.wurvoc.org/sensory_and_structure

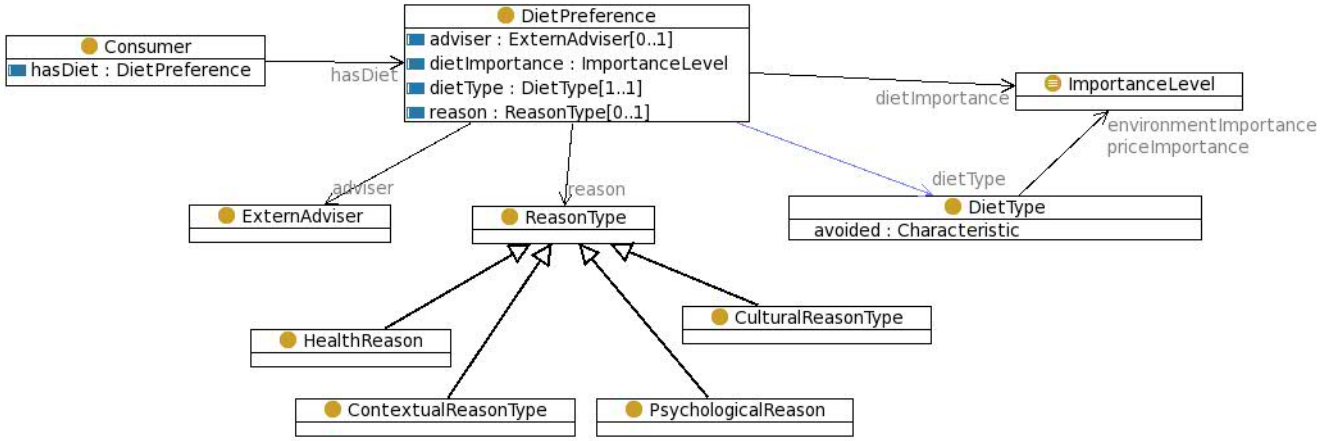


FIGURE 3.10 – Représentation de la préférence d'un régime alimentaire

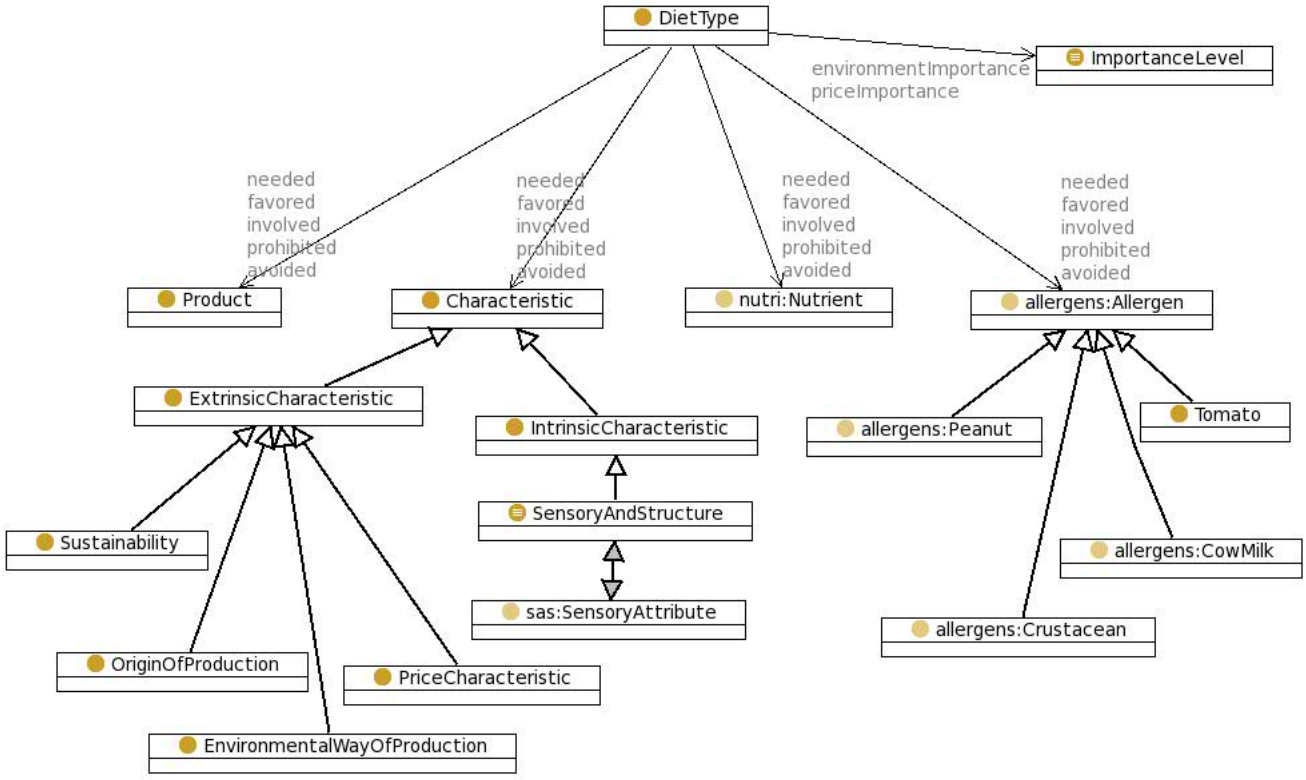


FIGURE 3.11 – Représentation des types de régimes alimentaires

3.4 Ontologies de recommandations

– conseiller ou éviter certains éléments du menu (produit, buffet ou catégorie),
– cibler les caractéristiques pertinentes pour le consommateur (personnalisation du contenu du message),
– guider l'affichage des informations et conseils au niveau de l'interface utilisateur(*Smartphone*).

Ne connaissant pas encore la forme de la sortie des "recommandations", j'ai choisi de modéliser un composant ontologique permettant de formaliser les "recommandations". Cette modélisation pourra ainsi utiliser les mêmes concepts (vocabulaire contrôlé) définis précédemment pour les produits et les consommateurs. Les "recommandations" ainsi modélisées seront utilisées pour exploiter les connaissances disponibles sur le menu du jour et les produits et ainsi fournir les fonctionnalités attendues du modèles de conseils.

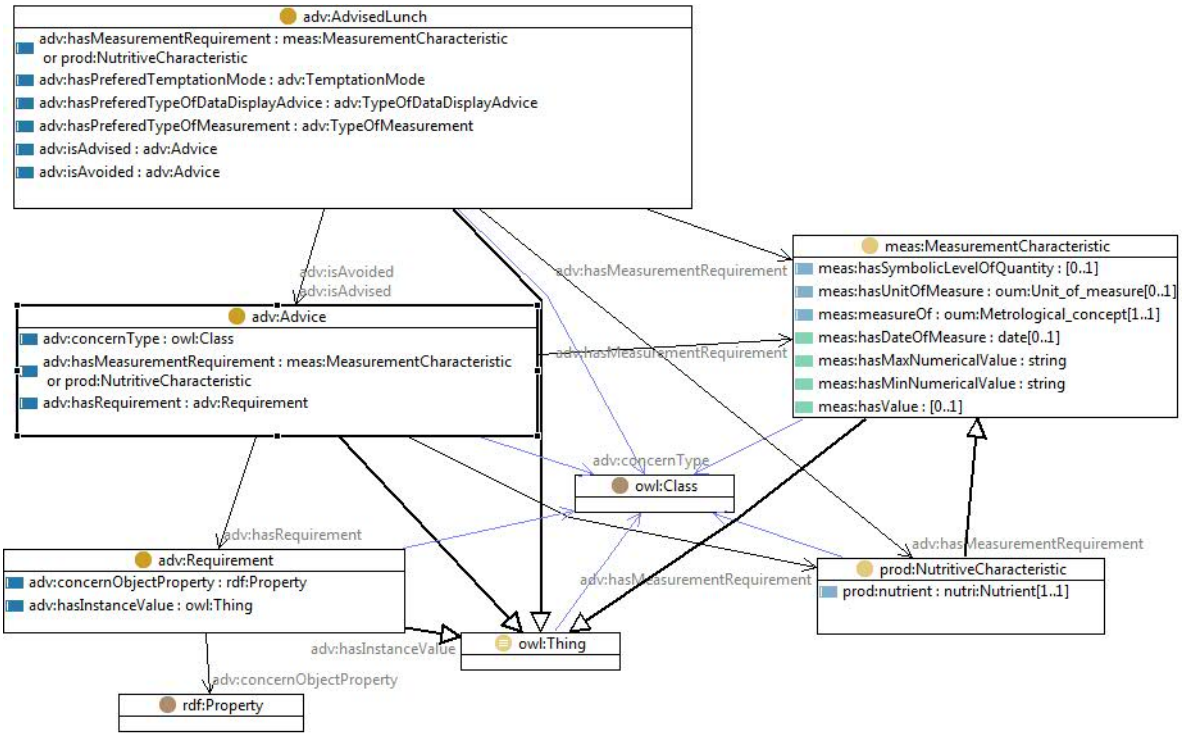


FIGURE 3.12 – Ontologie pour décrire les conditions pour le repas

Pour la modélisation, je me suis appuyée sur les profils définis avec les chercheurs et dégager les conditions types pouvant être fournies par le modèle de conseils. De plus, nous avons montré le scénario de John à une diététicienne qui a nous a donné le type de conseils diététiques qu'elle aurait pu prescrire.

Techniquement, l'idée est qu'en sortie du modèle de conseils, une instance du concept *AdvisedLunch* soit créée. Cette instance est composée de :

Une collection de conseils ou d'*Advice* La figure 3.12 montre que chacun des conseils peut être *isAdvised* ou *isAvoid*. Ils peuvent concerner (relation *concernType*) un concept de l'ontologie des produits, par exemple *Product* ou *Category*. Cette relation a pour cible un objet OWL de type *Class*. Ce conseil peut respecter des conditions qui concernent des caractéristiques de l'élément : *Requirement* ou qui concernent des variables mesurées : *MeasurementCharacteristic* ou *NutritiveCharacteristic*. La capture d'écran de l'éditeur TopBraid, figure 3.13, illustre la représentation de l'expression "John doit prendre des produits qui appartiennent à la catégorie des salades et qui ont la propriété d'avoir une présentation naturelle [6]".

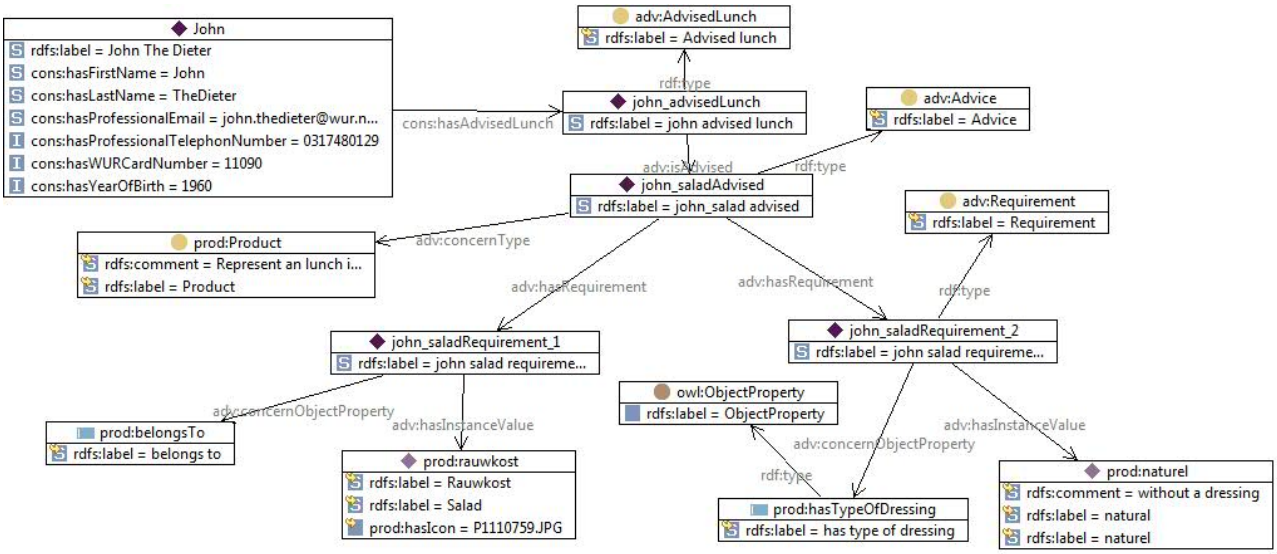


FIGURE 3.13 – Capture d'écran TopBraid, conseil pour John de prendre une salade fraîche

Une caractéristique pertinente pour le consommateur Il s'agit ici de définir une *MeasurementCharacteristic* ou une *NutritiveCharacteristic* préférentielle pour John. Cette variable a pour portée le repas complet et peut être associée à des conditions *MeasurementCharacteristic* ou *NutritiveCharacteristic* qui vont permettre d'exprimer le fait par exemple que John ne devrait pas ingérer plus de 600 kcal durant son repas (cf. figure 3.14). Par la suite, nous verrons que cette caractéristique est affichée en priorité et est associée à un traitement particulier dans l'application Smartphone.

6. Par opposition aux salades sous film disponibles au restaurant

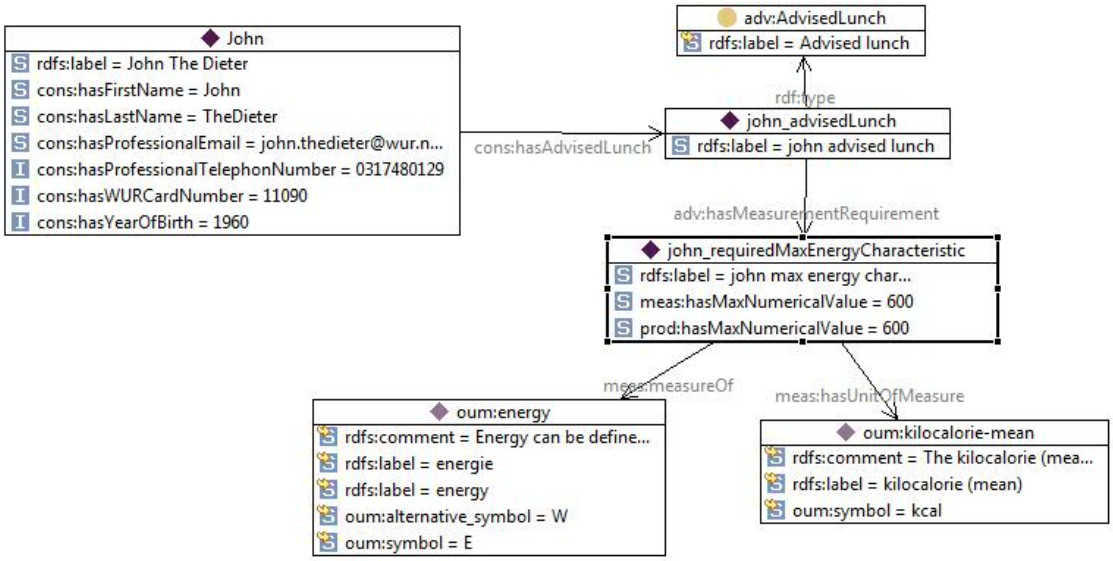


FIGURE 3.14 – Recommandation globale d'un repas de 600 kcal maximum pour John (TopBraid)

La notion de tentation durant le choix des produits est associée au concept *TemptationMode*. Il possède la propriété *owl :oneOf* contenant trois instances *fullTemptation*, *moderateTemptation* et *freeTemptation*.

Le mode d'affichage donné par le concept *TypeOfDataDisplayAdvice* va avoir des conséquences pour l'application Smartphone. En ce qui concerne notre version, seule le mode *rawData* existe et est implémenté dans notre système.

Conclusion

Nous avons modélisé plusieurs composants ontologiques réutilisables, composables d'ontologies locales (e.g. *Measure*) ou externes (e.g. OUM) et combinables en fonction des besoins afin d'avoir un ensemble de connaissances évolutives. Ces composants ontologiques ont été modélisés à partir des données disponibles, par discussion avec les experts du domaine et de l'expertise de l'institut de recherche à travers la littérature produites dans le domaine.

Trois principales structures ont étés produites et permettent de formaliser les connaissances suivantes :

– **le contexte du restaurant :** le menu, les catégories, les buffets avec les aliments ou produits disponibles accompagnés de leurs caractéristiques (nutritive, prix, etc.) ;

- **le consommateur :** identité, caractéristiques physiques, éducation, habitude alimentaire, activité sportive, etc. ainsi que le régime alimentaire souhaité ou nécessaire ;
- **les recommandations du modèle de conseils :** collection de conseils permettant d'annoter les éléments du contexte du restaurant comme "conseillés" ou "à éviter", définition de caractéristiques pertinentes, le type de gestion de la tentation et le mode d'affichage conseillé.

La suite de ce document est consacrée à la mise en oeuvre opérationnelle de nos ontologies à travers un environnement de connaissances. Son exploitation par la première version de notre prototype de diffusion de conseils alimentaires personnalisés est consultable depuis une plateforme mobile type *Smartphone*.

Chapitre 4

Mise en œuvre au travers d'une architecture logicielle

Introduction

Des modèles de connaissances appropriés ont été définis (voir chapitre 3), afin de satisfaire les exigences fonctionnelles du système PEA (Personal Eating Advisor). Ces exigences portent sur l'apport de conseils alimentaires personnalisés et adaptés, aux consommateurs du Restaurant du Futur. A cet effet, les modèles de connaissances construits, se consacrent à la représentation des aliments et des menus du jour proposés aux consommateurs par le restaurant, ainsi qu'à la représentation du profil du consommateur et aux recommandations qui peuvent lui être faites en terme d'alimentation.

Il s'agit maintenant de proposer une mise en œuvre de ces modèles, de mettre en place un prototype du système de diffusion de conseils alimentaires personnalisés, et ainsi de valider notre approche, basée sur de la connaissance et sur les technologies du Web sémantique. Le prototype à mettre en place, doit répondre à plusieurs contraintes imposées par le système PEA, et qui concernent la personnalisation du menu et du message diffusé, les *feedbacks* ou réactions du système et enfin la mobilité. Les solutions apportées pour la prise en charge de ces contraintes sont détaillées ci-dessous :

La personnalisation du menu du jour nous oblige à considérer chaque consommateur de manière différente. L'idée retenue, est d'ajouter des annotations aux éléments du menu afin d'orienter ces éléments du menu vers des profils de consommateurs. Ainsi par exemple, une annotation qui a pour propriété le *statut* et comme domaine de valeurs ("à éviter", *avoid* en anglais et "conseillé", *advised* en anglais), a été définie. La détermination du statut s'appuie sur les recommandations émises par le modèle de conseils (cf. sections 3.1 et 3.4).

La personnalisation du message diffusé permet de déterminer les informations qui vont s'avérer pertinentes à afficher, ainsi que leur format d'affichage. Pour cette première version du prototype, nous nous sommes limités à une seule caractéristique pertinente mesurable. La pertinence sera déterminée par la suite par le modèle de conseils. Toutefois, dans le cadre de notre prototype, la caractéristique considérée est paramétrable et manipulée à l'aide de l'ontologie des recommandations. De la même manière, le format donné au message, sera, dans le futur, défini par le biais du modèle de conseils, mais demeure, pour cette version, limité à l'affichage des données de base (variable, valeur et unité).

Les réactions du système au choix des utilisateurs vont se traduire, pour notre prototype, par l'implémentation de la notion de plateau virtuel. Cette fonctionnalité permettra d'avoir un retour global sur l'ensemble des produits choisis. L'utilisateur aura ainsi la possibilité de simuler son repas et d'ajuster ses choix.

La mobilité est nécessaire pour que l'utilisateur puisse avoir l'information à disposition alors qu'il est en train de faire son choix. Cet argument a joué en faveur du choix d'une plate-forme de consultation ordiphone, ou *Smartphone* en anglais.

La première section de ce chapitre est consacrée à la présentation générale de l'architecture logicielle mise en place, et du processus de personnalisation. La section suivante traite de la mise en oeuvre opérationnelle d'un système à base de connaissances (SBC), section 4.2. Puis les sections 4.3, 4.4 et 4.5 décriront, respectivement, la mise en oeuvre de la couche métier, la mise à disposition des ressources et l'application de consultation sur *Smartphone*.

4.1 Architecture du prototype *PEA* et démarche de personnalisation

4.1.1 Architecture

Le prototype mis en place, figure 4.1, est constitué de plusieurs éléments : un système à base de connaissances, un module métier, un Web Service, et une application sur *Smartphone*.

Système à base de connaissances (SBC) : son rôle est de permettre le stockage et l'exploitation de nos connaissances. Il contient les composants ontologiques développés (cf. section 3), une base de faits contenant les profils des consommateurs, ainsi que les menus, les produits et leurs caractéristiques.

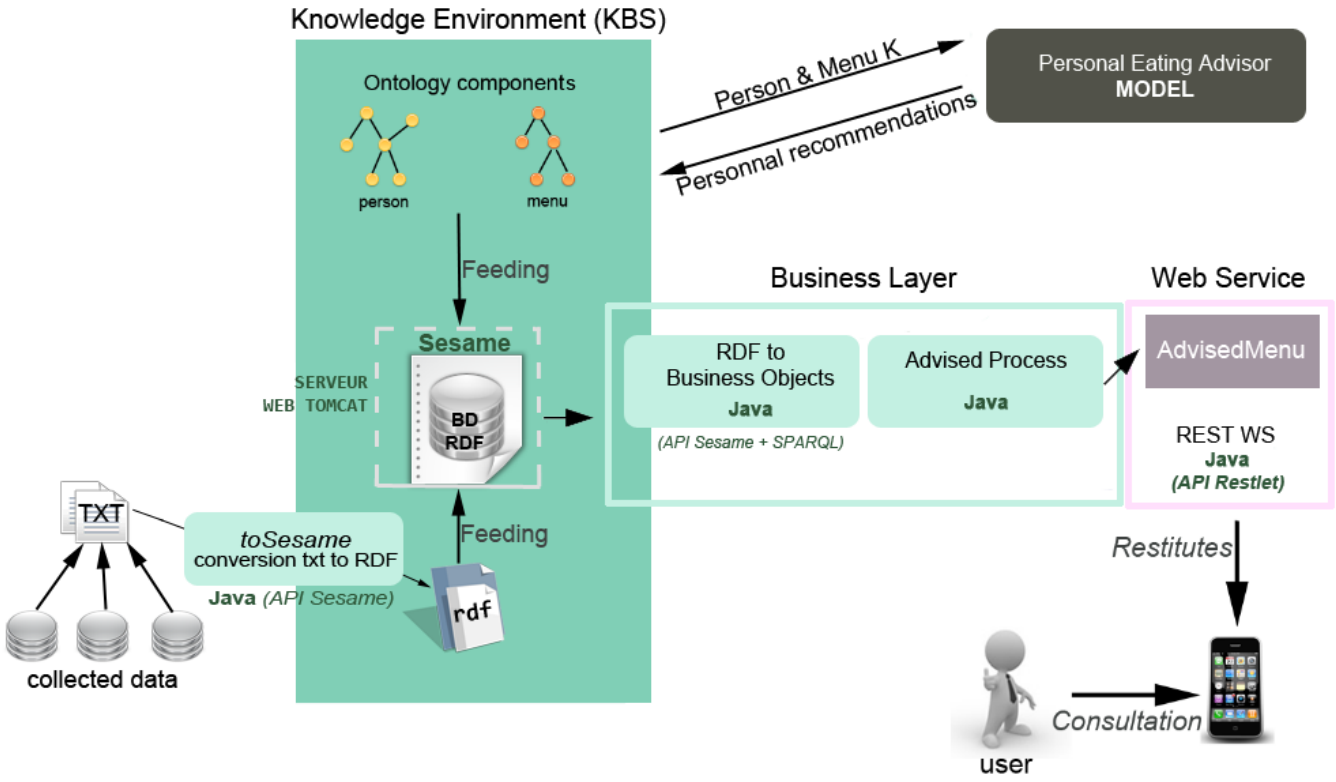


FIGURE 4.1 – Architecture générale du prototype

Module d'acquisition : son rôle est de convertir les données acquises sous format texte, en faits. Ce module est un module annexe, qui permet d'alimenter la base de faits à partir d'informations émanant du restaurant

Modèle de conseils : son rôle va être d'interroger le SBC afin de consulter les connaissances sur le consommateur et le menu du jour, puis de produire les recommandations alimentaires personnalisées sur les produits qui vont influer la constitution du menu et l'affichage des informations associées. L'implémentation de ce modèle n'entre pas dans le cadre de mon stage.

La couche métier : est constituée de deux modules. Le premier module a pour but de convertir nos connaissances en objets métiers, manipulables par les procédures de création de menu. Le deuxième module a pour objectif de constituer à partir du menu du jour un menu "métier" personnalisé pour un consommateur. Trois grandes fonctionnalités sont nécessaires :

- le marquage d'éléments du menu ou du contexte du restaurant (produit, catégorie ou buffet) comme étant "conseillé" ou "à éviter" ;
- la gestion de la caractéristique principale ;
- la gestion de **la tentation** par suppression des produits considérés comme à éviter ;
- la gestion au niveau des classes métier du plateau virtuel de l'utilisateur.

Web Service : permet de mettre à disposition les informations sur le menu du jour fourni par le restaurant, comprenant également les produits figurant dans le menu et leurs caractéristiques. Pour un consommateur identifié, son menu "personnalisé" est également restitué au travers du Web Service.

L'application *Smartphone* : est une application qui facilite la consultation du menu et gère la manière dont sont affichées les informations sur l'interface face aux recommandations émises par le système.

4.1.2 Démarche de personnalisation

Nous avons séparé dans notre prototype, les connaissances, gérées par le SBC, les traitements métiers développés sous forme de modules logiciels et l'interface de consultation (l'ordiphone). La démarche choisie, commence par fournir aux utilisateurs du système, un menu de base qui correspond plus précisémemt à celui du jour de la consultation. Ce menu consultable n'est donc pas immédiatement personnalisé, les données sont au format texte et la caractéristique pertinente est choisie par défaut pour le système, à savoir, la quantité de calorie d'un produit (exprimée en kilocalorie). L'idée est que lorsqu'un utilisateur "connu", pour lequel il existe un profil dans notre base de connaissances, se connecte, le système :

1. crée un menu métier, identique au menu général par copie,

2. extrait de la base de connaissances les recommandations associées à l'utilisateur,
3. applique chacune des règles métiers : annotation (*advised* ou *avoid*) des éléments et gestion de la tentation,
4. prépare le message à diffuser en fonction de la caractéristique pertinente issue des recommandations,
5. met en place, pour l'utilisateur, un objet métier correspondant à son plateau virtuel, concept de *Tray* dans notre projet, paramétré grâce aux recommandations (par exemple l'ingestion de 600 kcal par repas).

Nous avons choisi de modifier un objet métier plutôt que directement des connaissances car certaines règles de personnalisation peuvent, par exemple, nécessiter des calculs.

Nous avons également choisi de gérer, autant que possible, les traitements métier sur le serveur et non pas sur l'application de consultation. Par exemple, le contenu du plateau virtuel est géré au niveau de l'application principale et non pas sur le *Smartphone*. De cette manière, le déploiement de l'application, qui peut être distribué sur plusieurs plateformes, va s'avérer facilité. Enfin la plate-forme de consultation aura tout de même la charge du mode d'affichage, la disposition des éléments sur l'interface, car ces derniers éléments demeurent très dépendants de la configuration des machines clientes.

4.2 Mise en œuvre opérationnelle d'un système à base de connaissances

L'objectif principal d'un système à base de connaissances est d'exploiter des connaissances d'un domaine, afin de permettre la résolution de problèmes pouvant être complexes dans le champ de ce domaine. Le Ber et al. (2006) précise cette définition en ajoutant que ces connaissances doivent être de plus rendues disponibles et stockées à cet effet. La mise en œuvre s'adosse alors à une architecture comprenant : une base de connaissances relative au domaine, une base de faits ou données caractérisant le problème, et enfin un moteur d'inférence dont le rôle est de manipuler les bases pour conduire un raisonnement menant à la résolution d'un problème.

L'architecture proposée (figure 4.1) pour le prototype s'appuie sur le "framework" sémantique ***Sesame***, voir section 2.1. Sesame va fournir les fonctionnalités nécessaires au stockage de triplets RDF ainsi que des librairies d'interrogation et de manipulation de ces triplets. Sesame va également fournir le raisonneur RDFS et OWL DLP avec l'extension OWLIM. Le choix de Sesame dans ce travail, est en accord avec la large utilisation qui est faite de cet outil au sein de mon équipe d'accueil. Certains membres de l'équipe participent, d'ailleurs, activement à ses évolutions et enrichissements et ap-

partiennent à la communauté qui supporte ce projet de développement.

La mise en oeuvre du prototype s'articule autour de trois éléments d'importance qui sont abordés dans le détail dans cette section, à savoir, un serveur de stockage RDF, un jeu de données venant alimenter le référentiel de connaissances et un module logiciel d'exploitation de la base RDF adossé à l'API *Sesame* et au langage SPARQL.

4.2.1 Vocabulaire partagé

Afin de pouvoir exploiter dans l'application les concepts présents dans nos ontologies, nous avons défini un paquetage java structurant le vocabulaire de nos ontologies. Pour chaque composant ontologique utile, nous avons défini une classe qui est caractérisé par des membres statiques comme l'espace de nom, le préfixe et les concepts, propriétés et instances remarquables, qui nous sont d'utilité. L'annexe H montre la représentation UML partielle de ce paquetage.

4.2.2 Serveur de stockage de données RDF

Notre environnement de connaissances a pour élément central un serveur de base de données RDF afin de donner plus de flexibilité au système. Il s'agit aussi d'en faciliter l'accès en mode distribué. Différents modules, notamment d'acquisition et de consultation de données vont pouvoir se partager les accès avec le serveur de triplets. Dans ce sens, nous avons mis en place un moteur de servlets Tomcat, et nous avons défini des servlets permettant la connexion avec la base de données construite sous *Sesame*. L'architecture retenue, nous donne également accès à une interface d'administration de la base, *OpenRDF Workbench*, qui s'avère très pratique en terme de fonctionnalités : interface d'interrogation, gestion et d'exploration du dépôt, etc..

4.2.3 L'acquisition de données RDF et la constitution du jeu de tests

Les graphes RDF relatifs aux personnes ont été créés manuellement avec l'outil TopBraid. Nous avons instancié nos composants ontologiques des préférences des personnes et des recommandations avec les scénarios de John, Wim et Esther.

Le graphe RDF du contexte et des produits du RoF est produit automatiquement, au travers d'un composant logiciel défini au sein de notre application (appelé *toSesame*, sur le schéma figure 4.1). Ce module défini en java permet l'instanciation du contexte du restaurant, des catégories et des buffets. Il permet également d'instancier, à partir de fichiers textes dans un format propriétaire que nous avons nous même défini, les menus, par saison de trois mois et les produits disponibles avec toutes leurs caractéristiques, et

en particulier les caractéristiques nutritives. Cette deuxième phase est cependant semi-automatique, elle nécessite en effet une phase de validation et de consolidation, avec par exemple l'ajout d'un fichier d'illustration pour ce qui concerne les produits.

L'implémentation du composant logiciel Java de conversion est basée sur l'API *Sesame*. La première étape est de créer un référentiel RDF, puis d'y ajouter les triplets créés et enfin de l'exporter. La construction des triplets est basée sur la lecture des fichiers au format propriétaire. Un problème rencontré porte sur "l'éclatement" de l'information, ce qui nous oblige à incrémenter dynamiquement le graphe des triplets, au regard des données déjà instanciées. Un exemple en est l'ajout du prix à un produit instancié au préalable, listings 4.1 et 4.2. Il s'agit, d'abord, d'identifier le produit dans le graphe RDF à l'aide de son identifiant (*iddb*) de base de données (cf. listing 4.1), Puis, listing 4.2, d'instancier une mesure du prix avec sa valeur et son unité et enfin de mettre en relation cette instance avec celle du produit.

Listing 4.1 – Recherche de l'instance de produit

```
StringBuilder query = new StringBuilder();
query.append(" SELECT ?prod ");
query.append(" WHERE { ");
query.append("    ?prod a <" + PROD.PRODUCT + "> . ");
query.append("    ?prod <" + PROD.HAS_DATABASEID +"> \"" + iddb + "\" . ");
query.append(" } ");
TupleQueryResult result = con.prepareTupleQuery(QueryLanguage.SPARQL, query.toString())
    .evaluate();
```

Listing 4.2 – Ajout du prix à un produit

```
if(idProd != null) { // we found the product
  if(!tokens[1].trim().isEmpty()  && !tokens[1].trim().equals("0") ){
    // create a RDF object
    URI idMeasure = factory.createURI(PROD.NAMESPACE, "price_"+idDBProduct);
    // create an instance of type MeasurementCharacteristic
    con.add(idMeasure, RDF.TYPE, MEAS.MEASUREMENT_CHARACTERISTIC);
    // add price to product with the hasMeasurementCharacteristic predicat
    con.add(idProd, MEAS.HAS_MEASUREMENT_CHARACTERISTIC, idMeasure);
    // add the value of price
    con.add(idMeasure, MEAS.HAS_NUMERICAL_VALUE, factory.createLiteral(tokens[1].trim()
        , XMLSchema.STRING));
    // add variable
    con.add(idMeasure, MEAS.MEASURE_OF, PROD.COST);
    // add unit
    con.add(idMeasure, MEAS.HAS_UNIT_OF_MEASURE, PROD.EURO);;
  }
}
```

Notre environnement va donc contenir nos composants ontologiques ainsi que les données sous forme de graphes RDF provenant :

- de trois profils de consommateurs et leurs recommandations,
- du contexte du restaurant : catégories, buffets et menu sur trois mois,

– de la liste des produits disponibles avec leurs caractéristiques.

L'exploitation de la base RDF se fait grâce à l'API *Sesame* et au langage de requêtes *SPARQL.*

4.2.4 Des modèles de connaissances aux objets métiers : exploitation de SPARQL

Le module présenté, ici, est assujetti à l'exploitation de la base de connaissances afin d'en créer des objets métiers. Notre démarche est de construire, au préalable, un menu, vu comme un objet métier, qui sera consultable par tous les utilisateurs. Par la suite, ce menu est personnalisé par la couche métier de notre système en fonction des recommandations du modèle de conseils. La figure 4.2 correspond au diagramme de classes du module java développé.

Nous observons tout d'abord que, seul un sous-ensemble de concepts des composants ontologiques est représenté. En effet, nous avons besoin, sous forme d'objet métier, du concept de consommateur, des concepts consultables par l'utilisateur, à savoir, le contexte et les produits disponibles. De plus, certains composants ontologiques, non nécessairement consultables, servent à la sélection de produit. Enfin, certain concepts comme ceux des caractéristiques de produits sont regroupés. En effet, les caractéristiques des produits représentées de manière hiérarchique dans l'ontologie (cf. section 3.2), ont été "aplaties vers le haut", c'est-à-dire que nous avons choisi de ne construire qu'une seule classe métier *Characteristic*. Nous n'avons pas besoin d'exploiter la spécialisation des concepts pour nos traitements et c'est ce qui explique ce choix qui vise la simplicité de représentation.

Lien entre instances de concepts et d'objets métiers

Les objets métiers et les instances dans l'ontologie sont liés au travers de l'URI de l'instance RDF. Cette information est un identifiant unique dans les technologies développées par le W3C. La classe abstraite ***RoFElement***, cf. firgure 4.2, va fournir aux classes spécialisées un élément instance de concept de base. En effet, ils seront tous définis par les attributs *id* pour l'URI de l'instance, *name* pour l'argument de l'instance ou instance sans espace de nommage et par un *label* attribué en fonction de la langue choisie.

Schéma de réutilisation et de paramétrage par spécialisation

Le schéma de conception des classes principales correspond à un schéma de réutilisation en programmation orientée objets. Il est basé, ici, sur la spécialisation et la redéfinition de méthodes.

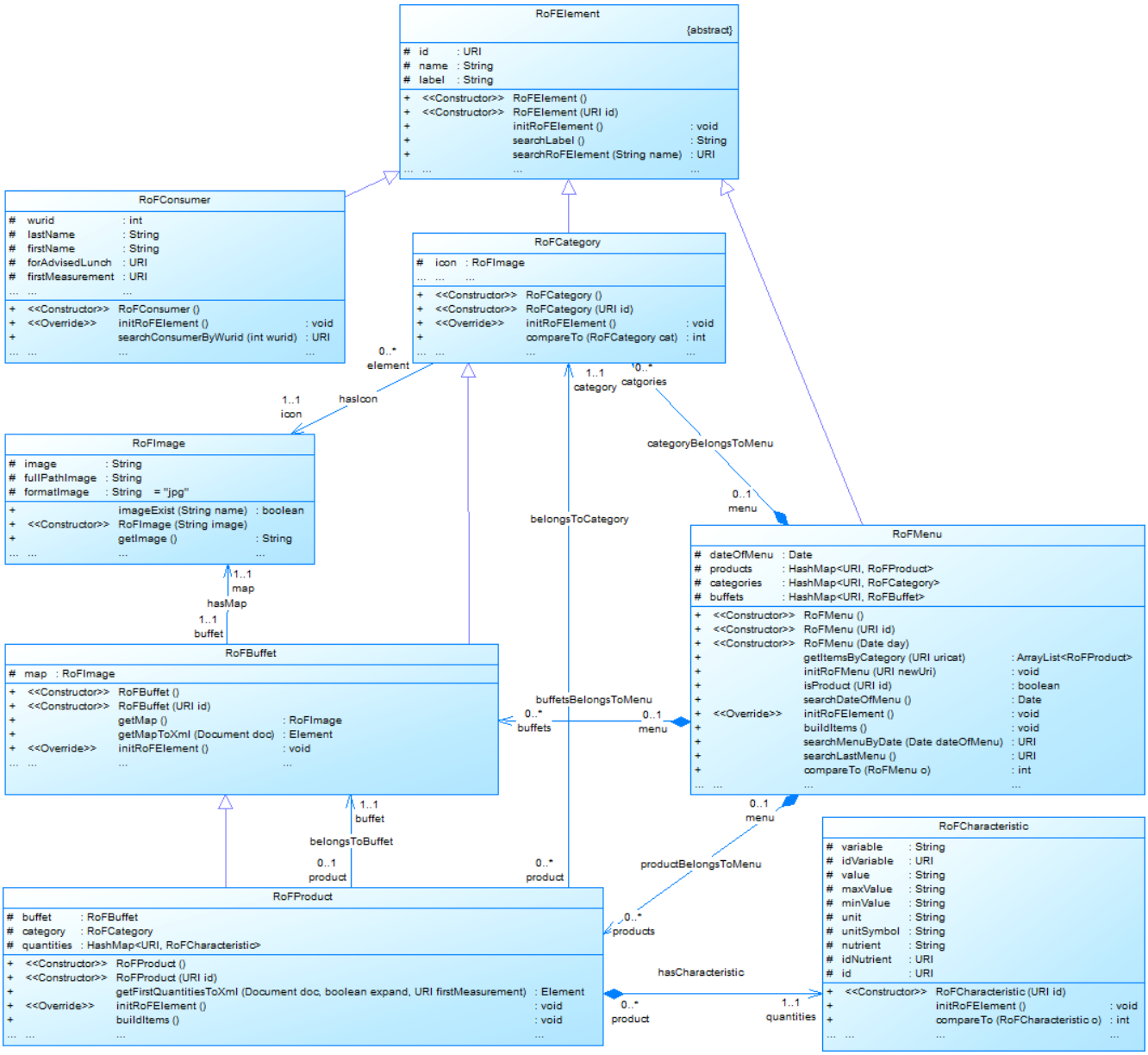


FIGURE 4.2 – Diagramme de classes UML partiel du module métier : conversion RDF vers objets métier

Spécialisation de classes : la classe abstraite *RoFElement* va correspondre à une instance d'un concept RDF de base, par spécialisation les autres classes *RoFCategory*, *RoFBuffet*, *RoFProduct* et *RoFMenu* vont hériter de ces attributs et être enrichies. Pour cela, nous utilisons la méthode *initRoFElement* dans le constructeur de cette classe mère afin de préparer le paramétrage des classes filles, voir listing 4.3.

Paramétrage par spécialisation : chaque sous-type de *RoFElement* va être paramétré grâce à la méthode ***initRoFElement***, définie comme abstraite dans la sur-classe *RoFElement* et qui devra donc être implémentée dans chaque sous-classe. Cette méthode construit une requête SPARQL et interroge la base de données RDF avec l'API *Sesame* afin de compléter l'instance d'objet par les attributs propres à la classe, voir l'appel listing 4.4.

Listing 4.3 – Code paramétrable : constructeur de la classe *RoFElement*

```
/**
 * From an URI, the object is initialize by the 'initRoFElement' which mean with Sesame
      repository in this version. * @param URI of the element.
 */
public RoFElement(URI id) {
  super();
  this.id = id;
  this.name = this.id.getFragment().toString();
  this.label = this.searchLabel();
  this.initRoFElement(); //plugin                architectureAnne
}
```

Listing 4.4 – Exemple de code paramétrant : méthode définie dans sa classe *RoFCategory*

```
public class RoFCategory  extends RoFElement implements Comparable<RoFCategory> {
  ...
  /**
   * Initialize the element from the Sesame repository with the uri of the object.
   */
  public void initRoFElement() {
    if (this.id != null)
      try {
        RepositoryConnection con = SesameForRoF.Instance().getSesame().getConnection();
        // build sparql query
        StringBuilder query = new StringBuilder();
        query.append("SELECT ?lab ?ic ");
        query.append("WHERE { ");
        query.append("  OPTIONAL { <" + this.getId() +"> <"+ PROD.HAS_ICON +"> ?ic}.");
        query.append("  OPTIONAL { <" + this.getId() +"> <" + RDFS.LABEL + "> ?lab. ");
        query.append("  FILTER ( lang(?lab) = \"" + ConfigRoF.LANGUAGE + "\" ) } .") ;
        query.append("} ");
        TupleQueryResult result = con.prepareTupleQuery(QueryLanguage.SPARQL, query.
            toString()).evaluate();
        // result
        List<String> reslib = result.getBindingNames();
        if(result.hasNext()) {
          BindingSet binds = result.next();
          if(reslib.contains("ic") && binds.getValue("ic") != null ) this.icon = new
              RoFImage(binds.getValue("ic").stringValue());
```

```
            if(reslib.contains("lab") && binds.getValue("lab") != null ) this.label =
                binds.getValue("lab").stringValue();
        }
        ...
} } }
```

Cette structure permet une réutilisation et une évolution future plus facile.

Traduction des relations *belongsTo* et *aggregates*

Les composants ontologiques de définition du RoF et des produits, cf. section 3.2, utilisent les propriétés inverses *belongsTo* et *aggregates* pour définir les relations entre les produits et les menus, les catégories et les buffets. Nous avons traduit cette notion de plusieurs manières, suivant les cas (cf. diagramme de classes figure 4.1) :

- par une composition la tentationla tentationla tentationproductsmenupour traduire qu'un menu est composé de produits, avec la relation *productBelongsToMenu* sur le diagramme de classes ;
- par l'association *belongsToCategory* entre un produit et une catégorie ;
- par l'association *belongsToBuffet* entre un produit et une buffet.

Les associations vont amener plus de flexibilité lors de la consultation des produits par catégories et/ou buffets.

La classe *RoFMenu* joue également le rôle de gestionnaire du contexte du restaurant par les deux autres compositions des classes *RoFCategory* et *RoFBuffet* qui apparaissent lors des menus. Une des raisons de ce choix de conception est que dans cette version du prototype, le menu est le point d'entrée pour la consultation des produits.

En raison du rôle de "conteneur" joué par le menu, nous avons choisi de gérer la profondeur de conversion des objets *Menu*, mais aussi *Product* avec les caractéristiques, grâce à la méthode particulière *buildItem*. Elle va construire le produit et les menus complets c'est-à-dire, dotés de leurs composants (cf. exemple du listing 4.5).

Listing 4.5 – Extrait de la méthode *buildItem* de la classe*RoFMenu*

```
/**
 * Find all products, categories, buffets available for a lunch in the repository.
 */
public void buildItems() {
  this.products = new HashMap<URI,RoFProduct>();
  ...
  RepositoryConnection con = SesameForRoF.Instance().getSesame().getConnection();
  StringBuilder query = new StringBuilder();
  query.append("SELECT ?prod ?buff ?cat");
  query.append("WHERE { ");
  query.append("  ?prod a <" + PROD.PRODUCT + "> . ");
  query.append("  ?prod <" + PROD.BELONGS_TO  + "> <" + this.getId() +"> . ");
  ...
  TupleQueryResult result = con.prepareTupleQuery(QueryLanguage.SPARQL, query.toString
      ()).evaluate();
  while(result.hasNext()) {
    ...
```

```
    RoFProduct prod = new RoFProduct(new URI(bindingSet.getValue("prod").stringValue())
        );
    this.products.put(prod.getId(), prod);
    ...
    }
...
```

Pour résumer, nous avons donc mis en place un environnement de connaissances comprenant :

- un serveur de base de données RDF *Sesame*, exploitable grâce à l'API Java *Sesame* fournissant également un système de raisonnement ;
- la définition de plusieurs composants ontologiques et des jeux de tests de consommateurs, menus, produits et caractéristiques ;
- un module de conversion d'objets RDF à des objets métiers java.

4.3 Personnalisation du menu du jour : *RoFAdvisedMenu*

Les développements précédents nous ont permis de mettre en place un environnement de connaissances et un module facilitant le passage d'instances RDF à des instances d'objets métiers. De ces développements, il en ressort la création, basée sur nos connaissances, d'un menu complet (catégories, buffets, produits, caractéristiques). Nous allons maintenant nous intéresser à la personnalisation de ce menu pour un consommateur. Les trois profils d'utilisateur sont gérés par le SBC. A chacun d'eux sont associés des recommandations (cf. section 3.4). La personnalisation du menu : annotation des éléments, personnalisation du message (contenu et forme), gestion de la tentation, est décrite dans les parties suivantes, ainsi que la gestion du plateau virtuel.

4.3.1 Évolution du modèle d'objet métier

La figure 4.3 illustre les évolutions du diagramme de classes du module de création d'objets métier à partir de la base RDF. Les évolutions implémentent maintenant les premières règles métiers de la première version de notre prototype. Nous avons fait apparaitre le concept menu personnalisé avec la classe *RoFAdvisedMenu*. Cette classe est une spécialisation de la classe *RoFMenu*. Ce menu personnalisé est lié à un consommateur, qui va pouvoir personnaliser les éléments du menu (produits, catégories et buffets), activer le processus de tentation et personnaliser le message diffusé en fonction de la caractéristique pertinente pour l'utilisateur. De plus, les éléments personnalisables ont un nouvel attribut ***advised***. Cet attribut peut contenir un entier correspondant à code défini dans la configuration du système : 1, conseillé ; -1,à éviter et 0, pas d'indication. Enfin, nous avons une classe *Tray* représentant le plateau virtuel de l'utilisateur.

4.3 Personnalisation du menu du jour : *RoFAdvisedMenu*

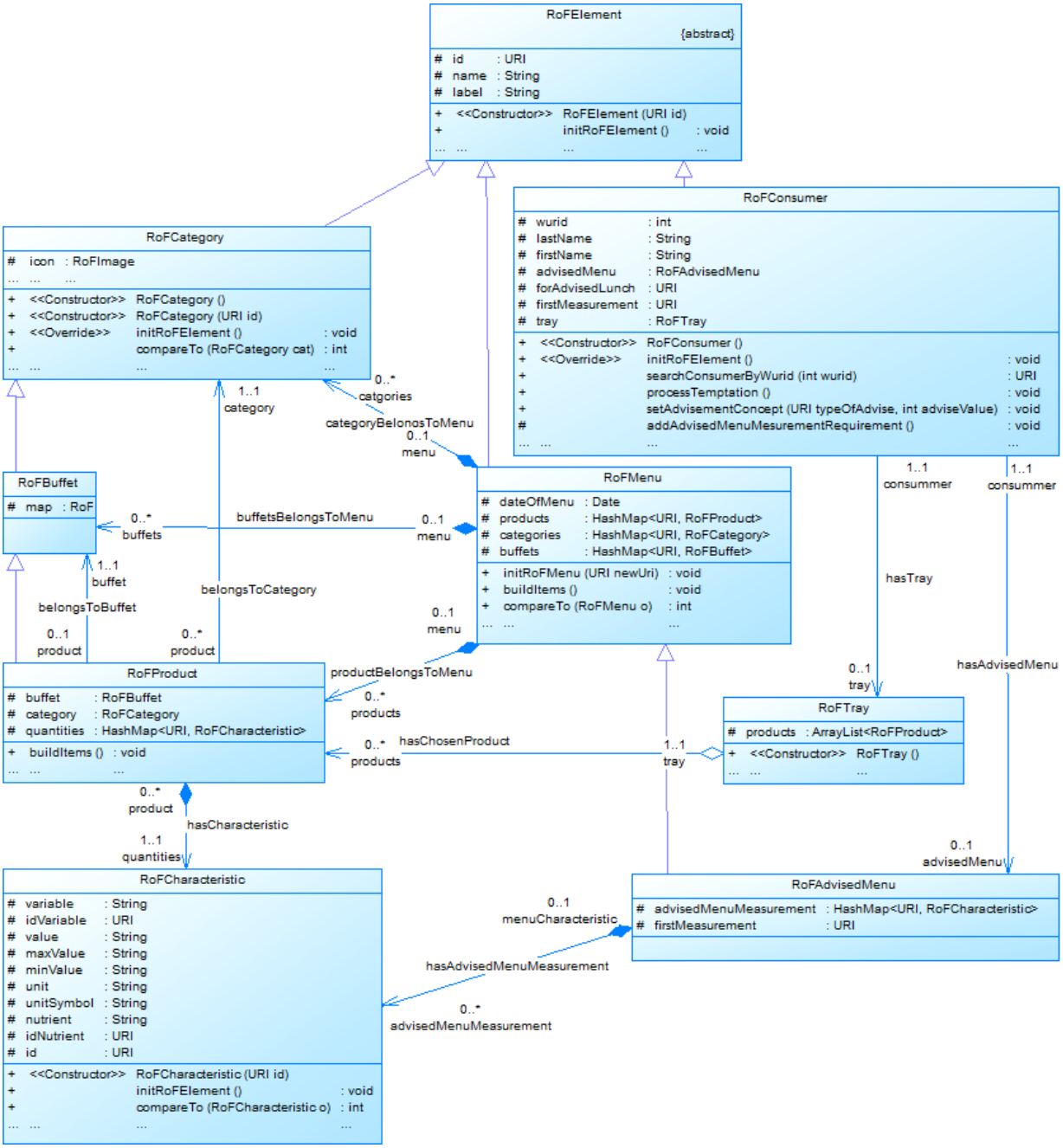


FIGURE 4.3 – Diagramme de classes UML partiel du module métier : personnalisation

4.3.2 Gestion des conseils et des recommandations

Quand un utilisateur, dont le profil existe dans notre base RDF est instancié, une instance de la classe *RoFAdvisedMenu* est créée. Celle-ci est initialisée, à l'image du menu du jour général. Lors de l'initialisation de l'objet métier du consommateur, l'attribut *forAdvisedLunch* va être lui-même initialisé avec l'URI correspondant à une instance RDF *AdvisedLunch*, concept central de l'ontologie des recommandations (cf. section 3.4). Pour rappel, ce concept est constitué d'une collection de conseils permettant d'annoter les éléments du contexte du restaurant comme "conseillé" ou "à éviter", d'une définition de caractéristiques pertinentes, du type de gestion de la tentation et du mode d'affichage conseillé. L'idée ensuite est d'exploiter ce "lien" vers les recommandations pour l'utilisateur pour appliquer les fonctionnalités de personnalisation du menu (annotation, tentation et message personnalisé), sans avoir à créer d'objet métier correspondant aux recommandations mais s'en servir, par contre, pour sélectionner les éléments dans la base RDF. Techniquement, on va être amené à créer des requêtes SPARQL dynamiques en fonction des recommandations.

4.3.3 Annotation *advised* et *avoid*

Au départ, l'objet menu consacré à l'utilisateur est le même que le menu général, sans personnalisation. Il contient tout les objets métiers produits, catégories et buffets disponibles ce jours là au RoF. La méthode *setAdvisementConcept(URI typeOfAdvise, int adviseValue)* va en fonction du type de conseil, donnée dans l'ontologie par les relations (*isAdvise* ou *isAvoid*) récupérer tous les conseils (instances du concept *Advice*) et construire la requete SPARQL permettant, pour chaque conseil, de récupérer l'URI des produits à annoter en fonction du type de relation visée. Si nous prenons l'exemple de John, la figure 4.4 illustre le fait que John a trois conseils alimentaires dont un concerne un élément du menu à éviter (*isAvoid*) et deux autres permettront d'indiquer les éléments considérés comme bénéfiques (*isAdvised*) et d'annoter comme tels nos objets métiers correspondants. Le listing 4.6 donne le code java permettant de sélectionner ces conseils.

Listing 4.6 – Extrait de la méthode *setAdvisementConcept* - construction d'une requête SPARQL pour récupérer les instances d'*Advice*

```
public void setAdvisementConcept(URI typeOfAdvise, int adviseValue) {
...
  RepositoryConnection con = SesameForRoF.Instance().getSesame().getConnection();
  // get advised product element without measurement requirement
  StringBuilder query = new StringBuilder();
  query.append("SELECT ?ctype ?advElem ?req ");
  query.append("WHERE { ");
  query.append(" <" +this.forAdvisedLunch + "> <" + typeOfAdvise + "> ?advElem . ");
  query.append(" ?advElem  <" + ADV.CONCERN_TYPE + "> ?ctype . ");
  query.append(" ?advElem  <" + ADV.HAS_REQUIREMENT + "> ?req . ");
```

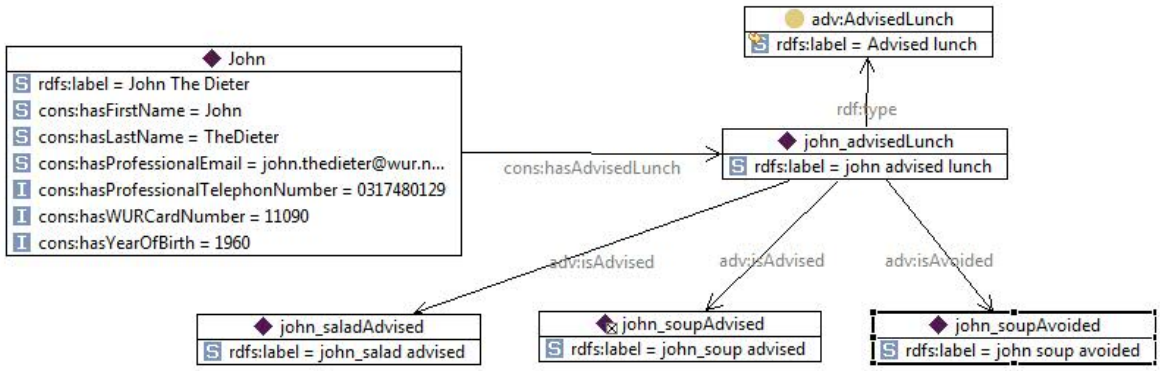


FIGURE 4.4 – Conseils pour John (TopBraid)

```
  query.append(" OPTIONAL { ?advElem <" + ADV.HAS_MEASUREMENT_REQUIREMENT + "> ?mr } .
      ");
  query.append(" FILTER ( ! bound( ?mr ) ) . ");
  query.append("} ");
  query.append("ORDER BY ?advElem ");
...
}
```

La deuxième étapes est la sélection des produits à annoter en fonction de la relation recherchée. Pour continuer notre exemple, nous allons nous intéresser à un des conseils pour John, illustré par la figure 4.5. Ce conseil est donné pour des éléments de menu de type *Product*, donné par la relation *concernedType*. Ce conseil est constitué de *Requirement* qui doivent tous être satisfait et permettent de sélectionner les produits. Ici le premier *Requirement* indique que le produit appartient à la catégorie des soupes, le deuxième que cette soupe doit être de type bouillon. L'extrait de la méthode *setAdvisementConcept*, listing 4.7, montre la création de la requête SPARQL en fonction des résultats de la précédente. Chaque *Requirement* est traduit pas des instructions SPARQL de sélection et ajouter au précédent (connecteur logique AND, OR entre *Advice*). La dernière instruction assigne à l'attribut *advised* du produit métier un entier traduisant pour le cas de John que le produit est conseillé.

Listing 4.7 – Extrait de la méthode *setAdvisementConcept* suite - construction d'une requête SPARQL pour récupérer les instances de produit et annotation

```
public void setAdvisementConcept(URI typeOfAdvise, int adviseValue) {
...
 while(advElem != null) {
  // new product to find
  if( !ctype.contentEquals(ctypeOld) || !advElem.contentEquals(advElemOld) ){
   query = new StringBuilder();
   query.append("SELECT ?prod ");
   query.append("WHERE { ");
 }
 // remained requirement to add
  query.append(" { ");
```

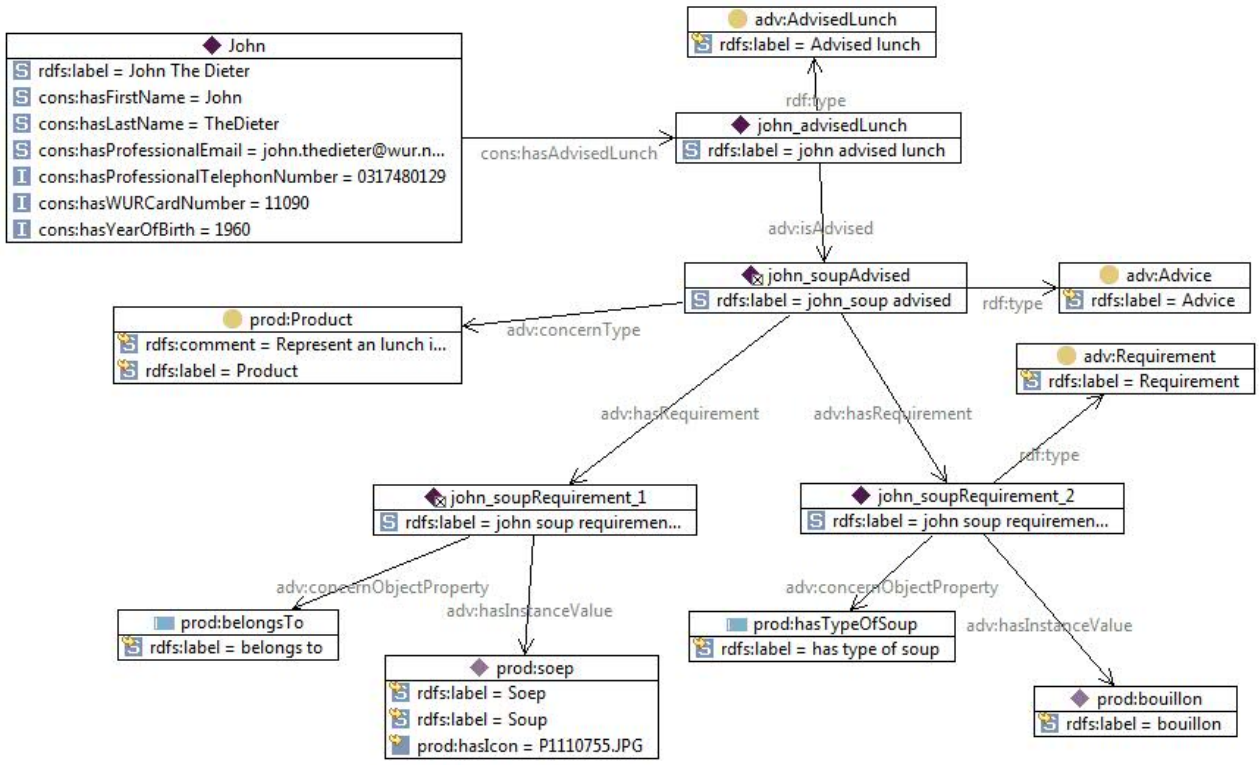


FIGURE 4.5 – Conseils concernant les soupes pour John (TopBraid)

```
  query.append(" ?prod a <" + ctype + "> . ");
  query.append(" <" + req + "> <" + ADV.HAS_INSTANCE_VALUE + "> ?ival" + i + " . ");
  query.append(" <" + req + "> <" + ADV.CONCERN_OBJECT_PROPERTY + "> ?oprop" + i + "  .
       ");
  query.append(" ?prod ?oprop" + i + " ?ival" + i + " . ");
  query.append(" } ");
  query.append(" UNION ");
  query.append(" { ");
  query.append(" ?prod a <" + ctype + "> . ");
  query.append(" <" + req + "> <" + ADV.HAS_INSTANCE_VALUE + "> ?ival" + i + " . ");
  query.append(" FILTER ( ?prod = ?ival" + i + " ) . ");
  query.append(" } ");
  ...
  // annotation
  prodResult = con.prepareTupleQuery(QueryLanguage.SPARQL, query.toString()).evaluate()
      ;
  while(prodResult.hasNext()) {
   prodBinds = prodResult.next();
   URI prod = new URI(prodBinds.getValue("prod").stringValue());
   // the product is available
   if(products.containsKey(prod)) products.get(prod).setAdvise(adviseValue);
  ...
}}
```

Ce processus d'annotation peut s'effectuer également sur d'autre concepts comme les catégories ou des buffets.

4.3.4 Gestion de la tentation

La tentation est traduite dans l'ontologie des recommandations par un mode de tentation. Trois modes sont proposé dans notre prototype :

fullTemptation tous les produits conseillé ou pas sont visibles ;

moderateTemptation les produits déconseillés sont éliminés des produits visibles par le consommateur ;

freeTemptation seul les produits considérés comme bénéfiques sont laissé, tous les autres, ceux considérés comme néfastes ou ceux dont on n'a pas fixé d'information sont supprimés du menu personnalisé.

Le mode de tentation est donné par l'ontologie des recommandations, pour John il s'agit du mode *fullTemptation*, donc les éléments ne sont pas supprimés. La figure 4.6 schématise la formalisation de la tentation pour John.

Cette règle métier est implémentée par la fonction *processTemptation()* qui récupère en premier le mode de tentation dans la base, puis suivant les modes de tentation va supprimer de l'objet métier *AdvisedMenu* les éléments non souhaitable d'être mis à disposition.

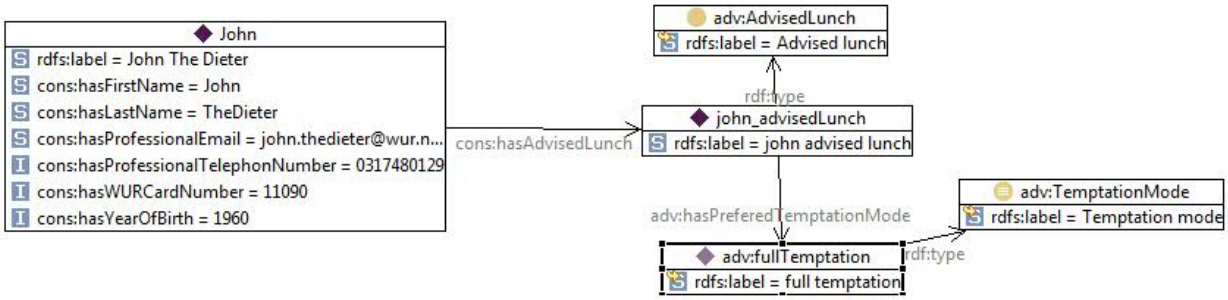


FIGURE 4.6 – Mode de tentation pour John (TopBraid)

4.3.5 Caractéristique pertinente

Dans notre prototype, une seule caractéristique est traitée comme pertinente et sera affichée en priorité, notamment un produit est associé à une seule information dans l'affichage et qui correspond à cette caractéristique. Par défaut, la caractéristique de définie dans la configuration du système est l'énergie en kilo-calorie que peut fournir un aliment. Pour un menu à personnaliser, elle est formalisée avec l'ontologie des recommandations. Pour John, c'est aussi l'énergie qui est la caractéristique la plus pertinente à afficher pour lui (cf. figure 4.7). Par contre pour Wim, c'est la quantité de vitamine et pour Esther le prix des aliments. Cette caractéristique peut donc être également un mesure nutritive.

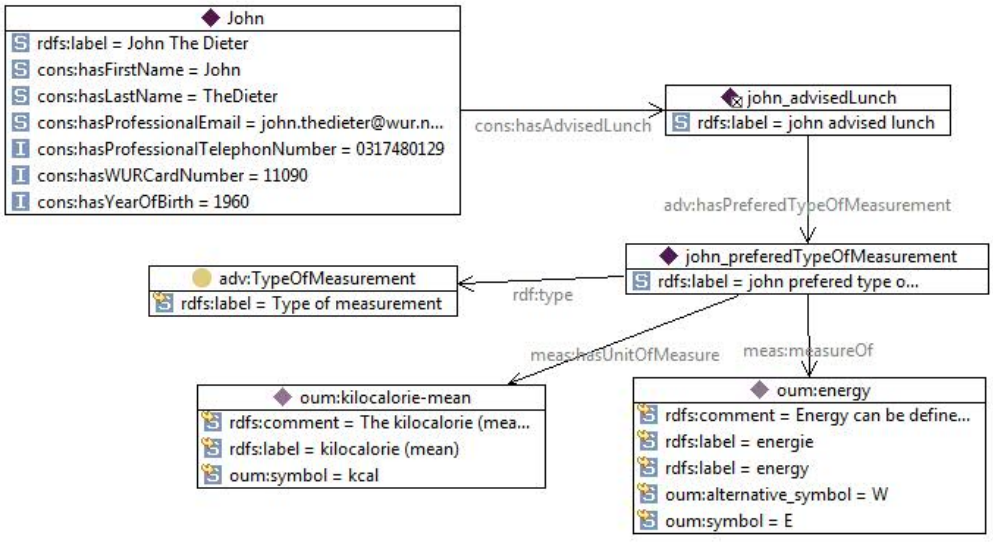


FIGURE 4.7 – Caractéristique pertinent à afficher pour John (TopBraid)

Au niveau implémentation des objets métier, l'attribut *firstMeasurement* de type

URI, va correspondre à l'instance du type de mesure préféré par le consommateur. Par exemple, pour John on aura http://www.wurvoc.org/rof/johnTheDieter#john_preferedTypeOfMeasurement
Cet attribut est initialisé par les objets métiers par la fonction d'initialisation de l'objet. Ceci implique également un traitement spéciale pour cette caractéristique, qui est sélectionnée parmi toutes celles accompagnant le produit.

4.3.6 Gestion du plateau virtuel

Un utilisateur "connu" va avoir la possibilité de gérer un plateau virtuel. Pour cela, le type *Tray* a été défini. Il y a une relation d'agrégation entre produits et plateau. Un attribut de type *Tray* existe pour un consommateur, cf. diagramme de classe 4.3. Il est intéressante de remarquer que les recommandation définies au niveau du menu vont être servir à le paramétrer au niveau de la consultation et de paramétrer les réactions du système. Par exemple, une des recommandations de John (recommandation *Rec. 6*, cf. chapitre 1.1) de la part de la diététicienne, dit que son repas global ne doit pas dépasser plus de 600 kcal. Celle-ci est traduite dans notre ontologie (cf. figure 4.8) et sera exploiter pour le plateau qui correspond au repas global. Cette exploitation, au niveau de la couche métier de notre prototype, va consister à indiquer des propriétés pour le plateau. La gestion de l'affichage est laissé à l'application *Smartphone*. Au niveau implémentation, cette caractéristique globale est de type *RoFCharacteristic* est initialisée de la même manière que les autres, par contre la gestion de ces caractéristiques se fait dans la classe *RoFAdvisedMenu* qui a pour attribut *advisedMenuMeasurement*, une structure de données (*HashMap*), issue de la relation de composition qui existe entre les types *RoFCharacteristic* et *RoFAdvisedMenu*, voir diagramme de classes figure 4.3.

Pour résumer, la personnalisation du menu et les règles métiers sont gérées à partir du module java développé précédemment (voir section 4.2.4). Il est basé sur une architecture de réutilisation par spécialisation et paramétrable grâce à des requêtes d'interrogation SPARQL. L'évolution de ce module, va permettre de faire apparaitre les types *RoFAdvisedMenu* et *RoFTray*, attribut d'un consommateur, permettant respectivement la gestion de menu personnalisé, sur lequel on va pouvoir appliquer les règles métiers implémentées et la gestion du plateau virtuel. Les règles métiers implémentées sont : marquage des éléments (produit, catégorie ou buffet complet) comme considérés comme bénéfiques ou néfastes au consommateur, personnalisation du message avec la sélection pour chaque produits des informations concernant une seule variable pertinente, gestion de la tentation par suppression des produits visibles en fonction d'un degrés de liberté.

Ces objets ainsi construits vont être ensuite mis a disposition à travers une architecture distribuée afin d'être consultés grâce a une application sur *Smartphone*.

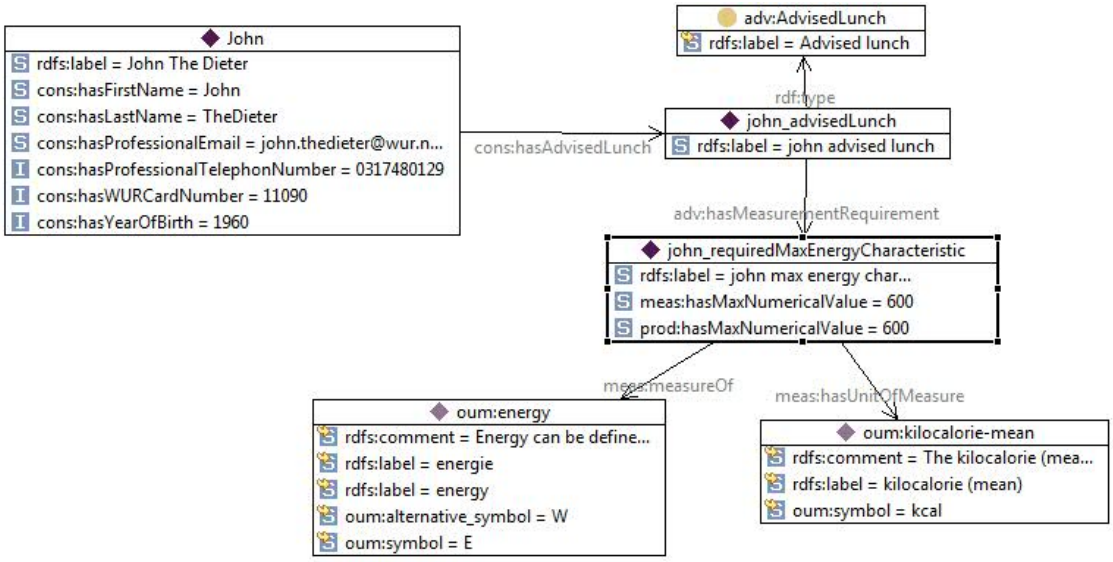


FIGURE 4.8 – Recommandation globale d'un repas de 600 kcal maximum pour John (TopBraid)

4.4 Web Service

Le système de diffusion a certaines exigences dont la consultation des menus et produits, avec leurs caractéristiques, sur des plateformes mobiles. Pour répondre à ce besoin, nous avons mis en place **une architecture distribuée**, basée sur un Web Service.

En effet, les caractéristiques des Web Services ont orienté notre choix :

- interaction entre systèmes hétérogènes, distants, avec couplage faible ;
- utilisation des protocoles internet, notamment le protocole standard HTTP ;
- échange de message XML possible ;
- utilisation de clients légers possibles, type navigateur web.

En effet, il existe une grande variation de plateformes clientes de type ordiphone, et qui possède des caractéristiques matérielles, des systèmes d'exploitation et des langages de programmation d'application différents, cas *iPhone* et *Windows Phone* par exemple.

L'architecture de services Web choisie est **REST** (voir section). Ce choix a été fait en fonction de plusieurs critères :

- léger car basé uniquement sur le protocole HTTP, et donc moins de complexité engendrée par l'utilisation des différents intermédiaires des architectures SOAP par exemple ;
- architecture orientée ressources par rapport à une architecture basée sur les traitements ;

– interface d'accès et de manipulation des ressources uniformes : GET, POST, DELETE du protocole HTTP ;
– possibilité, dans une version future, d'améliorer les interactions entre la base de données RDF, *Sesame*, et un Web Service type REST par des extensions disponible de *Sesame*.

Pour implémenter cette architecture, nous avons utilisé de le framework Open Source *Restlet*, écrit en Java. Il correspondait bien à nos besoins : filtres d'accès aux ressources basés sur XPath, support de représentation XML, extension permettant de déployer rapidement une application Restlet sur les téléphones mobiles *Google Android*.

Dans cette section est décrite l'implémentation de l'application est basée sur une architecture REST et permet la diffusion de ressources : les menus avec leur composition (produits, catégories, etc. mais aussi images).

Format d'échange

Nous avons choisi d'utilisé un format utilisant le langage XML pour communiquer entre le serveur et la plateforme mobile. Le listing 4.8 illustre un exemple de représentation XML permettant de décrire la catégorie "fruit" d'un menu.

Listing 4.8 – Extrait de représentation XML d'un menu

```
<rof>
 <menu>
  <id>http://www.wurvoc.org/rof/product#menu20100730</id>
  <name>menu20100730</name>
  <label>menu20100730</label>
  <dateOfMenu>2010-07-30</dateOfMenu>
  <categories>
   <category>
    <id>http://www.wurvoc.org/rof/product#fruit</id>
    ...
    <products>
     <product>
      <id>http://www.wurvoc.org/rof/product#appel</id>
      ...
      <buffet>
       <id>http://www.wurvoc.org/rof/product#dessertsAndJuicesBuffet</id>
      </buffet>
      <quantities>
       <quantity>
        <unit symbol="kcal">kilocalorie (mean)</unit>
        <value>98,74</value>
        <variable>energie</variable>
        <idvariable>http://www.wurvoc.org/vocabularies/om-mc-1.6/energy</idvariable>
       </quantity>
       ...
</rof>
```

Ces données sont produites par notre module métier, en effet chaque classe implémente la fonction *toXML* qui produit un élément de type XML correspondant à la

description de son type et de ses attributs (cf. extrait du diagramme de classes, figure 4.9). Cette implémentation est facilité par la spécialisation. Chaque élément de base va avoir un identifiant, *id*, correspondant à l'URI de l'objet métier et instance RDF, ainsi qu'un label pour l'affichage dans la bonne langue et un nom.

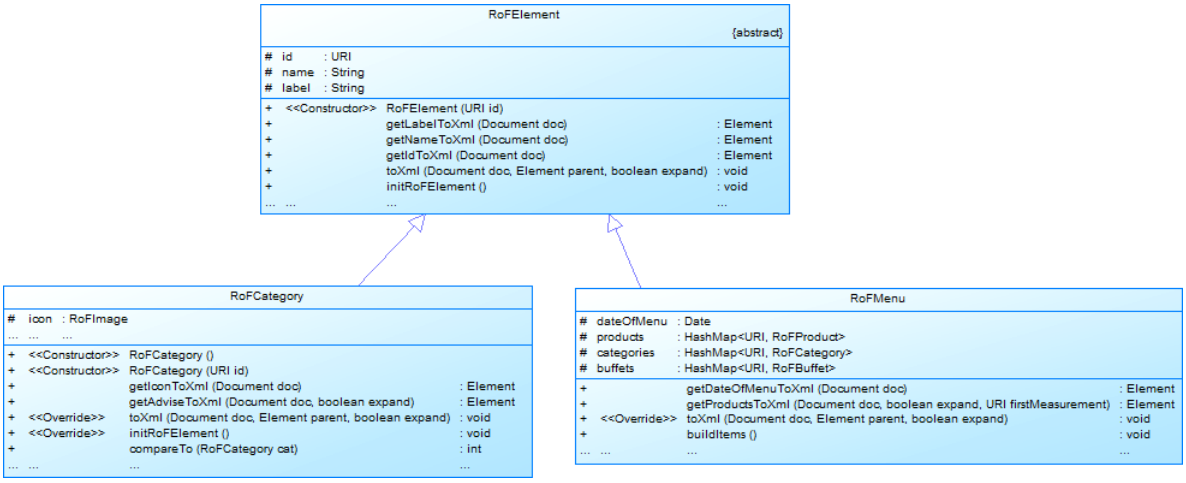


FIGURE 4.9 – Extrait du diagramme de classes du module métier

La structure des données XML correspond à la structure arborescente de consultation du menu. Dans notre prototype nous avons implémenté la consultation du menu générale par : "menu/catégorie/produit/caractéristiques". Dans le cas d'un menu personnalisé, elle se fait par : "utilisateur/identifiant/menu/catégorie/produit/caractéristiques". Le plateau virtuel est consultable par "utilisateur/identifiant/plateau". Cette structure correspond vraiment à un consultation de ressources.

Application Restlet

Les classes développées à l'aide du framework Restlet sont données par la figure 4.10. La classe *LunchAtRoFApplication* est la classe mère de l'application, elle va gérer les ressources. Pour cela, elle possède des attributs *consumer* et *menu*. Le premier est une collection java (*ConcurrentMap*) permettant de gérer les accès concurrents aux données contenues. C'est une map associant l'identifiant d'un utilisateur, son numéro d'employé, à une instance de type *RoFConsummer*. Ainsi chaque instance de consommateur du système est géré par cette structure. L'attribut *menu*, correspond au menu du jour général. Chaque menu est mis à jour automatiquement par le système, qui vérifie les dates des menus de la consommation du service Web.

Cette classe définie également, les routes ou chemins d'accès aux ressources. Le listing 4.9 illustre plusieurs définitions de lien entre un chemin d'accès à une ressource

4.4 Web Service

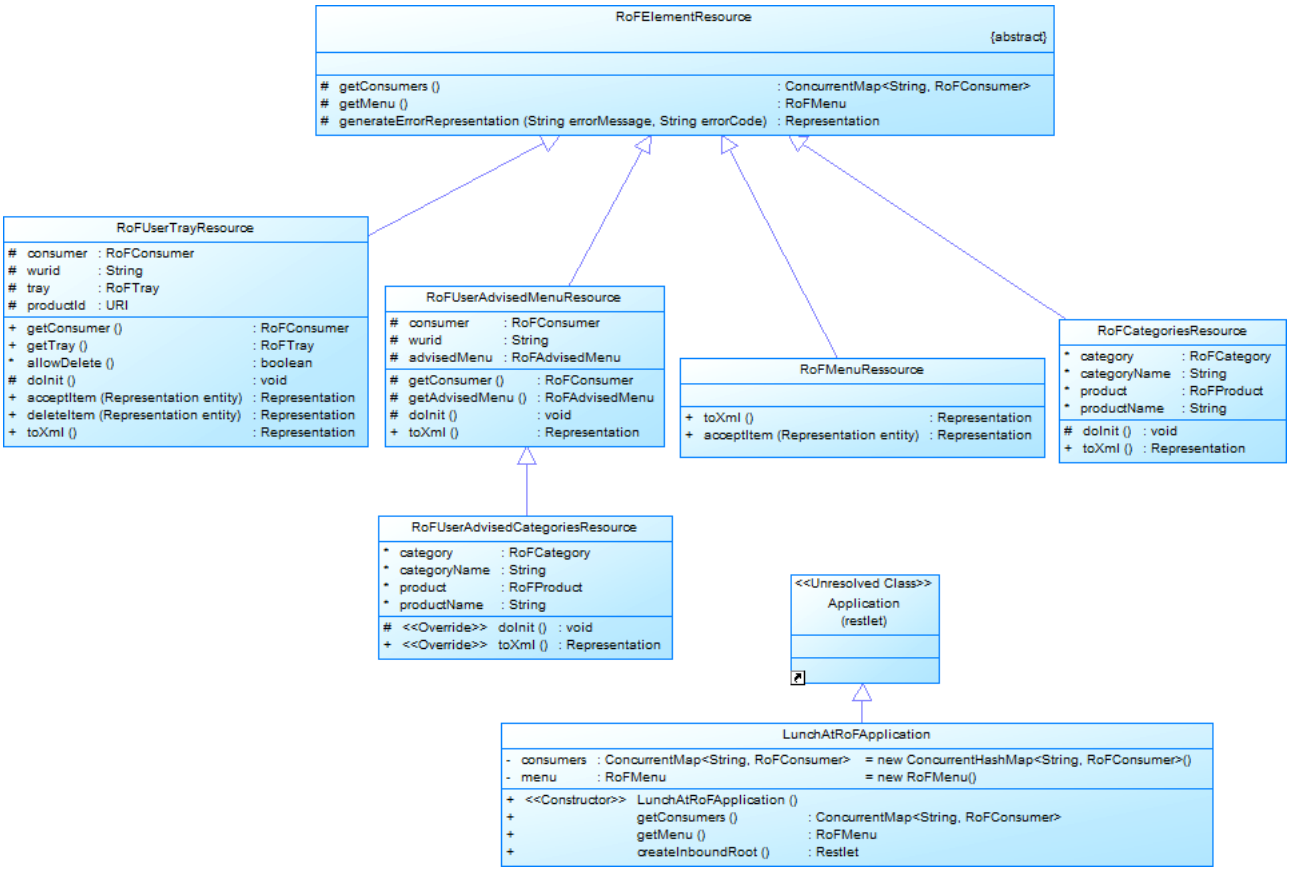


FIGURE 4.10 – Diagramme de classes du Web Service

et une classe ressource (cf. figure 4.10). Certains chemins sont paramétrables par des filtres, chaîne de caractères variable entre accolades. Par exemple, le paramètre *wurid* du chemin "/user/wurid/tray" permet de sélectionner en fonction de l'identifiant le bon consommateur et ensuite son plateau virtuel.

Listing 4.9 – Extrait de la classe *LunchAtRoFApplication* - initialisation des chemins d'accès aux resources

```
/** Creates a root Restlet that will receive all incoming calls. */
@Override
public synchronized Restlet createInboundRoot() {
   // Create a router Restlet that defines routes.
   Router router = new Router(getContext());
   // Define a route without advice to the menu
   router.attach("/menu/",RoFMenuRessource.class);
   ...
   // Define a route for the resource " list of categories
   router.attach("/menu/categories", RoFCategoriesResource.class);
   ...
   // Define a route for the resource " list of characteristics of one product
   router.attach("/menu/categories/{categoryName}/{productName}", RoFCategoriesResource
       .class);
   ...
   // Define a route for the resource "client" and his menu (date, id)
   router.attach("/user/{wurid}/menu", RoFUserAdvisedMenuResource.class);
   ...
   // define a route  for the resource "client" and his tray content
   router.attach("/user/{wurid}/tray", RoFUserTrayResource.class);
   ...
   return router;
}
```

Méthodes implémentées : GET, PUT et DELETE

Chaque classe ressource est chargée de fournir des représentations de ressources, représentations XML pour nous, mais aussi d'y appliquer des traitements :

- GET : envoi de la représentation au format souhaité, par exemple la représentation XML d'un menu ou des catégories d'un menu ;
- DELETE : suppression d'un élément d'une représentation, par exemple un produit du plateau ;
- POST : ajout d'un élément à une ressource, par exemple un produit au plateau.

La signification de ces méthodes sont définies dans la norme RFC2616 de HTTP1.1 (cf. Fielding et al. (1999)).

Avant d'exécuter une des trois méthodes désignées ci-dessus, une méthode d'initialisation du framework Restlet est exécutée automatiquement. Nous l'avons redéfinie afin de traiter les filtres et ainsi sélectionner les bonnes ressources ou objets métiers. Le listing 4.10 montre l'implémentation de la méthode GET de la classe ressource gérant le plateau virtuel d'un consommmateur.

Listing 4.10 – Extrait de la classe *RoFUserTrayResource* - implémentation de la méthode GET pour le plateau virtuel

```
public class RoFUserTrayResource extends RoFElementResource {
...
/**
 * Returns a description of the resource tray
 */
@Get("xml")
public Representation toXml() {
        // Generate the right representation according to its media type.
        try {
                DomRepresentation representation = new DomRepresentation(MediaType.
                    TEXT_XML);

                // Generate a DOM document representing the consumer
                Document doc = representation.getDocument();

                Element rElem = doc.createElement("rof");
                doc.appendChild(rElem);

                Element eCons = doc.createElement("consumer");
                rElem.appendChild(eCons);
                this.consumer.toXml(doc, eCons, false);
                eCons.appendChild(this.consumer.getAdvisedMenuToXml(doc, false));
                eCons.appendChild(this.consumer.getTray().getTrayToXml(doc, false));

                doc.normalizeDocument();

                // Returns the XML representation of this document.
                return representation;
        } catch (IOException e) {
                e.printStackTrace();
        }
        return null;
}
...
}
```

Serveur d'images

Dans notre projet, chaque élément du menu est associé à une illustration. Pour mettre à disposition ces images, nous avons ajouter un serveur de fichiers statiques à notre application. Nous avons utilisé pour cela le framework Restlet qui gère la notion de répertoire dédié et correspond à un accès vers un répertoire de notre serveur où se trouve stockées les images. Les images seront donc accessible pas leur nom.

Nous avons donc mis en place un Web Service REST, basé sur le framework Java Restlet, permettant :

– de produire sous forme XML, la représentation des menus, des catégories de menu, des produits par catégories, des caractéristiques par produit, etc. ;

- de donner accès par un système d'adresses paramétrables, à des ressources personnalisées ;
- de gérer l'ajout et la suppression de produit dans le plateau virtuel d'un consommateur ;
- de mettre à disposition les illustrations d'éléments de contexte du restaurant.

La suite du document est consacrée à l'application de consultation du *Smartphone*.

4.5 Application sur Smartphone

Le rôle de l'application ordiphone est essentiellement un l'affichage du menu. Son but est de permettre au consommateur de consulter son menu personnalisé par notre système suivant les recommandations du modèle PEA. Dans l'architecture du système mis en place, nous avons essayé de limiter le plus possible les traitements effectués par l'ordiphone, le but étant de rester le plus possible souple face à l'hétérogénéité du marché et des différents types de d'ordiphone.

Les cas d'utilisation utilisateur sont décrits par le diagramme UML figure 4.11. Pour résumer, depuis son terminal mobile, l'utilisateur doit avoir accès aux fonctionnalités suivantes :

- consultation du menu : les catégories, les produits avec leurs détails,
- consultation et gestion d'un plateau virtuel du repas,
- enregistrement de l'application auprès de notre système.

Parmi ces fonctionnalités, la manière d'afficher les informations va rester sous la responsabilité du concepteur de l'application Smartphone. En effet, un jeu de méta-données produit par le PEA va fournir la manière dont il est souhaitable que les informations soient affichées, par exemple les données brutes vs histogramme de la valeur nutritionnelle des aliments. Dans ce prototype, seul le mode d'affichage "données brutes" est implémenté.

Vues de l'application

La navigation entre les pages peut être schématisée grâce au diagramme UML d'activité fig. 4.12.

La partie vue de l'application est écrite avec le langage de balises XAML et le code joint en C#.
Il y a trois types de pages :

- les pages de registration et de bienvenue ;

4.5 Application sur Smartphone

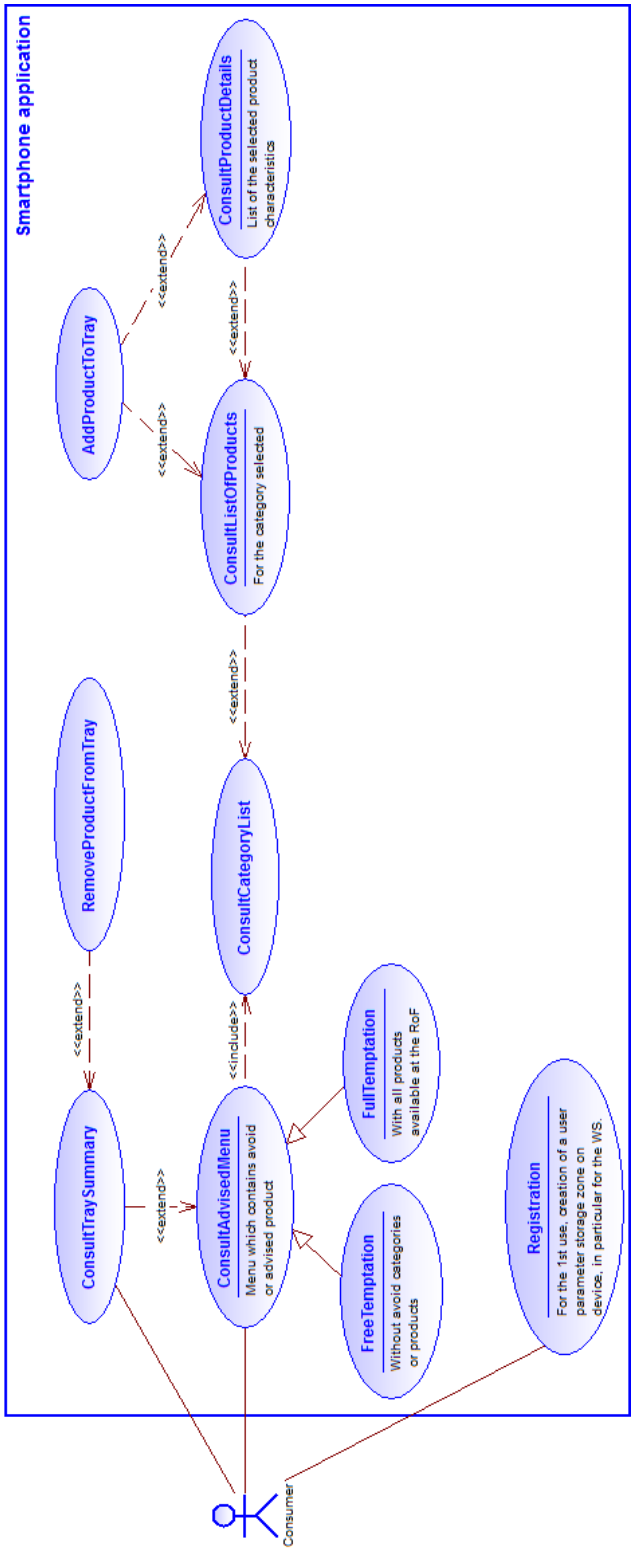


FIGURE 4.11 – Diagramme des cas d'utilisation de l'application smartphone

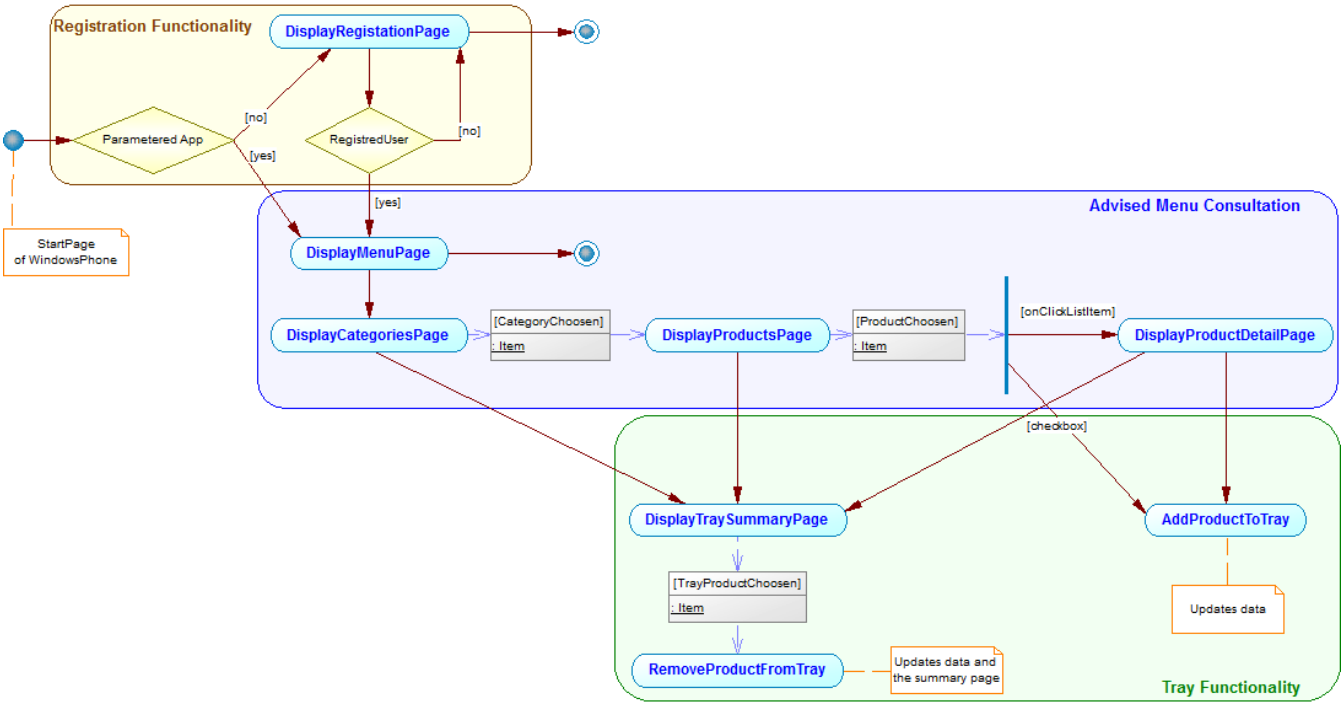


FIGURE 4.12 – Diagramme de navigation de l'application smartphone.
Les recommandations WP7 impliquent : (i) depuis chaque page, l'application peut être stoppée (ii) le bouton "BackKey" permet de remonter l'historique de navigation, depuis n'importe quelle page, jusqu'à la page de démarrage

- les pages de consultation du menu, au nombre de trois :
  - la liste des catégories,
  - la liste des produits pour une catégorie,
  - le détail d'un produit avec ses principales caractéristiques et sa localisation dans le restaurant ;
- la page du plateau virtuel contenant une liste de produits.

L'utilisateur du *Smartphone* est représenté par la classe *User*, visible sur le diagramme de classes (figure 4.14) des principales vues. Chacune des classes contient une propriété *User*. Cet objet correspond à l'objet *RoFConsumer* du module métier du système de diffusion et au concept *Consumer* de l'ontologie. Dans le cas de l'utilisateur, la liaison entre ces concepts dans l'ensemble du système, se fait grâce au numéro d'employé du consommateur. Il va nous permettre de pouvoir accéder à la ressource personnel sur le système de diffusion, voir section 4.4 sur le Web Service et l'accès au menu personnalisé. Par exemple, l'url *http ://www.wur.nl/LunchAtRoF/user/11090/tray* permet d'obtenir la ressource correspondant au plateau virtuel de l'utilisateur dont l'identifiant est 11090.

La fonctionnalité de *Registation* traite le cas de la première utilisation de l'appareil, et permet, après saisi du numéro d'employé *wurid*, de céer un espace de stockage isolé [1] et persistant au niveau de l'application. Cet espace stocke alors les paramètres utilisateurs nécessaires à la connexion au Web service.

Les classes *CategoriesPage*, *CategoryPage* et *ProductDetailPage* permettent respectivement d'afficher les catégories disponibles pour un menu, les produits disponibles pour une catégorie et les caractéristiques d'un produit donné. La classe *TrayPage* permet d'afficher la liste des produits contenus dans le plateau du consommateur ainsi qu'un bilan d'une propriété pertinente pour lui comme par exemple la somme des calories des produits. L'exemple de code XAML, listing 4.11, créé un *template* qui, associé par la liaison de données *{Binding Products}* d'une collection en C#, va pour chaque élément de la collection créer un nouvel item. La figure 4.13 montre le résultat obtenu pour une liste de produits, des fruits.

Listing 4.11 – Code XAML - liaison de données avec une liste

```
<ListBox ItemsSource="{Binding Products}" HorizontalAlignment="Stretch" Margin="
    2,92,0,52" Name="LBProducts" VerticalAlignment="Stretch" >
 <ListBox.ItemTemplate>
  <DataTemplate>
   <StackPanel Orientation="Horizontal" Height="72" >
```

1. *Isolated Storage* en anglais est un système de fichier virtuel mis à disposition pour stocker les données des applications Silverlight. cf. http://msdn.microsoft.com/en-us/library/ff402541%28VS.92%29.aspx

FIGURE 4.13 – *Smartphone* - liste de fruits

```
    <RadioButton Content="" GroupName="TrayProducts" Grid.Row="1" Height="58"
        HorizontalAlignment="Left" Margin="0,0,0,0" Name="{Binding Id}"
        VerticalAlignment="Top" Checked="RBSelect_Checked" Width="64" MinHeight="20"
        MinWidth="20" />
    <Image Source="{Binding ImageUrl}" Height="58" Width="77" VerticalAlignment="Center
        " Margin="0,0,0,0"/>
     <StackPanel Width="290" Height="72" VerticalAlignment="Stretch" >
      <TextBlock Text="{Binding Label}" VerticalAlignment="Top" TextWrapping="Wrap"
          FontSize="25" Foreground="White" FontFamily="Segoe WP" Margin="4,0,0,0" />
      <TextBlock Text="{Binding FirstCharact}" VerticalAlignment="Top" TextWrapping="
          Wrap" FontSize="18" Foreground="White" FontFamily="Segoe WP" Margin="6,0,0,0
          " />
     </StackPanel>
     <Image Height="30" Visibility="{Binding IMGKookVisibility}" Width="30"
         HorizontalAlignment="Left" Source="{Binding IMGKookSource}" Name="IMGKook"
         VerticalAlignment="Center" />
    </StackPanel>
   </DataTemplate>
  </ListBox.ItemTemplate>
</ListBox>
```

La liaison de donnée permet la mise à jours automatique de la vue. Pour cela, à chaque contexte de données de vue est associé un modèle de présentation que nous avons créé. Par exemple, au contexte de données de la vue *TrayPage* est associé le modèle de présentation *TrayViewModel*.

Le modèle de présentation de l'application

Le "Modèle de présentation" a un rôle de liaison entre les vues et les données (le modèle). Les données sont reçues par consommation du Web Service de notre système, sous forme de chaîne de caractères au format XML. La gestion du modèle dans notre conception va être géré directement par le modèle de présentation. Par exemple, l'instanciation du modèle de présentation du plateau virtuel, *TrayViewModel*, va impliquer les tâches suivantes :

- création d'un client web et envoie au Web Service, une requête GET sur le ressource correspondante au plateau de l'utilisateur ;
- une fois le téléchargement des résultats effectué, parsage de la chaîne de caractères de réponse avec l'API *LINK to XML* et initialisation du modèle de présentation ;
- utilisation des évènements, disponible en C#, afin de gérer les autres attributs dépendant du modèle.

Dans le cas, du plateau virtuel, nous avons développé les méthodes permettant de supprimer (DELETE) et ajouter (POST) des éléments à la ressources *RoFTray* de notre système.

Notons que nous avons choisi de faire appel au Web Service à chaque changement de vue, et non pas de charger la structure complète du menu au départ. Ce choix a été

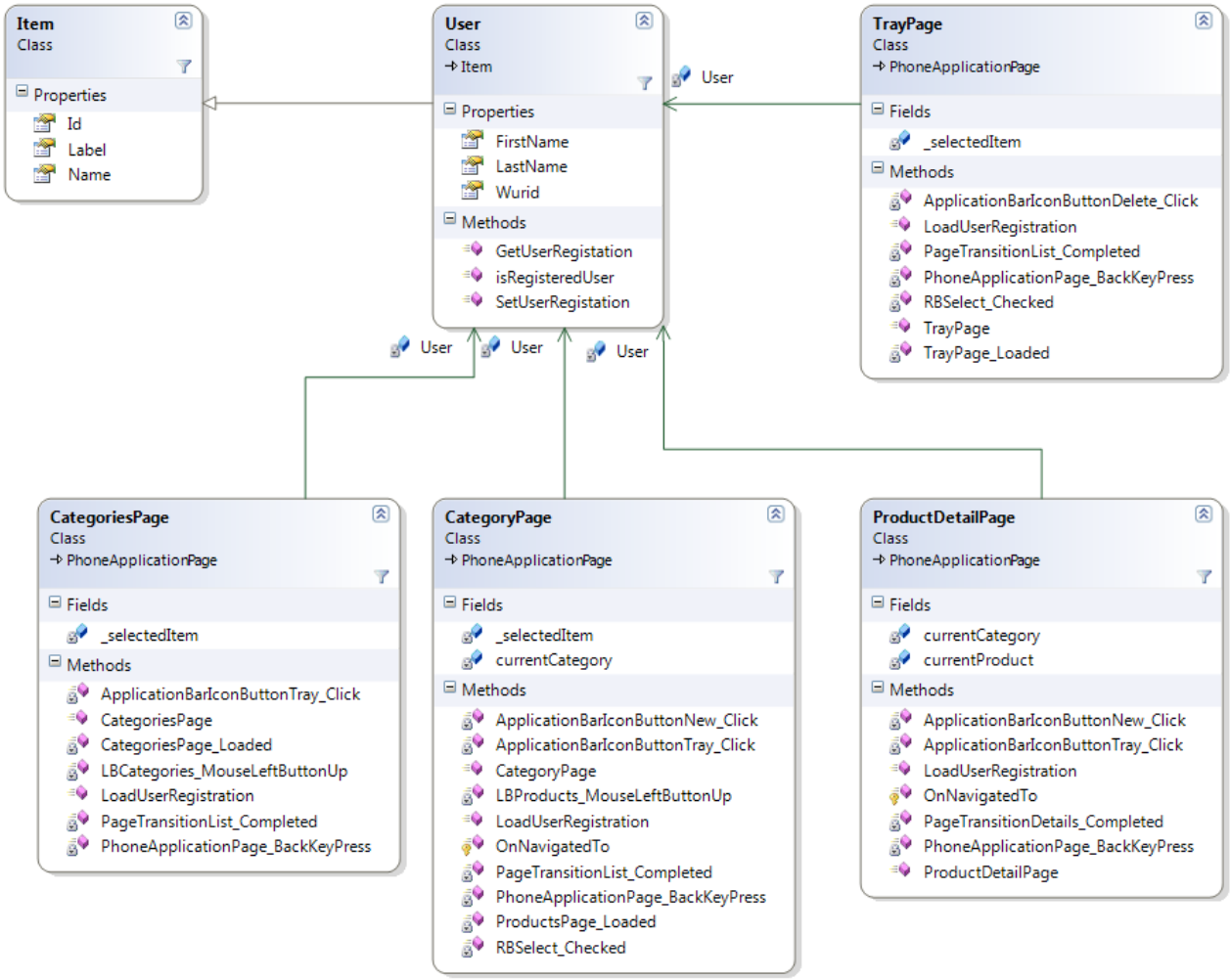


FIGURE 4.14 – Diagramme de classes des vues

fait car un des scénarios, à développer dans une future version, prévois que les conseils évoluent en fonction de chaque choix de l'utilisateur, ce qui impliquera de un recalcule du menu personnalisé.

Le diagramme de classes des modèles de présentation, figure 4.15, est proche des autres modèles développés dans notre prototype par les classes créées. Par contre, les relations entre elles sont plus forte car on utilise la composition de catégories dans un menu, de produits dans une catégorie et le plateau, de caractéristiques dans un produit. La conséquence est que la structure est moins souple mais correspond au modèle de navigation défini (cf. figure 4.12)

Par rapport à l'ontologie de départ, nous n'avons utilisé que les concepts nécessaires à l'application *Smartphone* comme *Category* et *Product*. Ces concepts vont correspondre directement à deux classes du même nom. Comme pour le module de la couche métier, une classe *Item* a été définie. Elle est composée de trois attributs dont *id* pour l'URI. L'attribut *label* de l'objet correspond à la chaîne de caractères, à afficher dans la langue souhaitée et *name* correspond à l'argument de l'URI. L'URI va permettre de lier les objets (catégorie, produit, caractéristique, unité, etc.) à travers l'architecture du système de connaissances jusqu'au *Smartphone*. De plus il permet d'identifier et de sélectionner des objets et ainsi de naviguer à travers les ressources (catégories, produits, etc.), tant au niveau de l'application *Smartphone* que du Web Service.

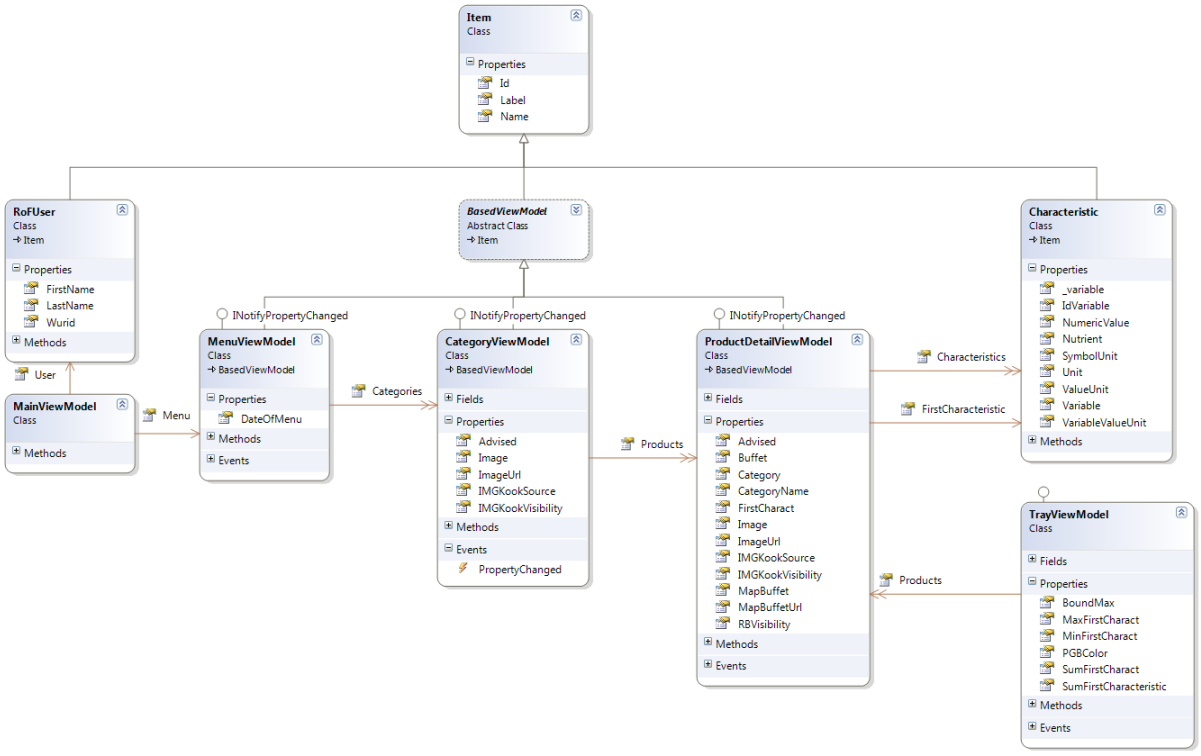


FIGURE 4.15 – Diagramme de classes de l'application Smartphone

Chapitre 5

Conclusion

A l'heure actuelle un environnement démontrant la faisabilité de l'approche est opérationnel sous forme de prototype et en cours d'expérimentation approfondie. Cet environnement intègre des composants ontologiques réutilisables, évolutifs et assemblables en fonction des besoins. Ils permettent de formaliser les deux grands domaines nécessaires au projet, à savoir :

- le contexte du Restaurant du Futur : les menus avec leurs produits, leurs caractéristiques et leurs valeurs nutritionnelles ;
- le consommateur : ses propriétés physiques et physiologiques, son de mode vie avec la description de son foyer et son type d'activité sportive, ainsi que ses objectifs diététiques personnels.

Un troisième composant ontologique a été conçu pour modéliser les recommandations du modèle de conseils. De nouveaux modules ontologiques sont en préparation. Ce travail a nécessiter d'aborder de nombreux sujets, au niveau de la conception d'ontologie, mais surtout celui du domaine pluridisciplinaire d'application : comportement du consommateur, nutrition, alimentation...

L'architecture informatique mise en place permet d'exploiter l'environnement de connaissances et de mettre à disposition des consommateurs des menus accompagnés de conseils alimentaires personnalisés en fonction du menu du Restaurant du Futur.

Une application *Smartphone* a été développée sur Windows Phone 7, tournant actuellement sur un émulateur, une des prochaines étapes est de la tester sur un appareil réel (date de sortie officielle octobre 2010). Mais dores et déjà, elle permet de tester plusieurs scénarios et les trois profils. Le système a déjà permis de rajouter des recommandations facilement et de faire évoluer la complexité des conseils.

Au terme de ce stage, nous pouvons considérer que les objectifs initiaux ont été atteints. Ils ont permis la rédaction d'un article, soumis en juillet, à une revue hollandaise dédiée à l'informatique pour l'agronomie.

Ce travail a été réalisé durant une période de cinq mois en Hollande, le premier mois a consisté en l'étude des besoins. Il a permis de proposer différents scénarios d'utilisa-

tion et de fonctionnalités de l'applications. Ce premier mois a conduit à la réalisation des trois profils de consommateur, qui ont été validés avec les experts en science du consommateur et de la nutrition lors de plusieurs présentations. Les deux mois suivants ont été consacrés à la modélisation des composants ontologiques, et à la constitution des jeux de test complets. Le mois suivant a été réservé à la conception et à l'implémentation du système à base de connaissances, de la base du module métier et à la mise en place du web service. Le dernier mois m'a permis de concevoir et d'implémenter l'application sur *Smartphone* et l'intégration des règles métiers au système. Le déroulement n'a pas été aussi linéaire, car la démarche a nécessité beaucoup d'aller retour entre les différents modèles.

Au niveau personnel, ce stage m'a permis de découvrir un autre pays, d'autres habitudes alimentaires, de vie et de pouvoir améliorer mon anglais. Au niveau professionnel, j'ai eu la chance d'intégrer une équipe de pointe dont les thématiques de recherches appliquées m'intéressent vraiment et qui ont déjà amené à des réalisations logicielles fonctionnelles. De plus, cela m'a permis de découvrir un autre mode de fonctionnement et de vision de la recherche en informatique appliquée. A cela s'ajoute que le domaine d'application correspond bien avec celui l'INRA, ou je travaille. Ce stage a permis de consolider des collaborations entre les deux équipes et qui vont pouvoir évoluer.

L'étape suivante est la réalisation d'un module avancé de conseils, avec mise en place d'un moyen permettant de combiner et donner la raison des différents conseils. En effet, plusieurs critères entrent en jeu (végétarien et calorie par exemple). Le module de réaction du système au choix du l'utilisateur, basé sur la géo-localisation des produits dans le restaurant, reste à faire évoluer. Au niveau scientifique, le modèle de conseils, actuellement en cours de création par les chercheurs en science du consommateur et de la nutrition, est à finaliser. L'interaction avec notre composant ontologique pour la représentation des recommandations sera alors a adapter. Au niveau de l'architecture, de futurs développements devront être prévus pour l'intégration de ce modèle de conseils. L'intégration devrait être facilitée par l'utilisation de la méthodologie objet, des patrons de conception qui apportent une souplesse à l'application.

Ce travail ouvre des perspectives de recherche par exemple, sur l'évolution dynamique des ontologies en fonction du comportement de l'utilisateur et de ses réactions aux conseils donnés, à différents pas de temps, y compris en temps réel. Une autre perspective a trait à l'amélioration de la communication entre les différents modèles : de connaissance, métier et consultation.

Bibliographie

Allaire J. (2002). Macromedia flash mx—a next-generation rich client. *Macromedia white paper*. Disponible à l'adresse http://download.macromedia.com/pub/flash/whitepapers/richclient.pdf.

Berners-Lee T., Hendler J. & Lassila O. (2001). The semantic web. *Scientific American*, **284**(5), 34–43.

Broy M., Deimel A., Henn J., Koskimies K., Plášil F., Pomberger G., Pree W., Stal M. & Szyperski C. (1998). What characterizes a (software) component ? *Software - Concepts & Tools*, **19**, 49–56. 10.1007/s003780050007.

Charlesworth A. (2009). The ascent of the smartphone. *Engineering & technology*, **4**(3), 32–33.

Fielding R. (2000). *Architectural Styles and the Design of Network-based Software Architectures*. Thèse de doctorat, University of California, Irvine. Disponible à l'adresse http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm.

Fielding R., Gettys J., Mogul J., Frystyk H., Masinter L., Leach P. & Berners-Lee T. (1999). Hypertext transfer protocol – http/1.1 (rfc 2616). Request For Comments. Available at `http://www.ietf.org/rfc/rfc2616.txt`, accessed 7 July 2006.

Gruber T. (2009). *Ontology*. Ling Liu and M. Tamer Özsu (Eds.), Springer-Verlag.

Köster E. (2009). Diversity in the determinants of food choice : A psychological perspective. *Food Quality and Preference*, **20**(2), 70 – 82. European Conference on Sensory Science of Food and Beverages 2006, European Conference on Sensory Science of Food and Beverages 2006. Disponible à l'adresse http://www.sciencedirect.com/science/article/B6T6T-4R5F002-2/2/9d36c0f3b95469e3d29887be9a49dfb9.

Le Ber F., Lieber J. & Napoli A. (2006). Les systèmes à base de connaissances. Dans *Encyclopédie de l'informatique et des systèmes d'information*, réd. J. Akoka & I. Comyn Wattiau, pp. 1197–1208. Vuibert. Disponible à l'adresse http://hal.inria.fr/inria-00201566/PDF/sbc-vuibert-06.pdf.

Manola F. & Miller E., (Réd.) (2004). *RDF Primer*. W3C Recommendation. World Wide Web Consortium. Disponible à l'adresse : http://www.w3.org/TR/rdf-primer/.

Mcguinness D.L. & van Harmelen F. (2004). OWL web ontology language overview. W3C recommendation, W3C. Disponible à l'adresse : http://www.w3.org/TR/owl-features/.

Noy N. & Mcguinness D. (2001). Ontology development 101 : A guide to creating your first ontology. Rap. Tech..

Noy N. & Rector A. (2006). Defining n-ary relations on the semantic web. *W3C Working Group Note*. Disponible à l'adresse http://www.w3.org/TR/swbp-n-aryRelations.

Odell J. (1998). Six different kinds of composition. *Advanced Object-Oriented Analysis and Design Using UML*. Disponible à l'adresse http://www.conradbock.org/compkind.html.

Prud'hommeaux E. & Seaborne A. (2008). SPARQL Query Language for RDF. W3C Recommendation. Disponible à l'adresse : http://www.w3.org/TR/rdf-sparql-query/.

Psyché V., Mendes O. & Bourdeau J. (2003). Apport de l'ingénierie ontologique aux environnements de formation à distance. *Revue des Sciences et Technologies de l'Information et de la Communication pour l'Education et la Formation (STICEF)*, **10**, 89–126. Disponible à l'adresse http://sticef.univ-lemans.fr/num/vol2003/psyche-06s/sticef_2003_psyche_06s.htm.

Rector A. & Welty C. (2005). Simple part-whole relations in owl ontologies. *W3C Working Draft*. Disponible à l'adresse http://www.w3.org/2001/sw/BestPractices/OEP/SimplePartWhole/index.html.

Rijgersberg H., Wigham M. & Top J. (2010). How semantics can improve engineering processes : A case of units of measure and quantities. *Advanced Engineering Informatics*. Accepted.

Segaran T., Taylor J. & Evans C. (2009). *Programming the Semantic Web*. O'Reilly, Cambridge, MA.

Stubenitsky K., Aaron J., Catt S. & Mela D. (2000). The influence of recipe modification and nutritional information on restaurant food acceptance and macronutrient intakes. *Public Health Nutrition*, **3**(02), 201–209.

Annexes

Annexe A

Questionnaire pour les consommateurs du RoF

Modify Survey

Modify Survey

The modify survey page contains all the functions related to Survey Design.

Page Help

Text Replacement:

Copy the text token from the below list, paste the copied text into the desired location in the survey.

- User Data Tokens
- Email Data Tokens
- Hidden Data Tokens
- Item Answer Tokens

Print Survey

Preview This Survey

Go To Survey List

This survey is currently LOCKED for editing by Anne Tireau. It will become available after they unlock it or after 30 minutes has passed with no activity. Click here to unlock and exit the survey

This survey has at least one response. Only limited changes are allowed on a survey with responses. To fully edit this survey, you must first delete all of the responses, which can be done by clicking here.

RvdT VU Studenten (Huidige enquete) edit

[No Title Entered]

edit page properties | move page

Welcome to the intake survey of the Restaurant of the Future.

The survey's length is approximately 60 questions. All questions must be answered. Per question only a single answer is allowed.

Your first impression is usually best. Don't overthink your answer, there are no good or bad answers.

edit | move | pipe

[No Title Entered]

edit page properties | move page

1. Given Access Code* edit | move | pipe

2. First name* edit | move | pipe

3. Last name* edit | move | pipe

4. Year of birth* edit | move | pipe

[No Title Entered]

edit page properties | move page

5. Gender*
   Male
   edit | move | pipe

6. Length (in cm)* edit | move | pipe

7. Weight (in kg)* edit | move | pipe

[No Title Entered]

edit page properties | move page

8. Work e-mail address (firstname.lastname@wur.nl if employed at WUR)* edit | move | pipe

9. Where are you employed?* edit | move | pipe
   - WUR others (Administration, FB, staff)
   - AFSG
   - PSG
   - ASG
   - ESG

10. What is your highest level of education?* edit | move | pipe
    -- Please Select --

[No Title Entered]

edit page properties | move page

11. What is your household type?* edit | move | pipe
    - single
    - living together without children
    - living together, children out of home
    - with children: (partly) at home, youngest under 9
    - with children: partly at home, youngest > 9 year
    - other

12. What are the first four figures of your zip code (home address)?* edit | move | pipe

13. How do you usually travel to work?* edit | move | pipe
    -- Please Select --

[No Title Entered]

edit page properties | move page

14. How often do you do sports (weekly average)?* edit | move | pipe
    -- Please Select --

15. In the last 3 years, where did you travel to (privately and professionally)?*

-- Please Select --

edit move pipe

[No Title Entered]

edit page properties | move page

16. Where do you, or somebody else in your household, go shopping (food products)?*

edit move pipe

	always	often	regularly	rarely	never
Supermarket					
Smaller shops					
Internet					
Market					
Petrol stations, etc					
Eating out of home					

17. How often do you buy organic and/or fair trade products?*

edit move pipe

- always
- often
- regularly
- rarely
- never

[No Title Entered]

edit page properties | move page

18. What is your reaction to the following statements:*

edit move pipe

	Disagree strongle	Disagree moderately	Disagree slightly	Neither disagree nor agree	Agree slightly	Agree moderately	Agree strongly
I am constantly sampling new and different foods.							
I don't trust new foods.							
If I don't know what is in a food, I won't try it.							
I like foods from different countries.							
Ethnic food looks too weird to eat.							
At dinner parties, I will try new food.							
I am afraid to eat things I have never had before.							
I am very particular about the foods I will eat.							
I will eat almost anything.							
I like to try new ethnic restaurants.							

[No Title Entered]

edit page properties | move page

19. Do you have the desire to eat when you are irritated?*

edit move pipe

Never Rarely Sometimes Often Very often

20. If food tastes good to you, do you eat more than usual?*

edit move pipe

Never Rarely Sometimes Often Very often

21. Do you have a desire to eat when you have nothing to do?*

edit move pipe

Never Rarely Sometimes Often Very often

22. If you have put on weight, do you eat less than you usually do?*

edit move pipe

Never Rarely Sometimes Often Very often

[No Title Entered]

edit page properties | move page

23. Do you have a desire to eat when you are depressed or discouraged?*

edit move pipe

Never Rarely Sometimes Often Very often

24. If food smells and looks good, do you eat more than usual?*

edit move pipe

Never Rarely Sometimes Often Very often

25. How often do you refuse food or drink offered because you are concerned about your weight?*

edit move pipe

Never Rarely Sometimes Often Very often

26. Do you have a desire to eat when you are feeling lonely?*

edit move pipe

Never Rarely Sometimes Often Very often

[No Title Entered]

edit page properties | move page

27. If you see or smell something delicious, do you have a desire to eat it?*

edit move pipe

Never Rarely Sometimes Often Very often

28. Do you have a desire to eat when somebody lets you down?*

edit move pipe

Never Rarely Sometimes Often Very often

29. Do you try to eat less at mealtimes than you would like to eat?*

edit move pipe

Never Rarely Sometimes Often Very often

30. If you have something delicious to eat, do you eat it straight away?*
Never Rarely Sometimes Often Very often

edit move pipe

[No Title Entered]

edit page properties move page

31. Do you have a desire to eat when you are cross?*
Never Rarely Sometimes Often Very often

edit move pipe

32. Do you watch exactly what you eat?*
Never Rarely Sometimes Often Very often

edit move pipe

33. If you walk past the baker, do you have the desire to buy something delicious?*
Never Rarely Sometimes Often Very often

edit move pipe

34. Do you have a desire to eat when you are approaching something unpleasant to happen?*
Never Rarely Sometimes Often Very often

edit move pipe

[No Title Entered]

edit page properties move page

35. Do you deliberately eat foods that are slimming?*
Never Rarely Sometimes Often Very often

edit move pipe

36. If you see others eating, do you also have the desire to eat?*
Never Rarely Sometimes Often Very often

edit move pipe

37. When you have eaten too much, do you eat less than usual the following days?*
Never Rarely Sometimes Often Very often

edit move pipe

38. Do you get the desire to eat when you are anxious, worried or tense?*
Never Rarely Sometimes Often Very often

edit move pipe

[No Title Entered]

edit page properties move page

39. Do you find it hard to resist eating delicious foods?*
Never Rarely Sometimes Often Very often

edit move pipe

40. Do you deliberately eat less in order not to become heavier?*
Never Rarely Sometimes Often Very often

edit move pipe

41. Do you have a desire to eat when things are going against you or when things have gone wrong?*
Never Rarely Sometimes Often Very often

edit move pipe

42. If you walk past a snack bar or a café, do you have the desire to buy something delicious?*
Never Rarely Sometimes Often Very often

edit move pipe

[No Title Entered]

edit page properties move page

43. Do you have the desire to eat when you are emotionally upset?*
Never Rarely Sometimes Often Very often

edit move pipe

44. How often do you try not to eat between meals because you are watching your weight?*
Never Rarely Sometimes Often Very often

edit move pipe

45. Do you eat more than usual, when you see others eating?*
Never Rarely Sometimes Often Very often

edit move pipe

46. Do you have a desire to eat when you are bored or restless?*
Never Rarely Sometimes Often Very often

edit move pipe

[No Title Entered]

edit page properties move page

47. How often in the evening do you try not to eat because you are watching your weight?*
Never Rarely Sometimes Often Very often

edit move pipe

48. Do you have a desire to eat when you are frightened?*
Never Rarely Sometimes Often Very often
edit move pipe

49. Do you take into account your weight with what you eat?*
Never Rarely Sometimes Often Very often
edit move pipe

50. Do you have a desire to eat when you are disappointed?*
Never Rarely Sometimes Often Very often
edit move pipe

[No Title Entered]

edit page properties | move page

51. When you are preparing a meal are you inclined to eat something?*
Never Rarely Sometimes Often Very often
edit move pipe

Close Done

Annexe B

Analyse : scénarios et domaines des ontologies

The ontologies

Anne Tireau

March, 29th 2010

Table des matières

Chapitre 1

Some scenarios for the application

1.1 Scenario 1 : John the dieter

John is a male, 50 years old and 1.80 meter tall (Fig. 1.1). He works in the Information Management team, in front of his screen all the day and doesn't travel a lot. He's a regular customer of the Restaurant of the Future (RoF), since he eats at least 4 days per week. He never comes at the restaurant with an external food. He has rarely working lunch, but he eats often with the members of his team. John isn't a sportsman and he lives too far from his work to come with a bike, so he comes by car. His BMI is about 30, so he has to loose weight and that's why he has a particular less calorie diet. The problem is that John is greedy and is used to choose sweet course, he always takes a dessert to finish his lunch.

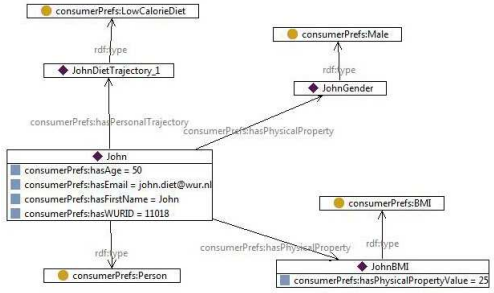


FIGURE 1.1 – John's case

During the week John's eaten at the Restaurant of the Future. We are Wednesday and so we know what he ate the last 2 days (Tab. 1.1). We also know of what his lunch is composed from the beginning of his coming in the RoF.

TABLE 1.1 – John's lunch

	Monday	Tuesday
Juice & Smoothies	Milk & fruit	Milk & fruit
Freshmade Soup	Soep Bospaddenstoelen	Soep chinese groenten
Bread & Bake-offs	Bol zacht wit	Bol zacht wit
Hot meals & snacks	Bami oriental	Frikandel
Dessert & Juices	Mono kwark aardbei Perz- Marac	Mono dessert
Coffee & Tea	cappuccino	

1.2 Scenario 2 : Wim the sportsman

Wim is a male, 26 years old and he's 1.92 meter tall. He works in the Vision team from 1 year now. His work is stressful because he has just begun a state-of-the-art project with a lot of pressure. But Wim like competition. That's why he has nevertheless chosen to work in part time (70%) since he can devote a lot of time to Football, his preferred sport. Thus Wim plays in the amateur championship. This year his team has every chance to win the championship which will begin in 2 months only. So, he has doubled all his training. Of course he also continues to come by bicycle to his work : 60 km every days. And to take no chances, he's decided to respect a particular sportive diet, with the help of RoFAdviseApps, to be in great shape during the championship ! Currently Wim don't have any health problem. He's not particularly difficult upon food, he likes snack food, because rapidly eaten and not expansive. He doesn't like big or official lunch.

From the beginning of the week Wim has eaten at the Restaurant of the Future. We are Wednesday and so we know what he ate the last 2 days (Tab. 1.2). We also know of what his lunch is composed from the beginning of his coming in the RoF.

1.3 Scenario 3 : Esther the environmentalist

Esther is a female, 39 years old and he's 1.72 meter tall. She is slim and in good health. The only problem is that she's tomato intolerance. What is really important for her is "to save the world", she's a humanist person and so, she wants the most she could eat organic, fair-trade or animal-friendly food. She is very carefully of what her eat. She works with John and come every day to the RoF. She often comes with her own food,

TABLE 1.2 – Wim's lunch

	Monday	Tuesday
Juice & Smoothies	Melk 1/4 liter	Melk 1/4 liter
Salads	Rauwkost/ saladebar middel	
Hot meals & snacks	Broodje hotdog	Pizza Turkse
Dessert & Juices		Banaan
Lemonades		Sprite flesje

drink most often (fruit juice or warm tea). Like a lot of women she loves chocolate and she often gives in the temptation to take chocolate dessert or warm chocolate at the end of the lunch, even if she has eaten one's fill. She makes a bit sports the weekend (hike in the nature) and she comes every days with her bicycle (against the pollution of course).

During the week Esther's eaten at the Restaurant of the Future. We are Wednesday and so we know what she ate the last 2 days (Tab. 1.3). We also know of what her lunch is composed from the beginning of her coming in the RoF.

TABLE 1.3 – Esther's lunch

	Monday	Tuesday
Juice & Smoothies		Tiramisu
Salads	Rauwkost/ saladebar middel	
Freshmade Soup		Soep Courgette
Hot meals & snacks	Zalmmoot	Omelet Met Kaas
Dessert & Juices	Dessert 1,30	
Coffee & Tea		cappuccino

1.4 Feedback scenarios

From information about the person and the RoF's menu we want to advise our consumer about food that could fit their. There are many solutions to advise them : by email, web or Smartphone. There also different solution to access to the information.

Means of communication

E-mail solution John is at his office and receives an e-mail around 11 o'clock. This e-mail contains a selection and a advise of foods for him (Fig. 1.2). John can print or just notice the e-mail advise and go to the RoF for the lunch.

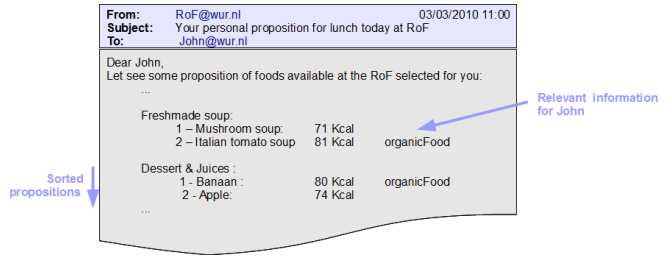


FIGURE 1.2 – John's e-mail

Web solution John is at his office and log on his count at the RoF Advise Website. Maybe he received an e-mail to inform him that he can connect to it. Then he can navigate through the product available at the RoF. The products are sorted for him. He has access to a predefine selection of food and information about their, but also to all the other food available today.

Web/Wap solution for mobile The early solution could have or be a mobile phone version and be used at the RoF.

Smartphone'Apps solution John synchronize his Smartphone to update the RoF Advise Apps at his office or with Wifi/3G access at the RoF. He can use it at the RoF.

Wednesday scenarios

For the different solution, we choose Smartphone representation."

Preselected product solution The food are selected and sorted according :
- the context of the lunch (mainly the daily menu of the RoF),
- John's preferences (what John's likes and wants),
- John's profile (John's nutritive needs).

Question : What do we propose a menu (link between courses) or a products per category ? In this document we consider that for each category we can advise one or more product.

After update his Smartphone John can watch on his Smartphone what are RoFAdviseApps proposes for him today. It could be at his office or at the RoF. He can navigate through the RoF's categories (Fig. 1.3) and looking for informations about foods and

make his final choice.

FIGURE 1.3 – navigation by Smartphone's Categories

When John explore one category, he can watch some product which are selected and sorted by the system. For each product he can access to complementary and relevant information for him (Fig. 1.4).

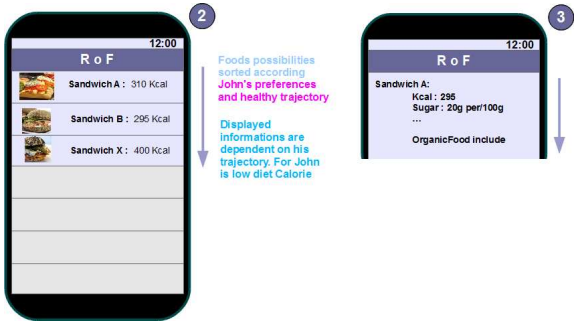


FIGURE 1.4 – selected and sorted products and more details

Dynamic alternatives solution In the same way that the early solution, John needs to update his Smartphone's RoFAdviseApps and maybe to have a Wifi/3G access (depending of the technology chosen). Then at his office or at the RoF he can consult he's

Smartphone and search what he wants to eat today. When he selects and item, the system dynamically propose to him alternatives. These alternatives are selected and sorted by the system according his preferences (Fig. 1.5).

FIGURE 1.5 – Others alternatives

Dynamic advice solution : My Tray Like the concept of basket in a Web site store, John can in the same way adds product on his virtual and real tray. From this he can access to a summarize of his lunch. The proposition of the system is more dynamic, because another criteria is what he has already chose. So, John can juggle with the product to compose and make compromise between his choice to build his lunch... (if he has the time to). (Fig. 1.6)

Information displayed

Like food, the informations about food and their granularity are selected for John. Not all the information are displayed by the system. There also are many means to display it. It could be John's choice or determined by his knowledge about food (nutrition, traceability, etc.). We could propose adding to raw data symbols to help (Fig. 1.7)

1.4 Feedback scenarios

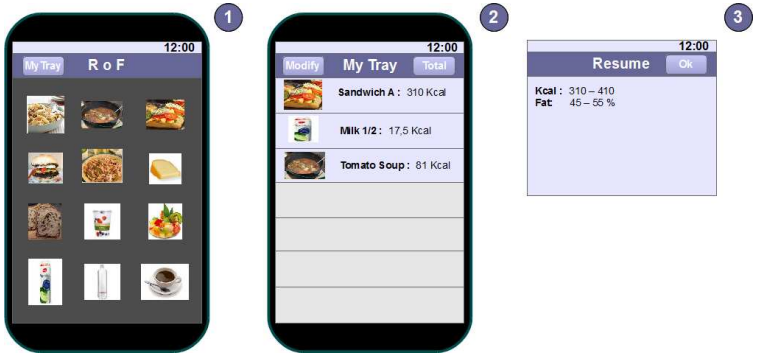


FIGURE 1.6 – My Tray

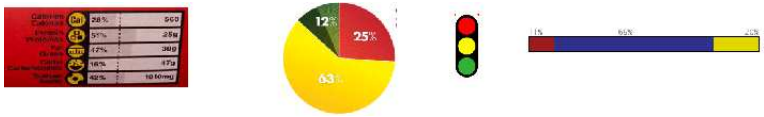


FIGURE 1.7 – raw data displaying

Chapitre 2

Determine the domain of the ontologies

2.1 Personal property

- What physical properties have the consumer ? (age, size, BMI...)
- What are his Biophysical properties ? (about oro-gastro intestinal physiology)
- What is his Genetic background ? (Weight problem family...)

2.2 Consumer household

- How many people are in his household ? age ? gender ?
- Who go to shopping for food ?
- Who is cooking ?

2.3 Consumer activity

- Does he work a lot ? (Measurement ?)
- How many hours per day does he work ?

- What does he do the weekend ? Sports ? Cooking ? Activities ?
- How many parties/dinner does he go by month ?

- Does he do sports ?
- What kind of sports ? Frequency ? Level ? Duration ?
- How does he come to work ?
- How many kilometre ? Frequency ?

2.4 Personal trajectory about food

- What personal trajectory has the consumer ?
  - Diet trajectory ?
    - Weight : lost ? Keep stable ? Get weight ?
    - Sodium : low ? Balance ?
    - → All component, nutriment, vitamins... (fat, sugar, salt, B...) treatment/diet
    - Sportive : high ? less ?
    - Vegetarian : what kind ?
  - Economic trajectory ?
    - Fair trade ?
    - Cheap ?
  - Environmental trajectory ?
    - Animal friendly ?
    - Organic ?
- Why this reason ?
  - Aesthetic
  - Health (feeling better...) / Illness (allergy, diabetes, anorexia, joint, backache...)
  - Religion
  - Personal conviction
  - Activity (sports...)
  - Social relation (family, work...)

2.5 Consumer behaviour

- What are his Personal knowledge about food ?
- What are his previous experiences or history in diet and food ?

- What is his type of cognition's kind ? (looks, heard, kinesics sensibility)

2.6 Consumer with food / habits

- How important is food / eating for the consumer ?
- Does he like eating ?

- How long is his lunch time during the week ?
- How long is his lunch time during the weekend ?
- Does he miss lunch time ? What is the frequency of this ?

- What is the composition of his lunch during the week ?

- Does he often drink wine, beer or an other alcoholic drink ?
- How many course of a meal has one lunch ?
- Does he take a coffee, tea or chocolate after lunch ?

- Is he fond of good food ?
- What is more important quality or quantity ?
- How much does he eat ?

- What taste does he prefer ? Sweet ? Salt ? Hot ? Spicy ? ...
- Does exist food that he unconditional loves or hates ?

- How does he choose his food : by smell, look... ?
- Which information does he need ? (organic...)
- Does he want eating a food if it looks good or if someone eats it ?
- Does he like trying new food ?
- Does he like vary his menu ?

- Does he eat without being hungry ?
- Does emotion/feel make he hungry or satiate ?
- Does he feel satiated when his meal was finished ?

- How many time per week does he go to the RoF ?

2.7 At the restaurant

- What does he already have on his tray ?
- Does he see all alternative food in the restaurant ?
- Is he hungry ?
- How many people are with him ?
- Is it a working lunch with negotiation... ?
- Is it a calm lunch with good colleagues ?
- How many time has he to lunch ?
- Does he have a drink in this tray ?
- Does he bring his own lunch ? What is it ? For what kind of course ?

2.8 Food

- In what category is it ? (Fruit...)
- What are their nutritive properties ?
- It is newer in the menu ? Special menu ? (Christmas...)

- With what does it combine well ?
- What is its origin ? (Foreign speciality...)
- What is its traceability ? (Fair trade)
- How does it look ? smell ?
- Is a photo of it available ?

2.9 Type of Feedback

- What kind of feedback / information does he need ?
- Does he provide information on where to find the food ?
- Does he need alternatives ?
- If he just need raw information about food ? What kind ?
- Have he need to access his lunch history ?
- Does he prefer receive an email with the menu and/or proposition before going to lunch ?
- Does he need an interactive navigation among the food's information ?

Chapitre 3

Important terms in the ontology

3.1 About Food

This concept could represent one of the final product of the restaurant (for example : Italian tomato soup).

- **Product** or **Food** (name of each course available in the RoF ? in english+dutch or only in dutch ?)

- **Ingredient** : composition of the Product (tomato, salmon, cumin...), all the name ? availability ?
- **Origin** of ingredient, product. This is information about traceability.
- **Category** of the ingredients (vegetable, fruit, meat, fish...)

- **Nutritive** properties : Energy (unit), Fat (unit), Vitamin, etc.
- **Appearance** of the Product / Presentation (look, smell, color, etc.)
- **Recipe** properties : temperature, fried food,...

- **Quantity**/Size of the product : per 100g, percent (unit ?)
- Category of the product(just in the RoF or maybe larger ?)
- **Cost** of the product

3.2 About Context

This context could be about the context of the lunch.

- Material context :
  - Menu
  - Restaurant of the Future environment : description of the room (table, color of sit...), of the disposition of the plat, staff, sound, etc.

  - Weather : temperature, light...,
  - Time/date of the of the lunch ;
- Human context :
  - description of the social context : how many people eat with him, relax/work lunch,
  - description of the time he has for the lunch ;

3.3 About objective/trajectory

What does the consumer want ? What is the objective of the system/diet ?

- Diet : vegetarian, sportive, restrict, variability of course, discovering, etc.
- Economic : fair trade, cheap/expansive
- Environment : animal friendly, organic
- Taste/Sensitive : Quality/Quantity
- Reason : Religion, Health, Aesthetic, Activity, Social...
- Constraint (strength of) / Importance of the criteria
- Date

3.4 About Preference

What does the consumer like (and would want) ?

- Food
- Context
- Objective

3.5 About Profile

The "Profile" could be composed of informations about the consumer and a nutrition profile, maybe view like expert rule.
For example :

→ a sportive person, female, 30 years old, 1.70 m needs 450 Kcal per lunch ;
→ a person with BMI>35 needs to follow a less calorie diet ;

- Type : Sportive, BMI >, BMI <
- Needs : Energy (unit), Fat (unit),

3.6 About Person

- Identity properties : first-name, last-name, sofinumber
- Work properties : employer, address, telephone number, email, WUR-card number
- Household properties : type : single, together without children... ;
- Health properties : illness
- Physical property gender, age/birth, weight (unit), length (unit), BMI (unit, label)
- Choice behaviour : what could influence him (food, context...)
- Genetic background
- Habits
- Activity

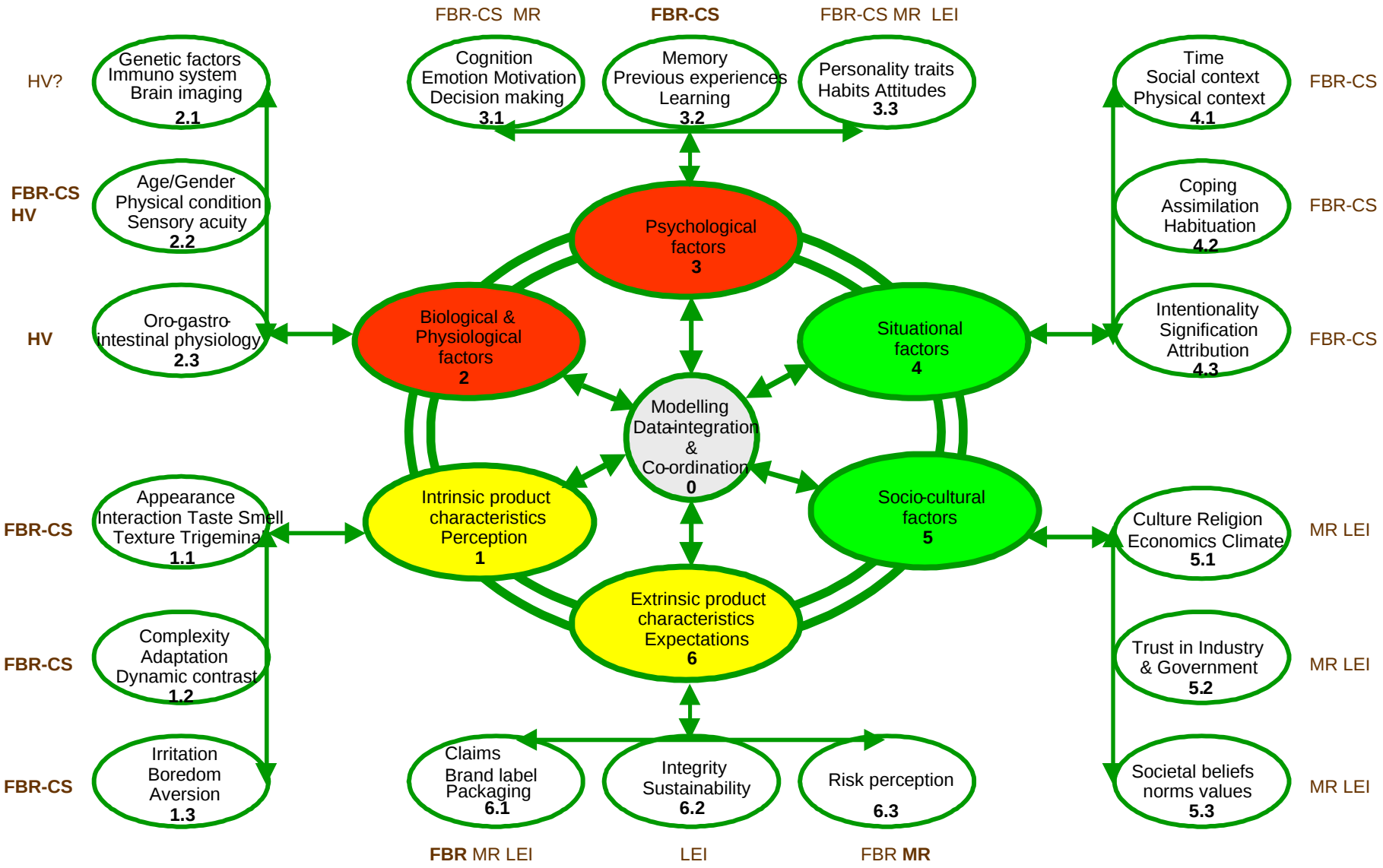
3.7 About TypeOfFeedback

- Food information needed
- Type
  - Communication means : email, web, iPhone
  - User-friendliness : text, numerical, color, traffic light, percent, graph

Annexe C

Facteurs essentiels du choix d'aliments

Essential factors that influence eating and drinking behaviour and food choice



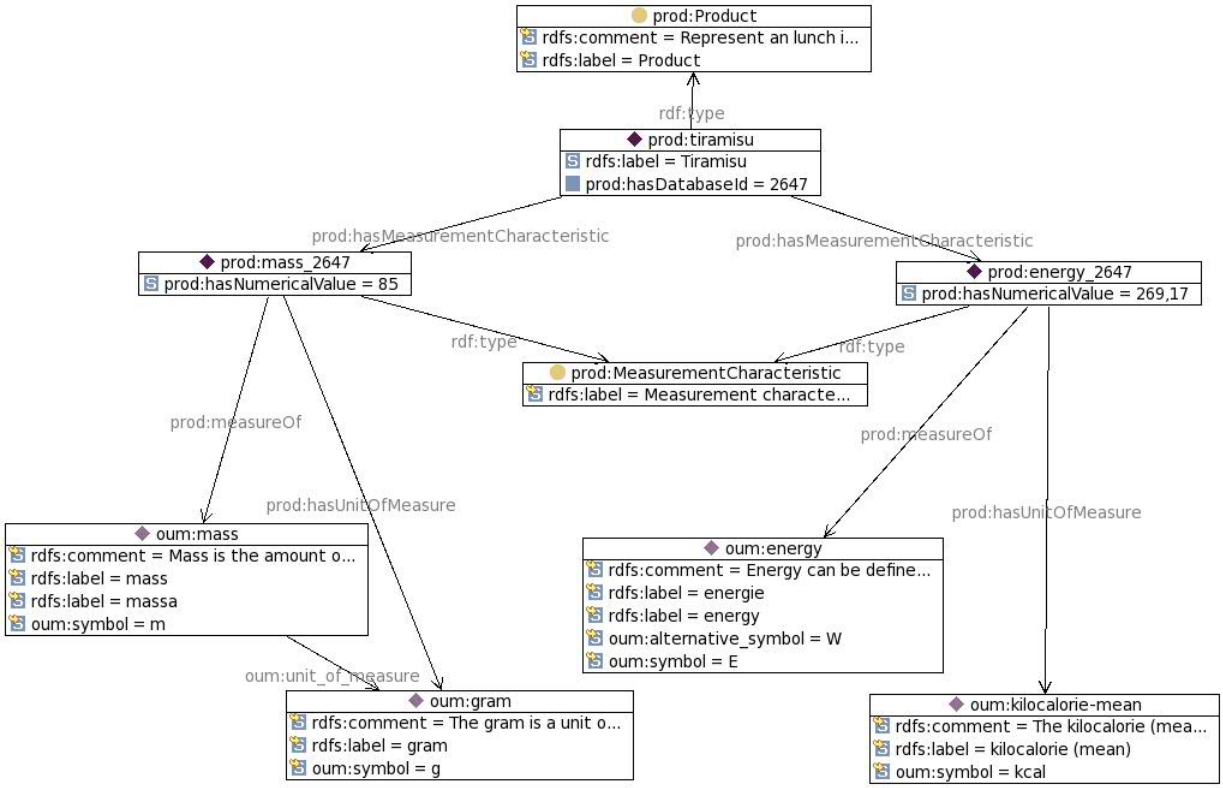


FIGURE C.1 – Représentation de la masse d'une part de tiramisu avec TopBraid

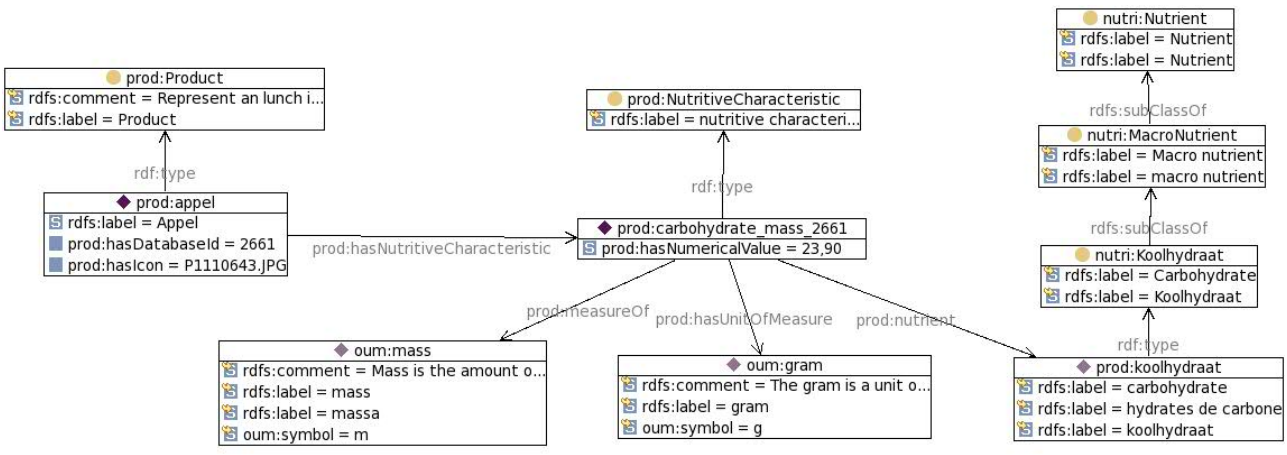


FIGURE C.2 – Représentation de la masse d'hydrate de carbone d'une pomme avec TopBraid

Annexe D

Mode de vie du consommateur

.

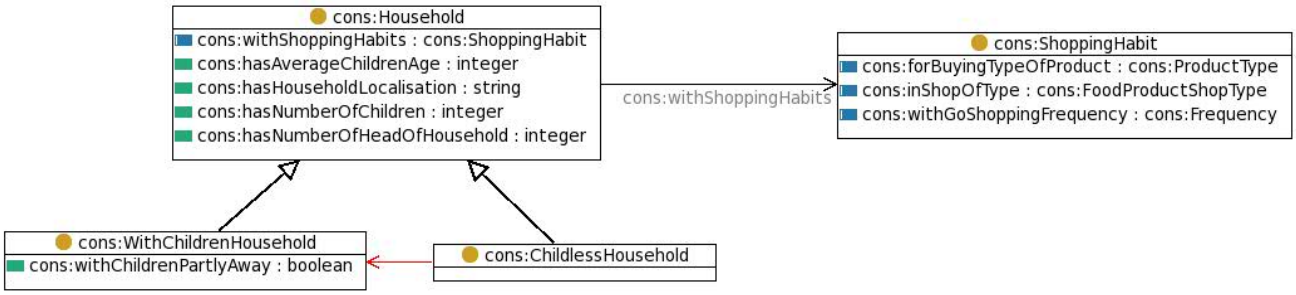


FIGURE D.1 – Représentation du concept *Household* avec TopBraid

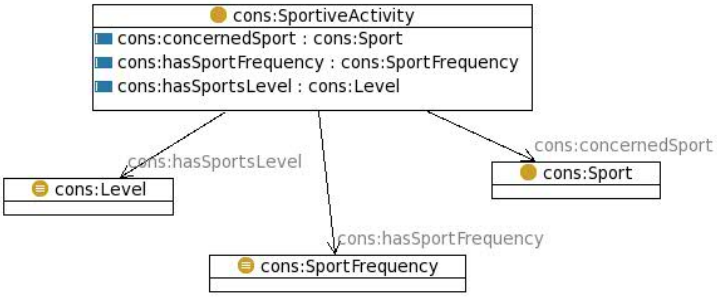


FIGURE D.2 – Représentation du concept *SportiveActivity* avec TopBraid

Annexe E

Concept de *MeasurementCharacteristic*

Listing E.1 – Définition de la propriété fonctionnelle *measureOf* en code Turtle

```
meas:measureOf
      a       owl:FunctionalProperty , owl:ObjectProperty ;
      rdfs:domain
              [ a       owl:Class ;
                owl:unionOf (<http://www.wurvoc.org/rof/advice#TypeOfMeasurement>
                    meas:MeasurementCharacteristic)
              ] ;
      rdfs:label "measure of"@en ;
      rdfs:range oum:Metrological_concept .
```

Listing E.2 – Définition de la propriété fonctionnelle *hasUnitOfMeasure* en code Turtle

```
meas:hasUnitOfMeasure
      a       owl:FunctionalProperty , owl:ObjectProperty ;
      rdfs:domain meas:MeasurementCharacteristic ;
      rdfs:label "has unit"@en ;
      rdfs:range oum:Unit_of_measure .
```

Annexe F

Turtle code of *Consumer*

```
cons:Consumer
      a       owl:Class ;
      rdfs:comment "Person who is a consumer of the Restaurant of the Future"@en ;
      rdfs:label "Consumer"^^xsd:string ;
      rdfs:subClassOf owl:Thing ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:cardinality "1"^^xsd:nonNegativeInteger ;
                owl:onProperty cons:hasWURCardNumber
              ] ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:cardinality "1"^^xsd:nonNegativeInteger ;
                owl:onProperty cons:hasLastName
              ] ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:cardinality "1"^^xsd:nonNegativeInteger ;
                owl:onProperty cons:hasFirstName
              ] ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:cardinality "1"^^xsd:nonNegativeInteger ;
                owl:onProperty cons:hasGender
              ] ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:cardinality "1"^^xsd:nonNegativeInteger ;
                owl:onProperty cons:hasHeight
              ] ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:allValuesFrom cons:Height ;
                owl:onProperty cons:hasHeight
              ] ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
                owl:minCardinality "1"^^xsd:nonNegativeInteger ;
                owl:onProperty cons:hasWeight
              ] ;
      rdfs:subClassOf
              [ a       owl:Restriction ;
```

```
                owl:allValuesFrom cons:Weight ;
                owl:onProperty cons:hasWeight
              ] .
```

Annexe G

Infrastructure des données

G.1 Infrastructure des produits

Productparameters

SCOMP0056\RvdT_DataModel

Data-model

SCOMP0056\RvdT_RawData

Backup-protocol

Data-bronnen geborgd

- *PersoonID*
- *producten*
- *prijs*
- *tijdstip van afrekenen*
- *...*

- *ProductID*
- *nutrienten*
- *ingredienten*
- *kcal*
- *...*

- *Weeknummer*
- *Menuitems (aanbod)*

Sync-protocol

Data-bronnen

Kassa DB

Sodexo DB?

Sodexo menu-cyclus DB?

G.2 Infrastructure des produits

Persoonsparameters

SCOMP0056\RvdT_DataModel

Data-model

SCOMP0056\RvdT_RawData

Backup-protocol

Data-bronnen geborgd

Persoon:
- *naam*
- *leeftijd*
- *opgegeven gewicht*
- *lengte*
- *geslacht*

Persoon:
- *naam*
- *e-mail*
- *pasnummer*

Persoon:
- *naam*
- *e-mail*
- *WUR-card nummer*

Gewicht:
- *tijdstip*
- *gewicht*

Sync-pro

Data-bronnen

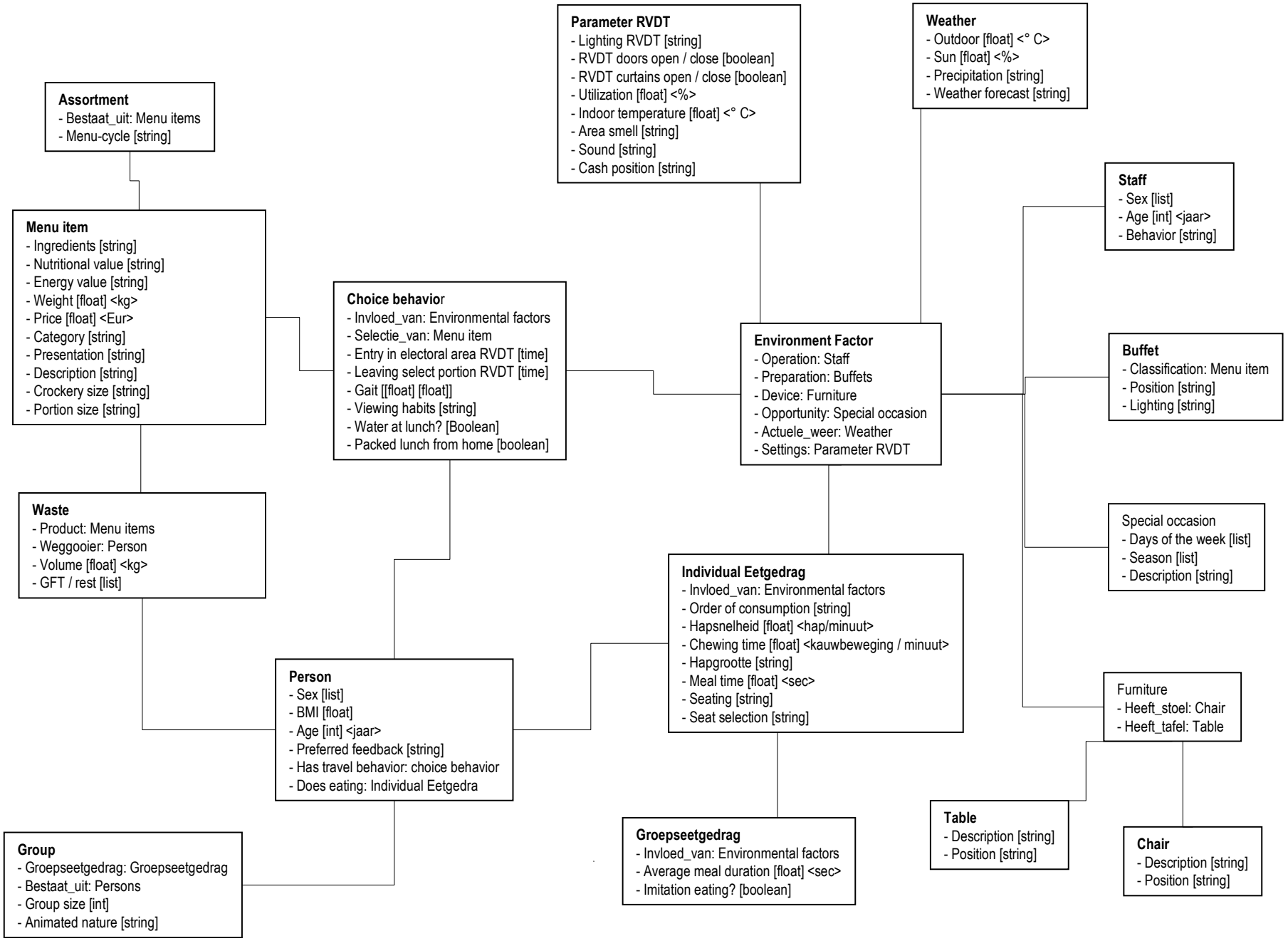
Internetenquête database

Acces DB met persoon-pas-koppeling

Wageningen Basis Administratie

Weegschaal

G.3 Infrastructure des données intégrée


Assortment
- Bestaat_uit: Menu items
- Menu-cycle [string]
Parameter RVDT
- Lighting RVDT [string]
- RVDT doors open / close [boolean]
- RVDT curtains open / close [boolean]
- Utilization [float] <%>
- Indoor temperature [float] <° C>
- Area smell [string]
- Sound [string]
- Cash position [string]
Weather
- Outdoor [float] <° C>
- Sun [float] <%>
- Precipitation [string]
- Weather forecast [string]
Staff
- Sex [list]
- Age [int] <jaar>
- Behavior [string]
Menu item
- Ingredients [string]
- Nutritional value [string]
- Energy value [string]
- Weight [float] <kg>
- Price [float] <Eur>
- Category [string]
- Presentation [string]
- Description [string]
- Crockery size [string]
- Portion size [string]
Choice behavior
- Invloed_van: Environmental factors
- Selectie_van: Menu item
- Entry in electoral area RVDT [time]
- Leaving select portion RVDT [time]
- Gait [[float] [float]]
- Viewing habits [string]
- Water at lunch? [Boolean]
- Packed lunch from home [boolean]
Environment Factor
- Operation: Staff
- Preparation: Buffets
- Device: Furniture
- Opportunity: Special occasion
- Actuele_weer: Weather
- Settings: Parameter RVDT
Buffet
- Classification: Menu item
- Position [string]
- Lighting [string]
Waste
- Product: Menu items
- Weggooier: Person
- Volume [float] <kg>
- GFT / rest [list]
Special occasion
- Days of the week [list]
- Season [list]
- Description [string]
Individual Eetgedrag
- Invloed_van: Environmental factors
- Order of consumption [string]
- Hapsnelheid [float] <hap/minuut>
- Chewing time [float] <kauwbeweging / minuut>
- Hapgrootte [string]
- Meal time [float] <sec>
- Seating [string]
- Seat selection [string]
Person
- Sex [list]
- BMI [float]
- Age [int] <jaar>
- Preferred feedback [string]
- Has travel behavior: choice behavior
- Does eating: Individual Eetgedra
Furniture
- Heeft_stoel: Chair
- Heeft_tafel: Table
Group
- Groepseetgedrag: Groepseetgedrag
- Bestaat_uit: Persons
- Group size [int]
- Animated nature [string]
Groepseetgedrag
- Invloed_van: Environmental factors
- Average meal duration [float] <sec>
- Imitation eating? [boolean]
Table
- Description [string]
- Position [string]
Chair
- Description [string]
- Position [string]

Annexe H

Vocabulaire partagé

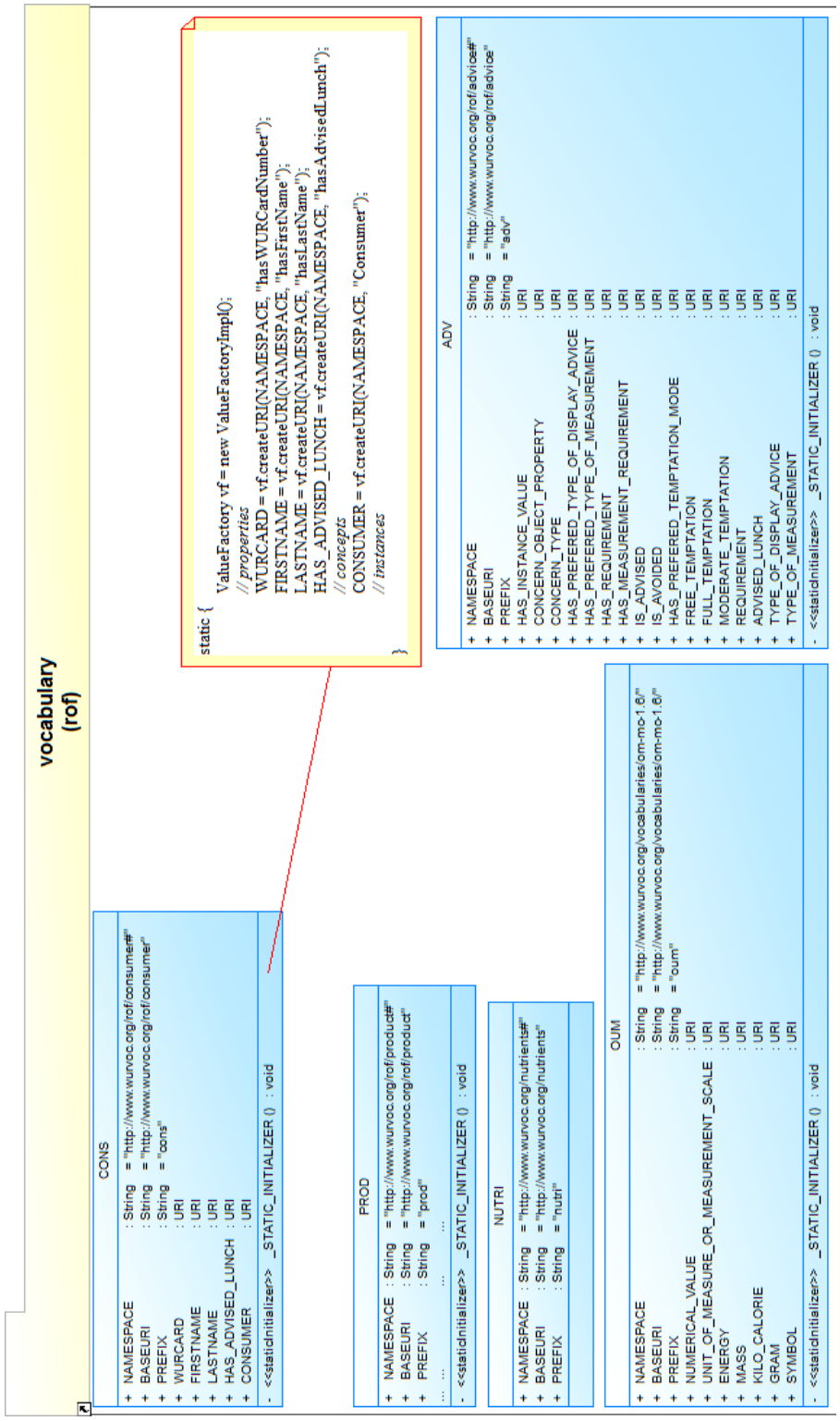


FIGURE H.1 – Paquetage *vocabulary*, diagramme de classes partiel