



Modélisation des champs des formulaires de LesFurets.com

FITUSSI Jonathan

Projet de fin d'études

Université Pierre et Marie Curie

Encadrant Université : CHAILLOUX Emmanuel

Encadrant Entreprise : DEGERBAIX Benjamin

Aout 2015

Table des matières

Introduction	i
1 LesFurets.com	1
1.1 L'assurance	1
1.2 Les comparateurs d'assurance	1
1.3 Historique de l'entreprise	2
1.4 L'application Web	3
2 Sujet de stage	5
2.1 Metamodel	5
2.2 Analyse du metamodel	6
2.3 Analyse Fonctionnelle	9
2.4 Conception	9
2.5 Fonctionnalités futures	9
2.5.1 Page du site	9
2.5.2 CRM - Envoi d'emails	10
3 Problématique	11
3.1 Contexte	11
3.2 Génération de code	11
3.3 API Reflection	12
3.4 Fichier de données	12
3.5 Interface	12
3.5.1 Eclipse Sirius	12
3.5.2 Application Web	12
3.5.3 Donnée Business	12
4 Gestion de projet	15
4.1 Agile	15

4.2	Kanban	16
4.3	Kanban chez lesFurets	16
4.4	Git	17
4.5	Intégration continue	17
4.6	Deploiement continu	17
4.7	Feature Branch	18
4.8	Ma methodologie	18
5	Solution	19
5.1	Eclipse Sirius	19
5.2	Application Web	19
5.3	Développement de l'outil	20
5.3.1	Tableau JSON	20
5.3.2	Génération de code	20
5.3.3	Génération du graphe	21
5.3.4	Zoomer, déplacer les éléments, se déplacer	22
5.3.5	Layout du graphe	22
5.3.6	Layout WireFrame	23
5.3.7	Layout DAG	24
5.3.8	Recherche d'un champs et autocompletion	25
5.3.9	Champs du formulaire	25
5.3.10	Modification des éléments et des dépendances	26
Modification du label du champs	26	
Modification type du champs	26	
Suppression d'une dépendance	26	
5.3.11	Opération sur le graphe	27
Sauvegarde du graphe	27	
Charger un graphe	27	
Calcul des différences	27	
Versionnement du graphe	27	
5.4	Données Business	28
5.4.1	Récupération des données	28
5.4.2	Affichage des données	28
Conclusion	29	
Remerciements	29	

Introduction

Dans le cadre de mon projet de fin d'études il m'a été demandé de réaliser un projet au sein de l'entreprise dans laquelle je réalisais mon alternance. Après avoir examiné l'ensemble des chantiers techniques en attente dans l'entreprise, je me suis trouvée nez à nez avec "Modélisation du graphe des champs des formulaires". L'architecture de l'application Web ainsi que le business de LesFurets.com est centré autour des **formulaires permettant aux internautes de cibler leur profil**. En effet le site LesFurets.com est un comparateur d'assurance (Auto, Moto, Habitation, Crédit et Mutuelle Santé), de crédit à la consommation et de forfaits mobile. Avec LesFurets.com, les utilisateurs peuvent comparer facilement les offres d'un panel d'assureurs pour trouver le produit qui convient le mieux. D'un point de vue technique l'équipe desFurets.com a décidé d'**abstraire les champs des formulaires ainsi que leurs dépendances à l'aide d'un metamodel**. A l'heure actuelle, l'application web est capable de transformer ce modèle de champs d'un formulaire au format XMI pour pouvoir les exporter dans un logiciel de modélisation comme MagicDraw, Rational. L'objet de mon projet sera de proposer un **outil capable d'afficher sous forme de graphe ce metamodel** ce modèle. Pour mener à bien le projet il me faudra, dans un premier temps, passer à d'autres format de modélisation (EMF, JSON) qui pourront être lus à l'aide d'outils open-source. Il est ensuite prévu de superposer des trackers analytics sur ces graphes. Il est aussi envisagé de garder une trace des différences entre les versions successives de l'application pour visualiser les modifications et/ou les régressions en fonctions des développements. Enfin au même titre qu'un IDE classique, l'outil devrait pouvoir proposer des fonctions d'édition, de sauvegarde et de partage des données traitées.

LesFurets.com

1.1 L'assurance

Le secteur de l'assurance fait partie du secteur économique tertiaire qui regroupe les industries de services essentiellement immatériels. Il s'agit d'un secteur d'activité où les acteurs économiques vendent une protection contre le risque avec un poids très important au sein du paysage économique mondial et, plus particulièrement, au sein du paysage économique français. Sa particularité réside dans son activité et sa diversité, puisqu'il rassemble une multitude d'acteurs : compagnies d'assurance, courtiers, mutuelles, acteurs internet, etc... Mondialement, le marché de l'assurance est actuellement dominé par les états-Unis suivi par le Japon. La France occupe la 4ème position, derrière le Royaume Uni, avec 6,5% des cotisations mondiales, soit près de 211 235 millions d'euros. (FFSA et Gema)

1.2 Les comparateurs d'assurance

Un comparateur de prime d'assurance est un outil informatique permettant de comparer toutes les offres d'assurances répondant à un ensemble de critères mentionnés par le client. Avec un marché fortement concurrentiel et très complexe, les comparateurs ont un rôle de plus en plus important auprès des clients en matière de comparaison, de choix et de souscription de produits d'assurance. Aujourd'hui, le marché est occupé par des concurrents connus tels que LeLynx.fr, LesFurets.com, Assurland.com, Hyperassur, automotocompare.fr, kelassur.com, etc... En effet, ces comparateurs présentent des environnements ergonomiques permettant aux visiteurs d'effectuer un comparatif de contrats auto, moto, multirisques habitation, emprunteur ou santé, et de sélectionner les offres les mieux adaptées à leurs attentes. Pour répondre aux besoins des clients, chaque acteur maintient son propre modèle qui repose sur plusieurs aspects : le tarif, la qualité des services proposés, les services après-vente, les garanties, les franchises, les prestations, etc. Ainsi, chaque comparateur présente des offres d'assureurs différents. Le référencement web et la publicité présentent des investissements lourds pour l'ensemble des concurrents qui cherchent à obtenir une visibilité conséquente auprès de leur clients. Les nouveaux arrivants

doivent forcément investir massivement pour trouver une place parmi les acteurs historiques sur ce marché.

1.3 Historique de l'entreprise

Courtanet, fondée en 2005 par Jehan de Castet, est la société éditrice du site AssureMieux.com. Avec ses sites www.assuremieux.com, www.creditmieux.com, et son logiciel Bénéfit, la société est devenue rapidement l'un des plus importants fournisseurs français de solutions de comparaison d'assurances. Plus de 1 500 courtiers indépendants français étaient équipés de ses solutions. Depuis Novembre 2010, Courtanet est majoritairement détenu par BGL Group créé en 1992, intermédiaire britannique d'assurances et propriétaire également du leader de la comparaison d'assurances en Grande Bretagne via www.comparethemarket.com, qui a emporté 35% du marché au Royaume-Uni. Le groupe est aujourd'hui structuré autour de quatre «piliers» : Entreprises intermédiaires, Entreprises pilotées par marque comme LesFurets pour la France, HoyHoy pour la Hollande, Beagle Street pour le Royaume Uni et d'autre services transverses. En Avril 2012, la plateforme de comparaison Assuremieux change d'identité et dévoile une nouvelle marque baptisée LesFurets.com, le comparateur qui « simplifie » l'assurance. LesFurets.com est un comparateur indépendant et impartial d'assurance auto, moto, santé, habitation et crédit. Sa mission est d'aider les consommateurs à trouver l'assurance la mieux adaptée à leurs besoins au meilleur tarif. La mission du site est d'apporter plus de transparence et de simplicité dans l'assurance. Ainsi, il permet de comparer facilement les tarifs, les garanties, les franchises et les services de grands assureurs auto, santé, moto, habitation et crédit. Son objectif est d'aider à trouver l'assurance la mieux adaptée aux besoins des clients au meilleur tarif. Dans le but d'offrir un service de qualité, LesFurets.com s'associe avec les plus grands assureurs du marché : Direct Assurance, Amaguiz, Allainz, Eurofil, AllSecur, Aon Assurances, AcommeAssure, SOS Malus, Assurpeople, L'olivier Assurances, 4assur, EuroAssurance, ActiveAssurances, AssurOnline, assurbike, aloa Assurances, SwissLife, Alptis, etc.



1.4 L'application Web

Le site LesFurets.com est une application Web écrite en Java SE et mise à jour régulièrement avec la dernière version de Java, aujourd’hui elle est pleinement compatible avec Java 8. Elle est déployée à l'aide de Apache Maven. Apache Maven est un outil pour la gestion et l'automatisation de production des projets logiciels Java. L'objectif recherché est comparable au système Make sous Unix : produire un logiciel à partir de ses sources, en optimisant les tâches réalisées à cette fin et en garantissant le bon ordre de fabrication. Il est semblable à l'outil Ant, mais fournit des moyens de configuration plus simples, eux aussi basés sur le format XML. Maven est géré par l'organisation Apache Software Foundation. Précédemment Maven était une branche de l'organisation Jakarta Project. Maven utilise un paradigme connu sous le nom de Project Object Model (POM) afin de décrire un projet logiciel, ses dépendances avec des modules externes et l'ordre à suivre pour sa production. Il est livré avec un grand nombre de tâches pré-définies, comme la compilation de code Java ou encore sa modularisation. Un élément clé et relativement spécifique de Maven est son aptitude à fonctionner en réseau. Une des motivations historiques de cet outil est de fournir un moyen de synchroniser des projets indépendants : publication standardisée d'information, distribution automatique de modules JAR. Ainsi en version de base, Maven peut dynamiquement télécharger des dépendances sur des dépôts logiciels connus. Il propose ainsi la synchronisation transparente de modules nécessaires. C'est ainsi qu'à chaque packaging, la gestion des librairies nécessaires, la compilation et l'exécution des tests présent dans le projet sont automatiquement effectuées. Enfin la base de code est mise à jour à l'aide du gestionnaire de version GIT héberger sur un serveur de l'entreprise.

Sujet de stage

D'un point de vue technique l'équipe de lesfurets.com a décidé d'abstraire les champs du formulaire ainsi que les dépendances à l'aide d'un metamodel. C'est à dire que les champs ainsi que leurs dépendances épousent un format défini à un niveau d'abstraction différent du code. Ils ne sont pas implémentés au moment de l'instanciation des champs mais au niveau du modèle de données sous forme d'Enum Java. L'implantation de la construction permet de caractériser chaque champ et de le présenter à l'utilisateur sous la forme voulu (combo box, radio bouton, date picker, ...). De plus chaque champ connaîtrait par la suite les comportement nécessaires lors de l'évolution de ses co-dépendants. A l'heure actuelle, l'application Web est capable de transformer ce modèle de champs au format XMI pour pouvoir l'exporter dans un logiciel de modélisation comme MagicDraw, Rational. Le stage débutera par un passage à d'autres formats de modélisation (EMF, JSON) traités à l'aide d'outils open-source qui permettront aux équipes de mieux visualiser le graphe des champs des formulaires ainsi que leurs dépendances. Il est prévu ensuite de superposer des trackers analytics sur ces graphes. Il est aussi envisagé de garder une trace des différences entre les versions successives de l'application pour visualiser les modifications et/ou les régressions en fonctions des développements. Enfin au même titre qu'un IDE classique, l'outil devrait pouvoir proposer des fonctions d'édition, de sauvegarde et de partage des données traitées.

2.1 Metamodel

Metamodel signifie littéralement modèle du modèle. Il peut être défini comme la représentation d'un point de vue particulier sur des modèles. On peut voir les couches d'abstraction entre le modèle et le metamodel. Certains langages à objets, comme Java, décrivent de la même manière les classes, les métaclasses et les objets. En génie logiciel, c'est avec UML que la notion de métamodèle a été développée. Le métamodèle décrit les modèles : classes, objets, attributs, relations... et se décrit lui-même. Au-delà de leur diversité, les modèles présentent une caractéristique commune : ils sont des représentations d'un point de vue (d'une conception, d'une théorie...) particulier sur un système sujet d'études. Ils sont écrits dans un code (un langage,

un formalisme...) approprié à l'expression et à l'usage des connaissances qu'ils véhiculent. La notion de « Modèle » est donc directement liée à celles de « Formalisme », de « Point de Vue/Theorie » et de « Sujet d'étude/Système ». On retrouvera l'application de la notion de model et de metamodel dans la figure ci-dessous.

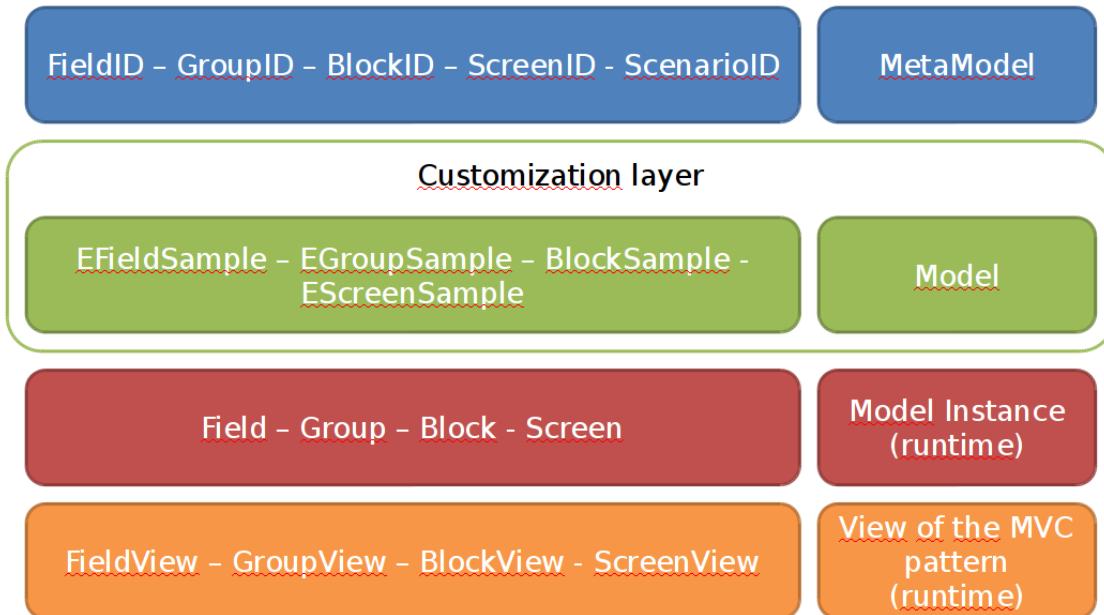
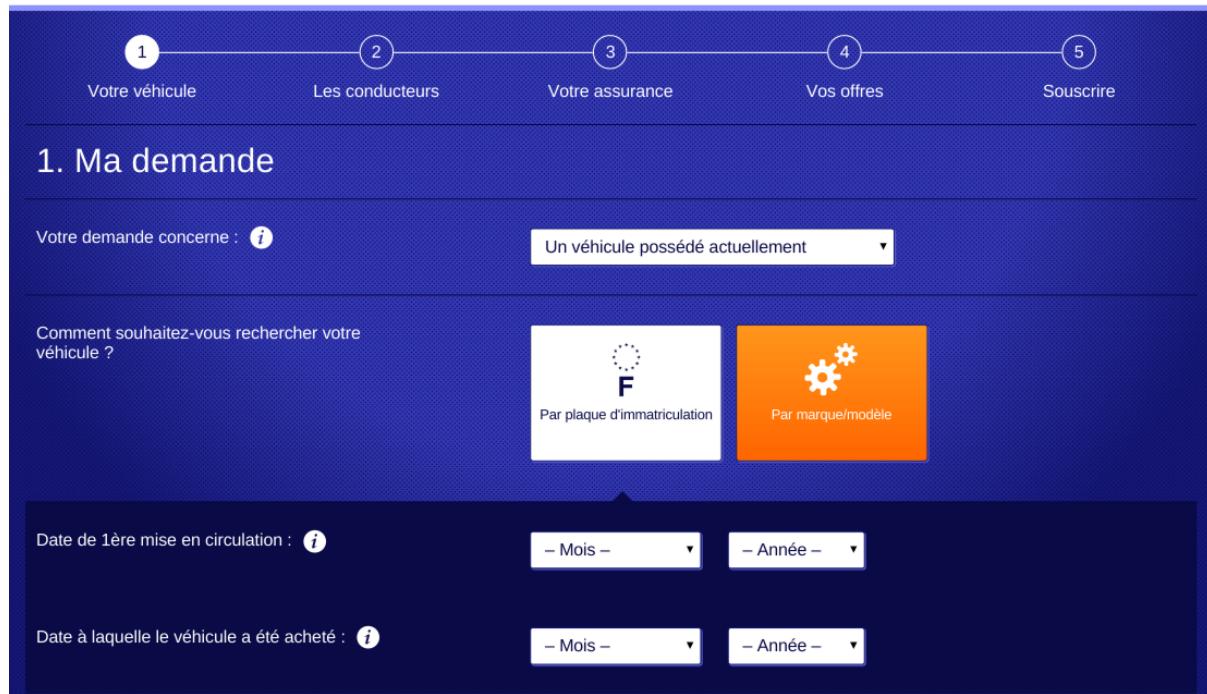


FIGURE 2.1 – Modele et Metamodel appliqué aux formulaires du site

2.2 Analyse du metamodel

Le metamodel étant déjà exploiter dans l'application sous un format exportable, ma première approche était d'afficher le modèle du formulaire pour trouver un moyen pertinent de l'afficher par la suite. Il m'a fallu me familiariser avec l'outil de modeling déjà utilisé dans l'entreprise : MagicDraw. Le diagramme UML présenté montre des cycles, rares mais présent, dans le graphe de dépendances. Le données peuvent être des "fields", "groups", "blocks" ou "screen" ce qui correspond respectivement à un champ du formulaire, un groupe de champs, un bloc de champs et un écran. Pour avoir une vision plus concrète de leur correspondance dans un formulaire du site vous pouvez voir dans l'image ci-dessous que nous sommes dans l'écran "Votre Véhicule", dans le bloc "Ma demande", qu'il existe un groupe pour la recherche du véhicule, en fonction de la marque et du modèle, regroupant plusieurs champs. De plus il existe une hiérarchie claire dans le graphe. Tous les champs, bloc et groupe sont compris dans un écran, les champs peuvent être compris dans un blocs ou groupe et les groupe peuvent être compris dans un bloc. Chaque groupe, bloc ou écran comporte au moins champ. Tous les éléments sont regroupés au sein d'Enum Java



1 Votre véhicule 2 Les conducteurs 3 Votre assurance 4 Vos offres 5 Souscrire

1. Ma demande

Votre demande concerne : i

Un véhicule possédé actuellement

Comment souhaitez-vous rechercher votre véhicule ?

Par plaque d'immatriculation Par marque/modèle

Date de 1ère mise en circulation : i

– Mois – – Année –

Date à laquelle le véhicule a été acheté : i

– Mois – – Année –

```

1 public enum EScreenAuto implements ScreenID, CodeLookup<EScreenAuto> {
2
3     SCR_VEHICLE("Vehicule", ID_STAT_VEHICLE, PRE_TARIFICATIONS, // 
4                 BLK_MA_DEMANDE, // 
5                 BLK_UTILISATION_DU_VEHICLE, // 
6                 GRP_QUOTE),

```

Listing 2.1 – Enum Java de l'écran Véhicule

```

1 public enum EBlockAuto implements BlockID, CodeLookup<EBlockAuto> {
2
3     // Votre v hicule
4     BLK_MA_DEMANDE("BLK_MA_DEMANDE", // 
5                     ID_STAT_VEHICLE, // 
6                     VEH_DTL_DEMANDE, // 
7                     VEH_DTL_CHOIX_RECHERCHE_VEHICLE,// 
8                     GRP_VEH_IMMAT, // 
9                     GRP_VEH_ETAT_FINANCE, // 
10                    GRP_VEH_IMMAT_ACHAT_ACHAT_PREVU, // 
11                    VEH_DTL_SELECTION_MODELE, // 
12                    VEH_DTL_CODE_SRA, //

```

```

13         VEH_DTL_VALEUR_REELLE, //
14         VEH_DTL_VALEUR_ESTIMEE, //
15         VEH_DTL_TITULAIRE, //
16         VEH_DTL_SECOND), //

```

Listing 2.2 – Enum Java du bloc Ma Demande

```

1 public enum EGroupAuto implements GroupID, CodeLookup<EGroupAuto> {
2     GRP_VEH_ETAT_FINANCE("GRP_VEH_ETAT_FINANCE", //
3                         VEH_DTL_ETAT), //
4     GRP_VEH_IMMAT("GRP_VEH_IMMAT", //
5                   VEH_DTL_IMMAT_NUM, //
6                   VEH_DTL_IMMAT_SERVICE, //
7                   VEH_DTL_IMMAT_DESCRIPTIONS, //
8                   VEH_DTL_IMMAT_REGISTRATION, //
9                   VEH_DTL_IMMAT VERSIONS), //
10    GRP_VEH_VEHICULE_MOBILE("GRP_VEH_VEHICULE_MOBILE", //
11                           VEH_DTL_VEHICULE_TITRE, //
12                           VEH_DTL_AVAILABLES, //
13                           VEH_DTL_MODE_CAROSERIE, //
14                           VEH_DTL_MARQUE, //
15                           VEH_DTL_MODELE, //
16                           VEH_DTL_CARBURANT, //
17                           VEH_DTL_CARROSERIE, //
18                           VEH_DTL_PUISSANCE, //
19                           VEH_DTL_VERSION), //
20    GRP_VEH_IMMAT_ACHAT_ACHAT_PREVU(" //
21                           GRP_VEH_IMMAT_ACHAT_ACHAT_PREVU", //
22                           VEH_DTL_IMMAT, //
23                           VEH_DTL_ACHAT, //
24                           VEH_DTL_ACHAT_PREVUE), //

```

Listing 2.3 – Enum Java du groupe choix du modèle

2.3 Analyse Fonctionnelle

Les utilisateurs de l'outil seront d'une part les architectes logiciel de la société, les développeurs mais aussi les Business Analysts (concepteurs fonctionnels) qui pourront mieux cerner les impacts des modifications au seins du formulaire. Ils pourront aussi mieux cibler les besoin des utilisateurs du site. Ainsi il faudra imaginer une interface qui pourrait s'apparenter à des wireframes qui ne sera plus l'apanage des seuls ingénieurs. On pourrait aussi proposer des interfaces différentes en fonctions des utilisateurs de l'outil et de l'utilisation qu'ils en feront.

2.4 Conception

Dans le cadre du projet, il me faudra explorer plusieurs pistes : La première consistera à me familiariser avec un outil de modeling basé sur Eclipse : Sirius (<https://eclipse.org/sirius/overview.html>). Sirius est un projet Open Source de la Fondation Eclipse. Cette technologie permet de concevoir un atelier de modélisation graphique sur-mesure en s'appuyant sur les technologies Eclipse Modeling, en particulier EMF et GMF. L'atelier de modélisation créé est composé d'un ensemble d'éditeurs Eclipse (diagrammes, tables et arbres) qui permettent aux utilisateurs de créer, éditer et visualiser des modèles EMF. Le projet est une initiative française et la communauté autour du projet est très active. Je me mettrai en relations avec les créateurs du projet pour pouvoir aborder les questions que j'aurai tout au long de mon stage. Il me faudra aussi explorer la piste de la création d'une application Web en charge d'afficher le modèle. Il existe de multiple librairies en JavaScript pour l'affichage des données sous la forme de graphe orienté. De plus il s'agit d'un graphe orienté faiblement cyclique qu'on pourra traiter, coté serveur avec des algorithmes implémenté en Java, Scala, Python ou JavaScript. Enfin aujourd'hui le graphe est généré sur un fichier au format XMI mais on pourra imaginer que l'outil introspecte les objets directement dans les binaires. De plus, étant donnée l'étendue de la problématique, la solution pourrait se répartir sur plusieurs outils.

2.5 Fonctionnalités futures

Une fois l'outil développé et adapté aux besoins des formulaires, les équipes souhaiteraient aussi modéliser d'autres données déjà représentées par des metamodels au sein de l'application Java.

2.5.1 Page du site

Les pages du site, les URL, les CSS, ainsi que la JSP utilisés sont représentés dans l'application sous forme d'Enum Java. Ainsi au même titre que pour les champs du formulaire on pourrait imaginer un outil affichant une carte des pages du site ainsi que leurs liens les unes vers

les autres. Nous pourrions par la suite lier les données SEO déjà présentes et les afficher à même le graphe.

2.5.2 CRM - Envoi d'emails

Les site communique avec ses utilisateurs grâce à des emails envoyés une fois leur parcours fini. Par exemple une fois un formulaire rempli si aucune offre n'est choisi l'utilisateur peut choisir de recevoir des offres en fonction des critères qu'il a choisi. C'est ainsi que l'application gère un nombre de règle pour un envoi d'email automatique à différents moments du cycle de vie d'une fiche utilisateur. Ce modèle d'email est lui aussi abstrait grâce à un metamodel et un modèle de donnée défini dans une Enum Java. On pourrait, comme pour les champs des formulaires, imaginer de modéliser le cycle de vie d'une fiche utilisateur avec le nombre d'envois d'emails, les dates ; et superposer les données de retour des utilisateurs en fonction des mails envoyé.

Problématique

3.1 Contexte

L'application du site LesFurets.com est construite sur une architecture JAVA SE packagée à l'aide de l'outil de gestion et d'automatisation de production des projets logiciels Maven. Maven s'occupe de gérer les dépendances de code générées par certaines classes ainsi que de compiler le projet. Chaque formulaire est représenté au sein du site par un tunnel, toute la logique est gérée par GWT (Google Web Toolkit) qui s'occupe de générer du code JavaScript depuis du code Java. L'intérêt du framework est de permettre aux développeur Java de l'entreprise d'aborder leur code coté client et coté serveur avec la même approche. De plus GWT met l'accent sur des solutions efficaces et réutilisables aux problèmes rencontrés habituellement par le développement AJAX : difficulté du déboggage JavaScript, gestion des appels asynchrones, problèmes de compatibilité entre navigateurs, gestion de l'historique et des favoris, etc... Au sein de l'application il existe déjà un projet pour générer des fichiers MDXML et CSV représentant les champs du formulaire c'est dans cette partie du code que l'on développera l'outil de modélisation. Pour la modélisation en elle même Il s'agira essentiellement d'identifier les entités logiques et les dépendances logiques entre ces entités. La modélisation des données est une représentation abstraite, dans le sens où les valeurs des données individuelles observées sont ignorées au profit de la structure, des relations, des noms et des formats des données pertinentes. Pour simplifier la problématique, l'équipe de LesFurets.com a décidé d'abstraire les champs du formulaire ainsi que les dépendances à l'aide d'un metamodel. Un metamodel décrit la structure des modèles et permet de raisonner sur les modèles comme sur les connaissances de premier niveau.

3.2 Génération de code

Une fois l'Enum Java créée pour un champ du formulaire, les labels et d'autres caractéristiques du champ sont créées à l'aide de différentes Classes qui sont code-générées à différents moment lors de la compilation Maven. La première problématique qui se présente est de bien

comprendre à quel moment sont générées toutes les classes liées à ces champs. Toutes ces informations seront nécessaires par la suite.

3.3 API Reflection

Pour récupérer les bonnes informations dans le JAR il me faut utiliser une API d'introspection. Pour cela il m'a faut comprendre comment fonctionne la compilation de Java, comment fonctionne un classloader et à quel moment je peux interroger mon API de réflexion. Ensuite il faudra traiter les données récoltées pour les repartir dans des fichiers plats.

3.4 Fichier de données

Il a aussi fallu stocker dans des fichiers les données des champs des formulaires pour pouvoir y accéder rapidement. Ces données n'étant qu'en lecture seul, il n'y avait pas nécessité de les stocker dans une base de donnée. Le format XML est le format le plus utilisé dans l'application et il est facile de transformer une format MDXML en format XML simple. La difficulté se trouvera dans la fabrications des fichiers JSON.

3.5 Interface

La partie interface dépendra de l'outil utilisé.

3.5.1 Eclipse Sirius

Dans le cadre de Eclipse Sirius, il faudra proposer une interface clair et un déploiement automatique sans qu'il n'y ai besoin de reprendre le système en main à chaque étape. Le placement des objet ainsi que leur design pourront se trouver dans la base de code et être interprété comme le reste du projet pour détecter les erreurs.

3.5.2 Application Web

Dans le cadre de l'application Web il s'agira de CSS et de JavaScript classique. Cependant il faudra placer l'application dans les pages de maintenance du site. Les pages de maintenance sont regroupées dans un package Java et il faudra rendre accessible les informations à ces pages ce qui n'est pour le moment pas le cas.

3.5.3 Donnée Business

Les données business sont disponible dans une base OLAP dans laquelle on pourra lire sans risque de perturber le site. Plusieurs choix s'offrent pour le traitement de ces données :

- Récupérer les données à l'instant où la page est affichée
- Mettre en cache à des périodes données les informations nécessaires
- Réagir de manière réactive aux données mises à jour pour afficher une évolution des données en fonction des changements dans l'architecture des formulaires

Gestion de projet

Après une semaine passée à essayer de mettre en place une solution technique élégante, il est apparu évident pour mon tuteur que nous devions définir une méthodologie pour construire l'outil de modélisation. Au seins de l'équipe IT de LesFurets.com, c'est la méthodologie Agile Kanban qui est en vigueur.

4.1 Agile

Les méthodes agiles sont des groupes de pratiques de projets de développement en informatique (conception de logiciel), pouvant s'appliquer à divers types de projets. Elles ont pour dénominateur commun l'Agile manifesto. Rédigé en 2001, celui-ci consacre le terme d'« Agile » pour référencer de multiples méthodes existantes. Les méthodes agiles se veulent plus pragmatiques que les méthodes traditionnelles. Elles impliquent au maximum le demandeur (client) et permettent une grande réactivité à ses demandes. Elles visent la satisfaction réelle du client en priorité aux termes d'un contrat de développement. Les méthodes agiles reposent sur une structure (cycle de développement) commune (itératrice, incrémentale et adaptative), quatre valeurs communes déclinées en douze principes communs desquels découlent une base de pratiques, soit communes, soit complémentaires. Les méthodes pouvant se qualifier d'agile depuis la publication de l'agile manifesto sont le RAD (développement rapide d'applications) (1991) avec DSDM, la version anglaise du RAD (1995) ainsi que plusieurs autres méthodes, comme ASD ou FDD qui reconnaissent leur parenté directe avec RAD. Les deux méthodes agiles désormais les plus utilisées sont : la méthode Scrum qui fut présentée en 1995 par Ken Schwaber puis publiée en 2001 par lui-même et Mike Beedle pour enfin être diffusée mondialement par Jeff Sutherland ainsi que la méthode XP Extreme programming publiée en 1999 par Kent Beck. Un mouvement plus large (management agile) couple les valeurs agiles aux techniques de l'amélioration continue de la qualité (plus particulièrement le Lean). On constate un élargissement de l'utilisation d'agile à l'ensemble de la structure de l'entreprise.

4.2 Kanban

Parmis les méthodes Agile il existe le Lean Kanban. Kanban est un terme japonais signifiant « fiche » ou « étiquette ». Cette méthode a été initialement mise en place dans les usines Toyota fin dans les années 60. Parmi les outils agiles, Scrum et Kanban sont aujourd’hui les plus utilisés dans le cadre de la réalisation de projets informatiques. Ce sont tous deux des outils de processus mais néanmoins bien différents. Il est à noter que Kanban est plus adaptatif que Scrum qui fixe un cadre de travail plus rigide.

L’approche Kanban consiste globalement à visualiser le Workflow (Le processus de traitement d’une tâche). On met en place un tableau de bord des items (demandes). Chaque item est placé à un instant donné dans un état. L’item évolue jusqu’à ce qu’il soit soldé. Chaque état du tableau peut contenir un nombre maximum prédéfini de tâches simultanées (défini selon les capacités de l’équipe) : on limite ainsi le WIP (Work In Progress). Il est primordial, pendant l’exécution des tâches, de mesurer le "lead-time". Il s’agit du temps moyen pour compléter un item. Cette durée sera progressivement de plus en plus courte et prévisible.

Les intérêts de la mise en place de cet outil Kanban sont principalement :

- Possibilité de mise en place progressive de la méthodologie Agile (moins directif que Scrum) Les points de blocages sont visibles très tôt. La collaboration dans l’équipe est encouragée pour résoudre les problèmes de manière corrective, le plus tôt possible.
- On peut se passer de la notion de sprint. Un sprint d’une ou deux semaines n’est pas envisageable dans certains cas de figure (réactivité supérieure exigée). La méthodologie Kanban est donc utilisée dans des services de support au client (gestion des tickets d’incidents).
- Facilité de communication sur l’état d’avancement du projet.
- La Définition du Done (Ensemble de critères permettant de considérer la tâche comme traitée) permet de garantir un niveau de qualité constant et défini de manière collective.

4.3 Kanban chez lesFurets

Depuis 3 ans lesFurets utilisent la méthodologie Kanban. Toutes les tâches sont gérées depuis JIRA, un logiciel disponible depuis un client Web. JIRA permet la création de tickets qui correspondent à des tâches. Ces tâches sont attribuées aux personnels de l’IT. Les tickets changent d’états et passent par une phase d’analyse, une phase de développement, une phase de test et enfin par la phase de production. Le processus décrit dans le tableau Kanban est automatisé pour les tickets et une vue sous forme de tableau Kanban est disponible depuis le navigateur Web. Cependant il existe un tableau physique représentant les tâches et leur état pour respecter le concept de management visuel.

4.4 Git

Ma plus grande difficulté à été de comprendre le gestionnaire de version qui était mise en place chez LesFurets. Issue du monde de SVN, je ne comprenais pas les mots tel que pull, push et même le comportement d'un commit me paraissait compliqué à prendre en main. C'est par la suite que je compris que plus qu'un simple gestionnaire de version comme je l'utilisais autrefois Git était une pierre angulaire chez LesFurets de la gestion du projet. Git leur sert à la fois à tester le code, à faire avancer les développements en cours et comme je l'expliquerai dans la partie sur Feature Branch la détection et la résolution de conflits entre plusieurs fonctionnalités.

4.5 Intégration continue

Chez LesFurets l'usine logicielle s'articule autour d'une intégration continue. L'intégration continue est un ensemble de pratiques utilisées en génie logiciel consistant à vérifier à chaque modification de code source que le résultat des modifications ne produit pas de régression dans l'application développée. Le concept a pour la première fois été mentionné par Grady Booch et se réfère généralement à la pratique de l'extreme programming. Le principal but de cette pratique est de détecter les problèmes d'intégration au plus tôt lors du développement. De plus, elle permet d'automatiser l'exécution des suites de tests et de voir l'évolution du développement du logiciel. L'intégration continue est de plus en plus utilisée en entreprise afin d'améliorer la qualité du code et du produit final. L'intérêt de cette pratique est que à chaque changement du code, l'application va exécuter un ensemble de tâches et produire un ensemble de résultats, que le développeur peut par la suite consulter. Cette intégration permet ainsi de ne pas oublier d'éléments lors de la mise en production et donc ainsi améliorer la qualité du produit. C'est ainsi qu'à chaque fois qu'un développeur termine une fonctionnalité, les modifications doivent passer une batterie de tests complète. Le logiciel utilisé pour gérer l'intégration est TeamCity il est toutefois couplé à Jenkins.

4.6 Deploiement continu

De plus LesFurets ont aussi expérimenté la culture DevOps. Devops est un mouvement visant à réduire la friction organisationnelle entre les "devs" (chargés de faire évoluer le système d'information) et les "ops" (chargés d'exploiter les applications existantes). Ce que l'on pourrait résumer en travailler ensemble pour produire de la valeur pour l'entreprise. Dans la majorité des entreprises, la valeur sera économique mais pour d'autres elle sera sociale ou morale. C'est ainsi que les mises en production sont réalisées par les développeurs. Il m'est arrivé à plusieurs reprises de mettre en production des fonctionnalités que j'avais développées.

4.7 Feature Branch

Enfin pour assurer l'indépendance des tâches, chaque nouvelle fonctionnalité est développée sur une Feature Branch qui est tirée à partir de la dernière version du master (qui correspond à la version du site en production). Une fois le développement terminé la branche doit passer par un mécanisme d'"octopus" qui permet de merger ensemble toutes les Feature Branch (et donc toutes les fonctionnalités en cours de développement) et de détecter des éventuels conflits entre les développements en cours. Ensuite grâce à une analyse de chaque conflit, on peut résoudre par différents moyens et mettre la fonctionnalité en production. On peut toutefois décider de développer une fonctionnalité qui ne sera pas tout de suite mise en production et la déposer néanmoins sur le repository distant. Il suffira de préciser que celle-ci n'est pas à prendre en compte par l'octopus.

4.8 Ma méthodologie

En vue de toute l'architecture mise en place chez lesFurets il m'a fallu m'adapter pour pouvoir développer l'outil tout en respectant l'écosystème mis en place et pouvoir profiter des avantages produits par le workflow en place.

Tout d'abord lors d'un BBL (Brown Bag Lunch : une conférence pendant la pause déjeuner) j'ai entendu parler du Kanban "Perso" qui était exempt de certaines règles du Kanban et utilisé pour des projets non collaboratifs. Par la suite l'équipe m'a pris en main pour poser les bases de mon workflow. C'est ainsi que sur mon tableau Kanban personnel (informatisé à l'aide de Trello) je me suis imposé les états suivant :

Backlog	To Do	Doing	Done
---------	-------	-------	------

J'ai du aussi effectuer un découpage précis des tâches que j'allais traiter. J'ai complété ce tableau à l'aide de macro tâches dans JIRA. Cela m'a permis de faire tester les tickets sur lesquels j'étais et de prévenir mon équipe de l'avancement de mon projet. Mon projet s'intégrant dans l'application il m'a fallu dans un premier développer mes fonctionnalité hors de l'octopus et régler les conflit plus tard lorsque l'outil pouvait commencer à être testé.

Solution

5.1 Eclipse Sirius

Les débuts du développement d'un modèle pour Eclipse Sirius fut très fastidieux. Après avoir passé un bout de temps sur le tutoriel présent sur le site je me suis confronté à un manque d'autres exemples me permettant de vraiment prendre en main Sirius. Le manque d'informations à été comblé par l'équipe en charge du développement de Sirius. Cependant les démarches pour disposer d'une formation dans leur locaux à Nantes à pris trop de temps pour différentes raisons et nous avons du mettre de coté cette partie de la solution.

5.2 Application Web

Pour la partie sur l'application Web, ma premier démarche a été de trouver une bibliothèque JavaScript capable de générer un graphe orienté cyclique. Google m'a diriger vers une question similaire sur stackoverflow. J'ai bénéficié d'un comparatif complet des existants, avec pour chacun leurs avantages et inconvénients. Mon choix s'est porté sur cytoscape.js. Cytoscape avait l'avantage de réaliser rapidement un graphe correspondant à mes besoin depuis un fichier JSON. Il offre de plus une intégration avec jQuery et/ou Angular pour la manipulation du DOM. Il est sous licence GPL ce qui faisait partie des impératifs du projet. Enfin Cytoscape propose par défaut une dizaine de layouts utilisables pour explorer le graphe de différentes manière.

5.3 Développement de l'outil

5.3.1 Tableau JSON

La première chose pour pouvoir exploiter un affichage du graphe des champs du formulaire avec Cytoscape est de générer un fichier Json. Le fichier contenant une liste d'objet représentant soit un noeud soit une arête.

```
1 { data: { id: foo }, group: "nodes" },
2 { data: { id: bar }, group: "nodes" },
3 { data: { id: baz, source: foo, target: bar }, group: "edges" }
```

En se servant de la bibliothèque de Google Gson qui permet de transformer un objet Java en objet Javascript, il a été nécessaire de créer des classes correspondant aux critères de Cytoscape. Il a fallu par la suite générer un fichier pour chaque formulaire au format Json contenant chaque champs avec les données qui lui sont propres et les dépendances sous formes d'arêtes.

5.3.2 Génération de code

Au seins du code Java il à fallu utiliser l'API Réflection pour pouvoir utiliser les informations sur les champs des formulaires. Ainsi à l'aide de ce mécanisme il m'a été possible de calculer les dépendances entre les champs, les types de champs utilisés ainsi que d'autres informations qui seront mises à jour à chaque compilation de l'application par Maven.

```
1 private Collection<Object> getFactories(String className,
2     ClassLoader classLoader) throws Exception {
3     final List<Object> factories = new ArrayList<>();
4     final Class<?> clazz = Class.forName(className, true, classLoader);
5     for (Class<?> enumType : new Reflections("net.courtanet", classLoader
6         ).getSubTypesOf(clazz)) {
7         if (Modifier.isPrivate(enumType.getModifiers()))
8             continue;
9         if (!enumType.getSimpleName().startsWith("BundleField"))
10            continue;
11         factories.add(enumType.newInstance());
12     }
13     return factories;
14 }
```

Listing 5.1 – Introspection du JAR

```

1  [
2    {
3      "data": {
4        "label": "SCR_CLAIM",
5        "id": "SCR_CLAIM",
6        "type": "SCREEN"
7      },
8      "group": "nodes"
9    },
10   {
11     "data": {
12       "label": "Votre assurance",
13       "id": "BLK_ASSURANCE",
14       "parent": "SCR_CLAIM",
15       "type": "BLOCK"
16     },
17     "group": "nodes"
18   },
19   {
20     "data": {
21       "label": "CLAIM_DATE",
22       "help": "no_help",
23       "tip": "no_tip",
24       "id": "CLAIM_DATE",
25       "parent": "BLK_ASSURANCE",
26       "type": "VALUE_OBJECT"
27     },
28     "group": "nodes"
29   },
30   {
31     "data": {
32       "id": "ADRESSE_LIVRAISON_DIF",
33       "source": "ADRESSE_LIV_DIF",
34       "target": "LIVRAISON_ADR1"
35     },
36     "group": "edges"
37   },

```

Listing 5.2 – Exemple de fichier JSON généré

5.3.3 Génération du graphe

La logique par la suite était de servir les fichiers Json de façon asynchrone à la demande d'affichage du graphe d'un des formulaire de LesFurets.com. Pour cela la solution vers laquelle je me suis tourner était la manipulation du DOM à l'aide de AngularJS. Cytoscape permet à l'aide de closure de modifier le graph qu'en cas de modifications défini dans le "controller" AngularJS. Ainsi à chaque demande du graphe d'un formulaire à l'aide d'un des boutons de



FIGURE 5.1 – Choix du formulaire

l'interface on demande si le tableau JSON existe. Si il existe on trace le graphe sinon on fait un appel au serveur, une fois le fichier reçu on lance le calcule du graphe. Si il y a une attente, on affichera un élégant loader invitant l'utilisateur à prendre son mal en patience.

5.3.4 Zoomer, déplacer les éléments, se déplacer

Une fois le graphe affiché il est apparu essentiel de pouvoir laisser l'utilisateur zoomer, se déplacer dans le graphe et déplacer les éléments du graphe. A l'aide de la documentation de l'API et aux fonctionnalités permises par la balise canvas de HTML5, j'ai pu définir des fonctions de glisser déposer pour déplacer les éléments ainsi que pour déplacer tout le graphe. Enfin pour les fonctions de zoom j'ai défini le graphe en position relative et j'ai utilisé l'évènement onscroll.

5.3.5 Layout du graphe

Cytoscape dispose d'un système pour permettre aux développeur de définir leur layout. Un layout n'étant qu'un ensemble de règles définies pour afficher les noeuds et arêtes sous forme de graphe.

```
1 var layoutCircle = {  
2   name: circle,  
3   fit: true, // whether to fit the viewport to the graph  
4   padding: 30, // the padding on fit  
5   avoidOverlap: true,  
6   radius: undefined,  
7   startAngle: 3/2 * Math.PI,  
8   counterclockwise: false,  
9 };
```

Listing 5.3 – Exemple d'un layout traçant le graphe sous forme de cercle

Enfin on laissera la possibilité à l'utilisateur de choisir le layout dont il aura besoin.

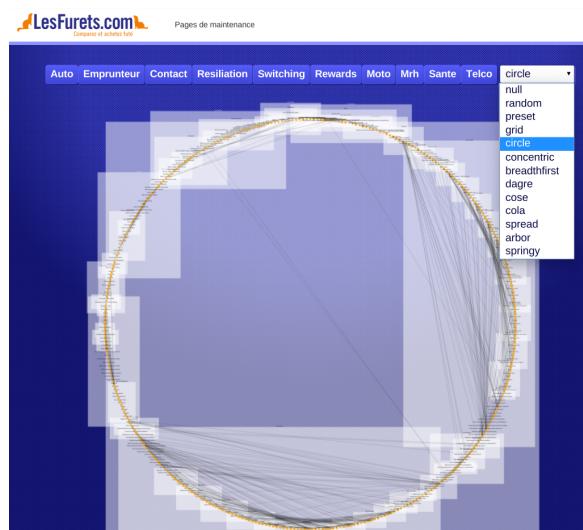


FIGURE 5.2 – Graphe tracé avec un layout "circle"

5.3.6 Layout WireFrame

On proposera un layout "wireframe" qui permettra d'afficher le graphe sous une forme similaire au formulaire proposé sur le site. C'est à dire que tous les écrans sont disposés à la suite les uns des autres et que sont disposé à l'intérieur, verticalement, les champs du formulaire. Pour cela il m'a fallut réaliser deux opérations non triviales :

- Modifier la structure du JSON pour inclure une donnée permettant de définir l'ordre des écrans, blocs, groupes, champs du formulaire.
- Écrire une fonction dans mon code JavaScript permettant de positionner horizontalement et verticalement les éléments.

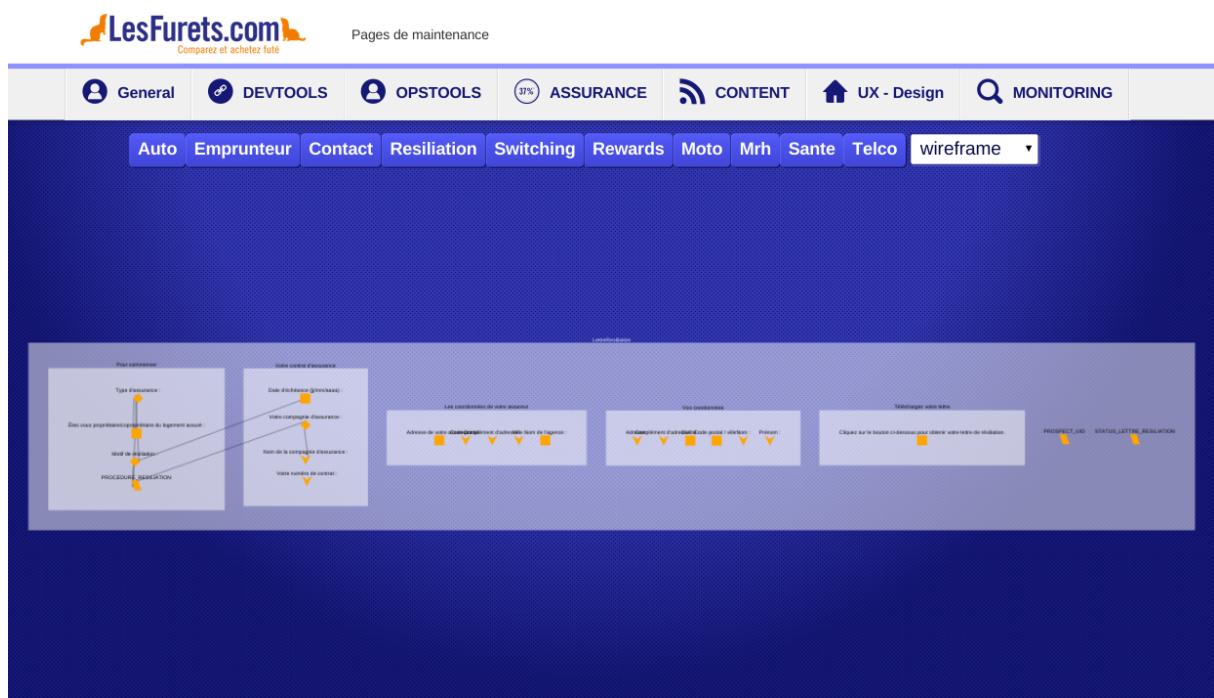


FIGURE 5.3 – Formulaire Résiliation avec le Layout Wireframe

5.3.7 Layout DAG

On proposera aussi un layout "dag" qui affichera le graphe avec en évidence les cycles et les dépendances plus fortes. Pour cela on calculera les dépendances entre les champs. On calculera le score de chaque élément en fonction de son nombre total d'arêtes liées à un élément parent différent, que ce soit un écran, un bloc ou un groupe. Enfin on disposera au plus proche les éléments en fonction des deux éléments contenus ayant les plus grands scores. Ce layout sera intéressant pour des formulaires avec un grand nombre d'éléments pour visualiser les dépendances entre les éléments du formulaire.

The screenshot shows a complex web form for car insurance. At the top, there's a navigation bar with tabs for General, DEVTOOLS, OPSTOOLS, ASSURANCE, CONTENT, UX - Design, and MONITORING. Below the navigation is a horizontal menu with categories like Auto, Emprunteur, Contact, Resiliation, Switching, Rewards, Moto, Mrh, Sante, Telco, and dagre. A search bar labeled 'Rechercher' is also present. The main content area displays a dense network of fields and dropdowns arranged in a non-linear, interconnected layout, illustrating the DAG (Directed Acyclic Graph) structure where fields are positioned based on their dependencies.

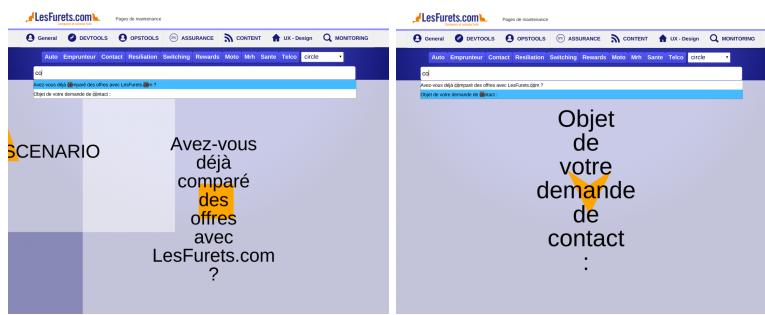
FIGURE 5.4 – Formulaire Assurance Auto avec le Layout DAG

This screenshot shows a different section of the LesFurets.com form, specifically the 'REWARDS_ZSU VALIDATION' part. It features a similar layout structure with a navigation bar at the top and a horizontal menu below it. The main area contains several input fields and dropdowns, such as 'Vos coordonnées' (Your coordinates), 'REWARDS_ZSU VALIDATION', 'COORD_ADM', 'Ville', 'Adresse', 'Nom', 'Prénom', 'Température', 'L'adresse de livraison', 'LIVRAISON_AJOU', and 'Envoyer par e-mail à une personne'. A prominent feature is a large, complex network of arrows connecting these fields, visualizing the dependencies between them according to the DAG layout.

FIGURE 5.5 – Formulaire Rewards avec le Layout DAG

5.3.8 Recherche d'un champs et autocompletion

Une fois le graphe affichant tous les champs du formulaire il nous a paru nécessaire de pouvoir rechercher un champs et de pouvoir zoomer sur celui-ci. Ma première approche était d'écrire une fonction capable de zoomer sur un élément du graphe depuis le code. Une fois celle-ci implémentée, il a été facile de zoomer sur le champs et par la suite de zoomer sur son élément parent direct.



5.3.9 Champs du formulaire

Un autre gros morceaux du problème était de pouvoir afficher chaque champ dans un style semblable à celui présent dans les formulaires. Après avoir récupéré une liste de tous les types d'éléments dans le formulaire : bouton, textbox, checkbox, boutons radio, liste, date, ... Cette liste était disponible depuis une Enum Java. Ensuite il a fallu dessiner dans la balise canvas des éléments ayant la forme des champs du formulaire.

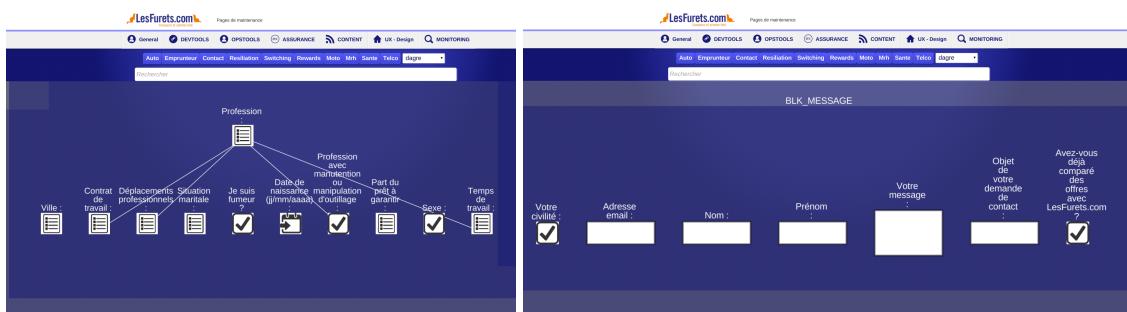


FIGURE 5.6 – Différents type de champs du formulaire

5.3.10 Modification des éléments et des dépendances

Une fois la phase de "Layout" réalisée il a fallu mettre en place des fonctions de modification/suppression des éléments du graphe. Et une fonction de modifications des dépendances entre les champs.

Modification du label du champs

La première fonction était sans doute celle qui avait le moins d'incidence sur le formulaire. Il s'agit de proposer aux utilisateurs de pouvoir changer le nom d'un champ du formulaire.

Modification type du champs

On proposera aussi à l'utilisateur de pouvoir changer le type d'un champ sélectionné. Pour cela il a fallu lister tous les types de champs disponibles. J'ai pu servir directement dans la JSP la liste des types de champs possibles à l'aide d'une Enum présente dans le code Java.

Suppression d'une dépendance

Il s'agit juste de mettre en évidence les champs affectés par la suppression d'une dépendance. Les dépendances ne sont pas interdépendantes. On soulignera les champs impactés par la suppression d'une couleur rouge.

5.3.11 Opération sur le graphe

Sauvegarde du graphe

Une fois les fonctions de modifications mise en place il a paru évident qu'il a fallait pouvoir sauvegarder l'état du graphe modifié pour pouvoir l'envoyer par la suite. La première approche simple était de pouvoir faire une capture d'écran de l'état du graphe dans une résolution assez grande pourvoir voir toute les détails du formulaire. La seconde approche fut de sauvegarder un fichier JSON avec toutes les données modifiées ainsi que les positions des champs. En effet Cytoscape accepte de bloquer une position à la création du graphe sous la forme de coordonnées cartésiennes. Il a ensuite fallu définir un repère qui correspondait au repère de base de Cytoscape. Enfin on enregistre la collection d'éléments sous la forme d'un fichier JSON. Une fois le fichier JSON créé on encode une chaîne de caractère en base64 et on crée le fichier.

Charger un graphe

Une fois qu'on a donné la possibilité aux utilisateurs de sauvegarder le graphe, il a fallu implémenter un module d'import. A l'aide d'une balise input file, on télécharge le fichier précédemment sauvegardé et on affiche le graphe.

Calcul des différences

Une fois le graphe importé nous avons voulu aussi proposer une visualisation des différences entre la version courante et une version importée. Une fois le fichier importé on se retrouve en présence de deux objets JavaScript que l'on peut aisément comparer. Dans le calcul des différences on proposera deux scénarios. Le premier est d'afficher graphiquement les différences sur le graphe, à l'aide de couleurs différentes. Le second scénario correspond à servir une liste des différences sous la forme d'une réponse JSON.

Versionnement du graphe

En se servant du module de calcul de différences en mode non graphique, l'outil peut versionner un fichier des différences que l'on commitera avec nouvelle version du master. Une future amélioration serait de pouvoir lever une alerte si un trop grand nombre d'objets sont différents.

5.4 Données Business

5.4.1 Récupération des données

Après différentes discussions avec les équipes en charge de la conception de l'application nous avons décidé de ne pas faire des appels à la base SQL à chaque affichage du graphe. Cependant il m'a été conseillé d'utiliser les données retournées par des batches (Jobs) en charge de récupérer des informations à intervalles réguliers. Les informations fournies par ces batches servent à créer des dashboards réguliers pour les équipes d'analyse. Ainsi Il nous a fallu quand même définir des règles de tri pour créer un agrégat de données pertinent. Nous avons défini le bloc comme grain d'analyse des données, car c'est le plus petit élément où pour passer d'un bloc à l'autre il faut effectuer une action.

5.4.2 Affichage des données

Une fois les données disponibles, il m'a été demandé de les afficher sous la forme d'une jauge en pourcentage en fonction du nombre d'utilisateur passant l'écran suivant à coté du label de chaque bloc. Si la jauge est en dessous des 50% on affichera aussi la date du champ ou de la dernière dépendance modifié dans le bloc.

Conclusion

Au terme de cette deuxième période d'apprentissage, j'ai pu parfaire mon expérience chez LesFurets.com. Au sein d'une équipe motivée et dynamique j'ai réussi à m'intégrer au groupe, tout en intégrant les méthodologies utilisées. J'ai essentiellement travaillé sur mon projet de fin d'études, mais j'ai aussi pu travailler sur divers sujets en cours. J'ai pu développer de nouvelles fonctionnalités et j'ai pu les mettre en production. J'ai pu assister à des réunions, des points techniques, des conférences et des formations, m'enrichir en débattant autour de relecture du code de mes collègues et aussi de leur relecture du mien. Mon sujet de stage m'a permis d'élaborer une solution technique à un problème qui concernait des gens habitués à être confronté à des développeurs. J'ai pu être challengé aussi bien d'un point de vue technique que fonctionnel. Aujourd'hui l'outil que j'ai développé se trouve dans les pages de maintenance de l'entreprise et peut être utilisé par celui qui en a besoin. De plus il est amené à évoluer au fil du temps avec l'application.

Remerciements

J'adresse mes remerciements aux personnes qui m'ont aidé dans la réalisation de ce projet. Tout d'abord l'équipe pédagogique du master STL/INSTA de l'UPMC. M. Emmanuel Chailloux, M. Binh-Minh Bui-Xuan et M Philippe Trebuchet pour nous avoir suivies toute l'année avec bienveillance et pédagogie. Toutes les personnes de l'entreprise Courtanet qui m'ont aider dans la réalisation de l'outil et avec qui j'ai passé de très bon moments tout au long de l'alternance, Benjamin Degerbaix, Gilles Di Guglielmo, toute l'équipe "Traffic", l'équipe "Journey" ainsi que tout le département IT de l'entreprise. Enfin je souhaiterais adresser des remerciement à tous les étudiants de la promo STL/INSTA 2015 avec qui nous avons construit un grand esprit de camaraderie, d'écoute et d'entraide.