

GestaAE : Facturation auto-entrepreneur

**Mémoire présenté en septembre 2013
par**

Anthony Bobenrieth

**en vue de l'obtention de la Licence Professionnelle
Systèmes Informatiques et Logiciels**



**Spécialité
Métiers du Génie Logiciel**



Alternance effectuée à l'entreprise



*2 av. du General Leclerc
B.P. 63 54703 Maldières*

Remerciements

Je tiens à remercier toutes les personnes qui m'ont permises d'arriver où je suis actuellement et ayant contribué indirectement à la rédaction de ce mémoire.

D'abord la société Dune, qui a cru en moi, et m'a chaleureusement accepté fraîchement sorti d'un bac pro, alors que je ne trouvais pas d'entreprise d'accueil dans ma région. Mais également à l'ensemble de ses salariés à qui je dois désormais une très grande partie de mes connaissances en développement et en vie d'entreprise, et grâce auquel ces trois années sont passées terriblement vite.

De plus, je tiens à remercier plus particulièrement Monsieur N. Roche pour avoir été un tuteur hors pair durant ces trois années, pour avoir pris le temps de m'exposer les ficelles de son travail et pour son grand soutien.

Je salue également le travail de l'ensemble des enseignants de l'université pour nous avoir dispensé un cours de qualité et très enrichissant de même que le personnel chargés du parc informatique et réseaux pour m'avoir agréablement surpris sur la qualité de toute l'infrastructure informatique mise à disposition.

Enfin, j'aimerais saluer les clients de Dune et leurs intermédiaires pour le temps qu'ils m'ont accordé, et l'expérience qu'ils m'ont apportés dans leurs métiers respectifs.

Sommaire

● Abstract	p4
● Introduction	p5
● Dune	p6
○ Présentation de l'entreprise	p6
○ Sujets traités	p8
● Traitement des sujets	p9
○ Site de la société	p9
○ GED Mentat	p13
● Projet Industriel	p22
○ Contexte de création	p22
○ Model	p23
○ View	p28
○ Controller View Model	p35
○ Conclusion	p38
● Conclusion de l'année	p39
● Annexes	p40

Abstract

Already working for the Dune company since 2010, it was a great pleasure for me to re-apply for a new year of sandwich courses with them, and this time at the university.

That company is based in pont à mousson, in a lovely cottage, and is specialized in software development and law consulting. We were five developers, three managers, and one lawyer, and the working atmosphere was as soft as the office's appearance, and this is probably why the relationship between the managers and us were so family-like.

In this company, the projects were always oriented on Microsoft tools, and this year was the opportunity for me to learn to use new ones. For example, WPF (Windows Presentation Foundation), which is a great framework, including specific language to define some beautiful user interface. I also increased my skill in database accessing by using the Entity Framework in a « code first » way.

All these things were used in the two main projects that i led during this year. The first one Mentat, was an Electronic Document Management application that was supposed to work with a lot of external software. The second one, Gestae, was a simply an invoice creator software specifically created for the french « auto-entrepreneurs ».

Meanwhile, the university taught me some stuff quietly different of the work, because the target of our lesson was more wide, not simply supposed to work on Windows. I also learn some pretty things that were new to me, like the basis of team management using « agile methods », or what's a « web services ».

Unfortunately, the company got high financial problems at the end of October, and they brought it to a judicial liquidation in June... We all got laid off some days after...

Introduction

Présent depuis 2010 chez Dune, ce mémoire est un condensé de l'ensemble du travail qui m'a été attribué durant cette année 2012/2013.

Une année où pour la première fois l'approfondissement a pris le pas sur l'apprentissage (même si, comme tout développeurs, on reste apprenti à jamais des nouveautés), et m'a permis de prendre gout à tirer parti des différentes technologies proposée par Microsoft, le partenaire de toujours de Dune (qui exploitait alors, essentiellement VB6 sur ses projets et souhaitait passer ses compétences sur du C#).

Arrivé fervent défenseur du libre et des langages plus bas niveau (C/C++), j'avais alors quelques réticences à exploiter un langage managé, qui plus est, orienté pour ne tourner que sur un seul OS. OS qui, même s'il est le plus répandu, est assez vivement critiqué par la communauté des développeurs depuis un long moment maintenant.

Pourtant ces années d'apprentissage m'ont permis d'entrevoir de nombreuses qualités de l'environnement .NET sur ses équivalent multiplateforme, et m'ont également permis de mettre un pied dans le développement logiciel avec une très grande aisance que je n'aurai peut-être pas forcément eu avec un autres Framework ou d'autres langages.

Ce mémoire ne sera cependant pas une ode à la société de Redmond, mais plutôt une mise en avant des dites qualités qui, même si on pourrait leur reprocher leurs rigidités, gagneraient à être reprises sur d'autres plateformes. En outre j'exposerai également certains nouveaux enjeux encore trop souvent négligés des développeurs, tels que l'accessibilité et le design.

L'opportunité m'est donné de mettre en avant ses éléments en travaillant sur une application de facturation résolument orientée particulier car s'adressant aux auto-entrepreneurs. Cette application sera le sujet de ma partie de projet industriel.

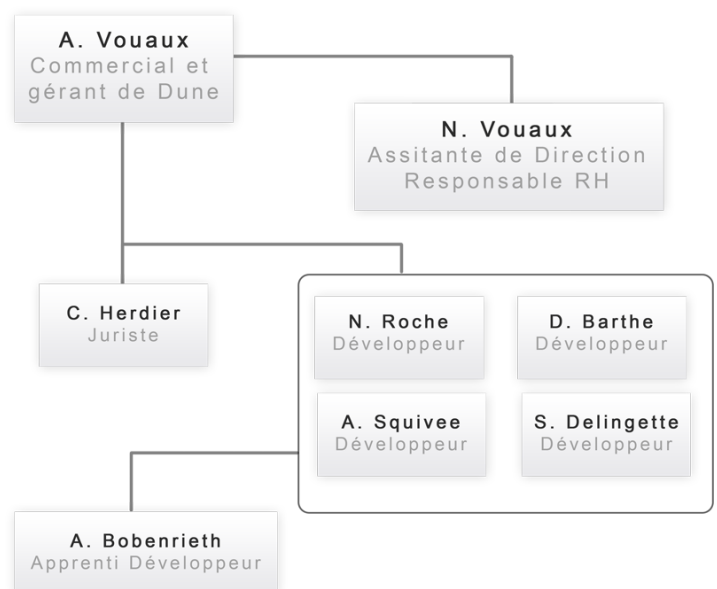
Dune

Présentation de l'entreprise

Dune est une startup Maidiéroises (près de Pont à Mousson) crée en 2009 spécialisée dans le développement et la formation en informatique industrielle ainsi qu'un conseil juridique (suivi et construction de projets innovants).

Organisation de l'entreprise

L'effectif de la société est composé d'une partie développement constituée de 4 développeurs, d'une partie administrative constituée du gérant et de son assistante, et d'un juriste.

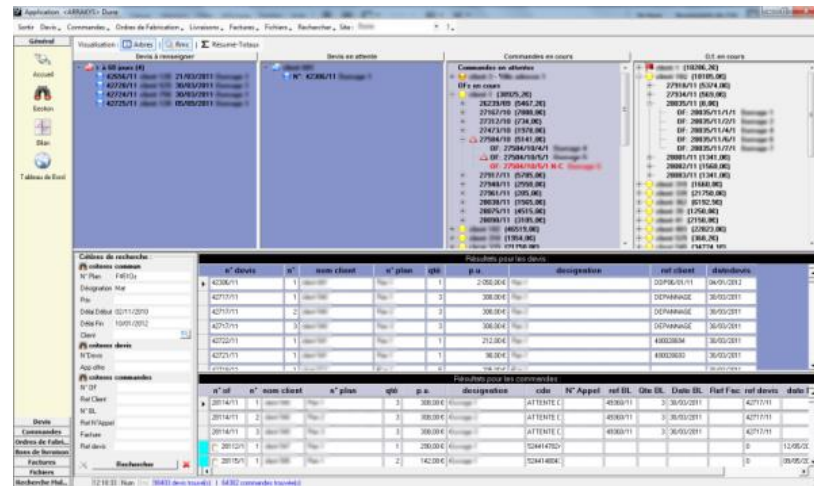


Activité de l'entreprise

L'activité de l'entreprise se découpe donc en deux parties, parfois liées :

D'une part la partie juridique qui s'occupe de conseiller et de suivre nos différents clients dans leurs démarches d'aides de financement.

Et la partie développement, qui se concentre sur la construction des produits phare de la société, la GPAO Arrakys et Caladan.

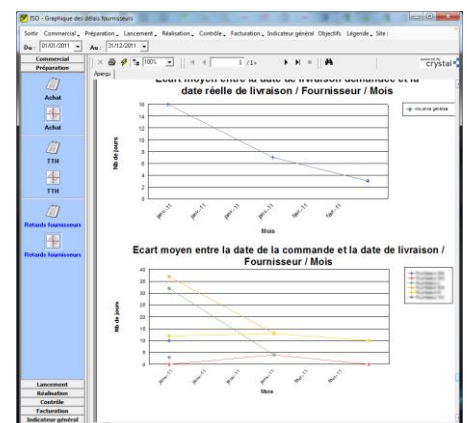


Arrakys, la solution phare de Dune en Gestion de Production Assistée par Ordinateur

Des logiciels de gestion de production conçus en lien direct avec les différents acteurs Lorrain de la métallurgie. Elles tirent leurs forces des différentes adaptations qu'elles ont subies pour chaque nouveau client.

En outre, la société conçoit régulièrement via des développements spécifiques des petites applications qui servent souvent d'extension à terme à l'application principale.

La société revendique son exploitation des technologies Microsoft, ainsi que sa proximité avec les clients pour des solutions sur mesure.



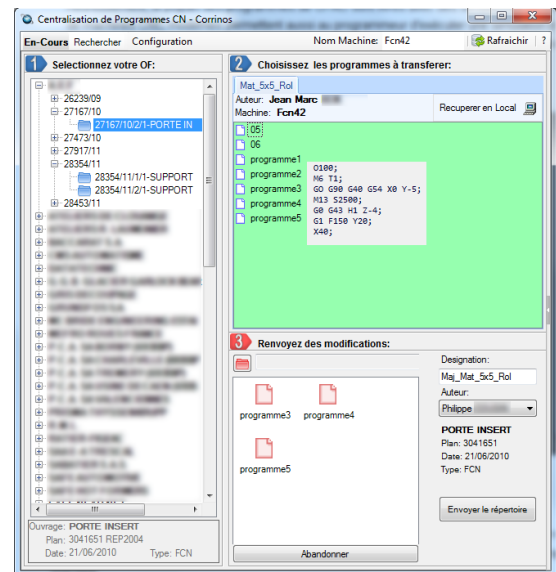
Sujets traités

Rappel bref

J'avais déjà effectué mes deux années de BTS IG en apprentissage avec Dune, au cours desquelles j'avais travaillé principalement sur un système de gestion de demandes de clients extranet (Salusa) disposant d'autres fonctionnalités tels qu'un wiki, et un gestionnaire de documentation de code...

J'avais également travaillé sur un projet de centralisation de programmes pour machine outils à commande numérique (Corrinos).

Aussi, il ne m'aura pas été assigné de petites tâches au cours de cette année mais plutôt la réalisation de deux gros projets à mener de toutes pièces en autonomie.



Corrinos, mon projet de fin de BTS

Site de la Société

Existante désormais depuis 3 ans, et en perte de vitesse dans ces prospections, il m'a été demandé fin août de réaliser le site web de la société afin promouvoir notre activité et d'affirmer la crédibilité de notre statut de société de développement informatique.

G.E.D. Mentat

Il m'a également été demandé fin septembre, sous réclamation de différents clients, de réaliser un module pour GPAO de la société (Arrakys) permettant d'attacher des pièces jointes aux documents générés (facture, devis...).

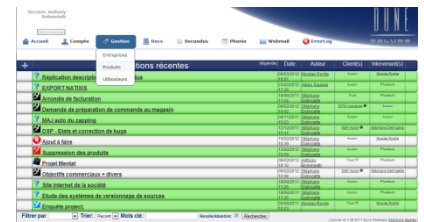
Traitement des sujets

Site de la société

Contexte

Tandis que l'été se terminait au même rythme que les différents projets lancés dans l'année, la direction a estimé qu'il était propice de promouvoir les services de la société afin d'étendre sa prospection (qui marchait jusque-là majoritairement avec les relations existantes et le bouche à oreille).

En tant que seul développeur disposant des compétences web, acquises en cours et sur mon projet d'extranet réalisé lors de mes années en BTS, j'ai donc eu la responsabilité de la réalisation intégrale du site.



	nom	Date	Auteur	Statut	Remarques
+	Produit de base	10/10/2012	Admin	En cours	
+	Produit de base	10/10/2012	Admin	En cours	
+	Produit de base	10/10/2012	Admin	En cours	
+	Produit de base	10/10/2012	Admin	En cours	
+	Produit de base	10/10/2012	Admin	En cours	
+	Produit de base	10/10/2012	Admin	En cours	
+	Produit de base	10/10/2012	Admin	En cours	
+	Produit de base	10/10/2012	Admin	En cours	
+	Produit de base	10/10/2012	Admin	En cours	
+	Produit de base	10/10/2012	Admin	En cours	

Éléments fournis

Dans toutes ses nouvelles démarches commerciales, la société a pris l'habitude de se présenter à travers des documents de promotion (joint par mail ou au format papier), ces documents m'ont de fait été fourni comme base solide pour le contenu du site.

Concernant les ressources graphiques, outre le logo de la société (réalisé la même année par une professionnelle), aucun élément ne m'est fourni. On me charge donc de la réalisation du contenu graphique dans la limite de mes compétences.

Objectifs

On ne cherche pas de quelconque performances ni d'optimisations d'exécution à travers ce genre de projet, mais à faire plus que jamais la publicité de son domaine et donner confiance aux potentiels nouveaux clients.

On veillera également à une réutilisabilité du code : il faut que tout autres développeurs/intégrateur web soit en mesure de mettre à jour simplement le contenu du site, pour qu'il reste synchronisé avec l'actualité de la société.

Choix technologique

En tant que simple « site vitrine », aucune interaction lourde n'est prévue avec le client (sauf un simple formulaire de contact).

Par crainte de perdre trop temps sur l'apprentissage d'un *CMS* et de souffrir par la suite de leurs rigidité pour les futurs demandes de la direction, j'ai préféré opter pour du code brute. Cela permet de se distinguer plus facilement des autres sites

Les technologies majeures seront donc la combinaison l'*Html/Css*, un peu de *JavaScript* pour dynamiser le contenu, et quelques scripts *PHP* pour le formulaire de contact et l'organisation des pages via des *includes*.

Pour l'hébergement, on utilisera notre serveur web (mutualisé) chez ovh, qui servait jusqu'alors à l'extranet et aux mails.

Outils

Principalement orienté sur du code *Html/Css*, le projet ne nécessite pas l'utilisation d'un IDE lourd, les fonctions qu'on pourrait appeler de « débogage » étant déjà suffisante dans les navigateurs web (par défaut sous chrome et IE, avec firebug sous Firefox) je me contenterai donc de Notepad++ pour la partie code.

Partie 1 : Organisation

Un projet aussi simple sur la partie technique ne peut déboucher sur un code illisible et difficilement maintenable, d'autant qu'il faut anticiper une réutilisabilité futurs, car si dans l'immédiat le site se veut principalement statique (aucun contenu généré en fonction des visites), il doit être également facilement modifiable pour quand même suivre les évolution majeurs de la société (publication d'une nouvelle grosse application, agrandissement du champ de compétences de la société, etc....).

Pour faciliter la prise en main et la compréhension, le site est décomposé (pour être à terme réassemblé via les propriétés GET et directive d'inclusion *PHP*).

La gestion des diapositives est elle aussi triviale car elle consiste simplement en une liste d'éléments <div> contenus dans le fichier *diapos.php* et définissant l'ensemble de chaque diapositive.

<http://dune-strategie/index.php>

header.php
+ diapos.php

```
switch($_GET['page'])
{
    case 'temoignage': temoignages.php
    case 'presentation': presentation.php
    ...
}
```

footer.php

Partie 2 : Dynamisme et esthétique

En tant que porte-parole de la société, le site se doit d'être un minimum présentable et véhiculer les valeurs qui doivent être mises en avant par une entreprise spécialisée dans la conception de logiciels.

Afin de garantir une rapidité de chargement des pages, on évitera l'utilisation abusive d'images, on essaiera en priorité d'utiliser les fonctionnalités graphiques introduites en html5 (dégradés, ombres portées...) sans en abuser pour garder une lisibilité optimale.

Partie 3 : Gérer son exposition

L'objectif majeur étant de promouvoir la société, il faut s'assurer pour cela d'une forte exposition sur les moteurs de recherche, en particulier celui le plus utilisé par notre cible (les français) : **Google**.

Les astuces sont innombrables, et sont parfois même dépassées suivant les évolutions des algorithmes du moteur de recherche.

On peut citer dans l'exemple du site de dune :

- Un générateur *PHP* de mots-clés qui sont insérés proche du titre aléatoirement et font référence au développement en Lorraine.
- Un *url rewriting* qui masque les variables \$GET lors du passage de page en page.
- Une omniprésence sur les autres services de Google, tels que *Maps* et *Google+*

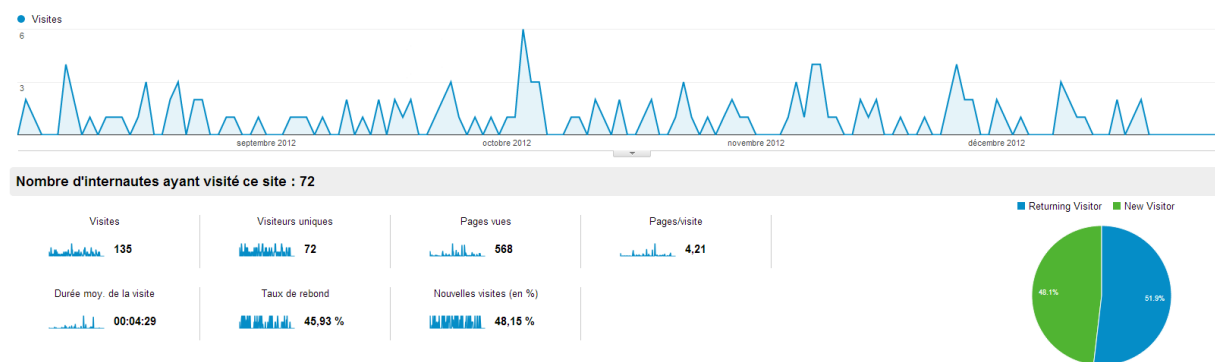
Mais la société part avec une balle dans le pied... à cause de son nom...

En effet, « dune » est constamment traduit par « d'une » dans les moteurs de recherche français. Aussi « dune stratégie Lorraine » sera repris en « d'une stratégie Lorraine ». Un grand handicap pour le référencement.

Statut du projet/Evolutions possibles

Le site fut terminé après 3 semaines de travail et se suffit à lui-même pour présenter la société tel qu'elle existait.

Toutefois en dépit de pas mal d'efforts, le site n'a jamais réellement connus un succès efficace dans le référencement des moteurs de recherche, aussi, le nombre de visiteurs est resté assez bas, avec une moyenne de 2 visites par jour et une moyenne de 18 visiteurs uniques par mois.



Relevé des visites du site de début septembre à fin décembre 2012

Parmi les évolutions possibles, on pourrait donc imaginer un futur budget alloué à des campagnes de promotion payantes sur les moteurs de recherche.

Toujours dans les idées d'évolution, pour un suivi plus précis de l'actualité de Dune, je gardais la possibilité d'y intégrer un flux *Twitter* (rapide à installer, tout le monde sait l'utiliser, et on peut s'en servir partout...), mais cela ne m'a jamais été demandé.

GED : MENTAT

Contexte

L'un des manques majeur souvent reproché à la solution GPAO de Dune est de ne pouvoir fournir des fonctionnalités de gestion des documents suffisantes pour faire le lien entre les éléments générés par l'application (devis, facture...), et les pièces qui peuvent y être rattachées mais provenant des logiciels de bureautique annexes (mail, documents *Word*...).

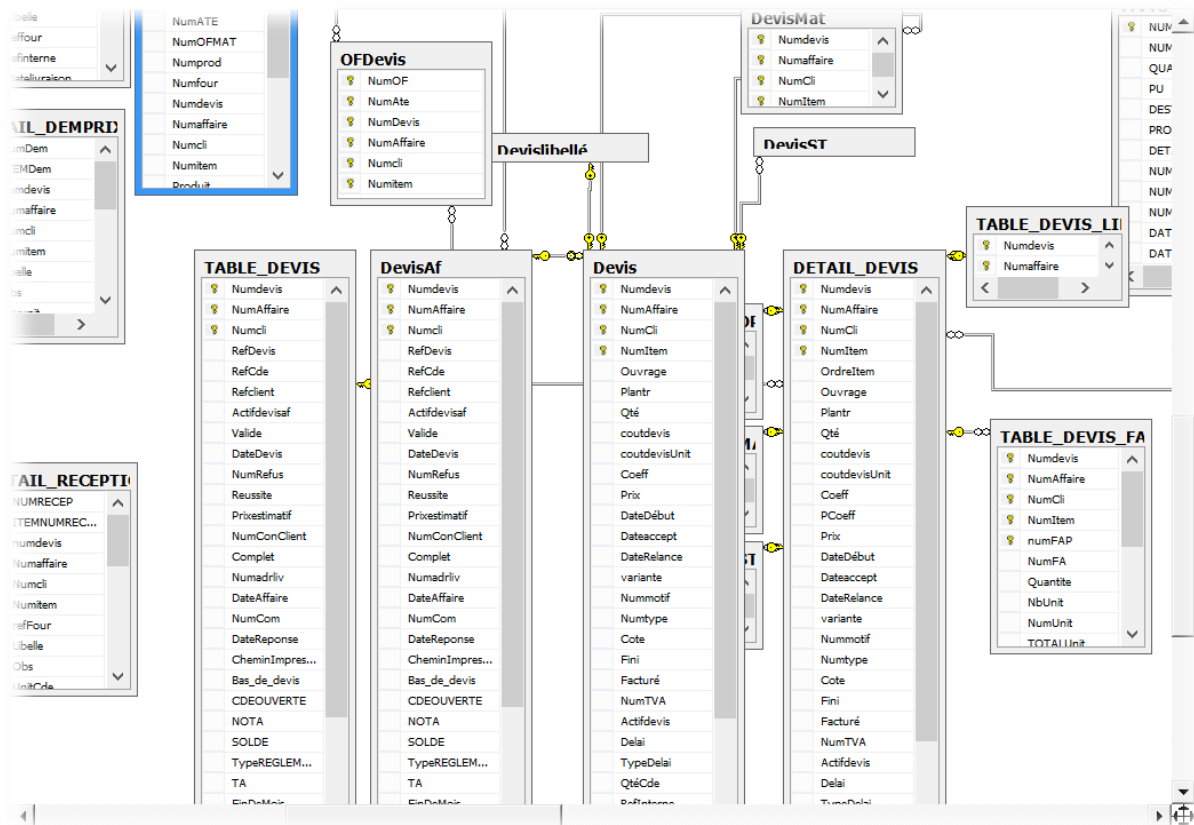
Pourtant il s'agit là d'un point majeur qui est indispensable à la gestion de toute entreprise : on ne doit pas avoir à chercher à 3 endroits différents pour consulter un dossier complet. Avoir un aperçu de l'ensemble des événements d'une affaire est indispensable à la bonne tenue de celle-ci.

Il m'a donc été confié dans un premier temps le développement de petits *addins* pour Microsoft Office qui permettraient de lier des mails directement aux différents éléments de la base de données de la GPAO. Avec le client HM Energie & Automatismes comme entreprise pilote, sans aucune date précise de livraison, simplement un vague « fin 2012 ».

Les limites de la GPAO : Sa base de données

Si le projet initial semblait simple (de par le *Sdk* fournis par Microsoft pour la réalisation d'*addin*), l'engouement suscité par les différents clients de Dune a posé un problème.

En effet, l'un des principaux avantages de la société est de proposer un service complet pour adapter la GPAO aux différents clients. Par exemple, ajouter des étapes supplémentaires dans la réalisation d'une affaire, ou en retirer. Les fonctionnements ne sont plus du tout les mêmes d'un client sur l'autre, et il était très difficile d'imaginer avoir à reprendre le code de l'*addin* en fonction des choix et des bases de chaque client.



Aperçu de Sorinter, la base de données de la GPAO de Dune

De plus, réalisée à l'origine sous Access, la base de données reprise par Arrakys souffre d'une dette technique colossale (noms des tables incohérents, duplication des données, 5 clés primaires par tables en moyenne, champs inutilisés...).

Si le module devait toucher l'ensemble du parc de client de Dune et satisfaire les besoins de chacun, il fallait forcément que celle-ci soit en mesure de s'adapter.

Objectifs

Dès lors, à défaut de pouvoir faire autrement, le projet Mentat avait largement dépassé le statut de simple module de jointure de mail pour devenir une Gestion Electronique de Document à part entière qui aurait l'avantage, en plus de s'exécuter directement au sein des applications de bureautique, de s'adapter aux bases de données clientes en passant par un assistant.

Technologies utilisées

Suivant la demande de la direction, le langage imposé était le c#, j'étais libre ensuite de choisir les différents éléments de la plateforme .NET pour mener à bien mon projet.

Ayant eu un aperçu des avantages apportés par la technologie WPF lors des *Techdays 2012*, et de son essor face au Winform en terme de conception d'interface pour Windows, j'ai

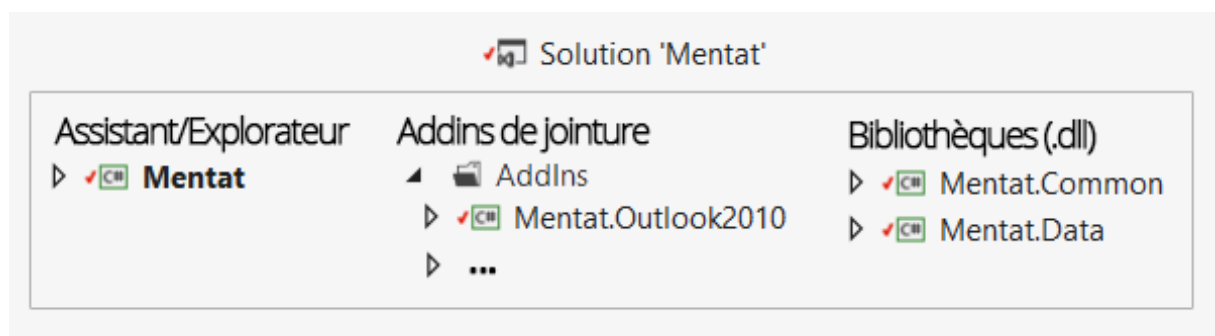
librement décidé de m’y essayer, le projet n’ayant pas de lourds besoins en interface, il était propice à l’expérimentation de nouveaux outils.

La partie données plus riche elle, ne me permettait pas de donner libre cours à ces expérimentations, et ce, malgré l’intérêt que je pouvais porter à la technologie WCF et à ses web services qui aurait pu scinder complètement la partie données des parties clientes, et permettre ainsi son exploitation par tout type de langages et OS en passant par des requêtes SOAP.

Même si cela pouvait limiter le projet par le futur aux environnements Microsoft, j’ai préféré reprendre l’accès aux données via *Entity Framework* (par schéma) sur lequel j’avais déjà accumulé une certaine expérience sur mes projets de BTS. Mais cela impose pour la partie cliente d’utiliser .Net pour bénéficier du même confort, ou alors envoyer des requêtes SQL brutes.

Organisation du projet

Le projet était constitué à terme de 4 sous-projets :



- **Mentat:** Il s’agit de l’explorateur de documents joint, il sert également au premier lancement à la configuration du poste.
- **Addins:** Liste des *addins* extérieurs qui ajoutent les documents à Mentat (depuis MS Office, Arrakys...).
- **Mentat.Common:** Bibliothèque de contrôles communs en *Winform*, permet dans les projets compatibles .NET d’intégrer directement le formulaire d’ajout d’un document.
- **Mentat.Data:** Bibliothèque d’accès à la base de Mentat, est nécessaire pour l’explorateur et pour les *addins*.

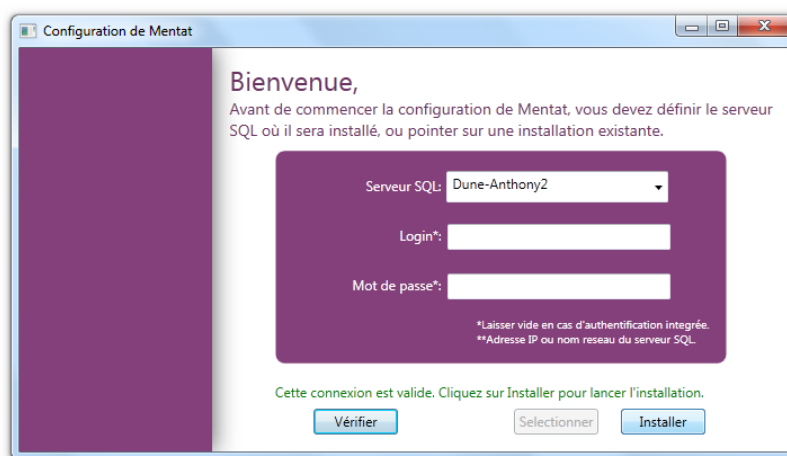
Comment se calque-t-on sur une BDD inconnue ?

Dans notre cas, il n’est pas nécessaire de connaître l’ensemble de la structure de la base de données cible. Je me limiterai donc au stricte minimum pour qu’à terme, l’utilisateur puisse lier un document sur une « arborescence » de plusieurs éléments de la base étrangère (par

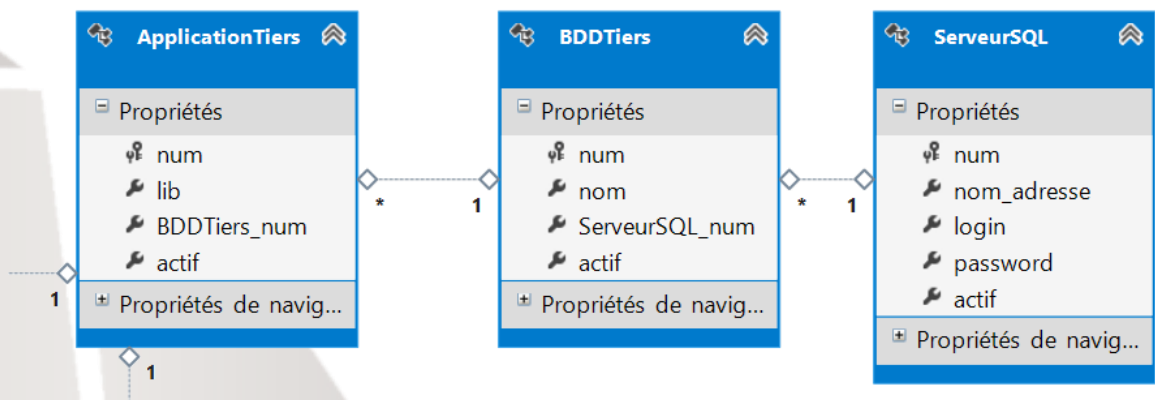
exemple assigné un mail à un client et une affaire, en filtrant la sélection des affaires sur client sélectionné).

La configuration suivant ces étapes :

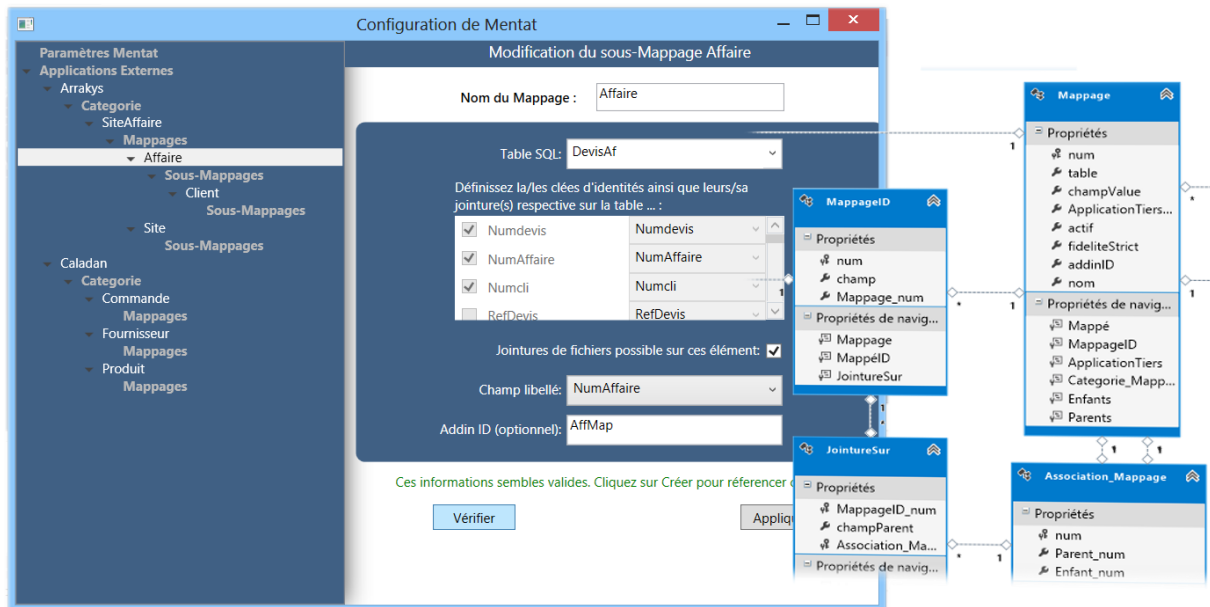
1. Configuration du serveur SQL de Mentat (on propose une installation automatique de la base si aucune instance Mentat n'est trouvée). Cette base servira à conserver les configurations des bases étrangères, une copie sommaire de leurs éléments, ainsi que les fichiers joints.



2. Sauvegarde des identifiants des bases de données externes pour chaque application tiers gérée par Mentat :



- Création de l'arborescence de « **mappage** » à partir des tables des bases de données précédemment renseigné. C'est à partir de ces mappages que les éléments vont être récupérés de la base externe pour pouvoir réaliser une jointure sur un fichier.



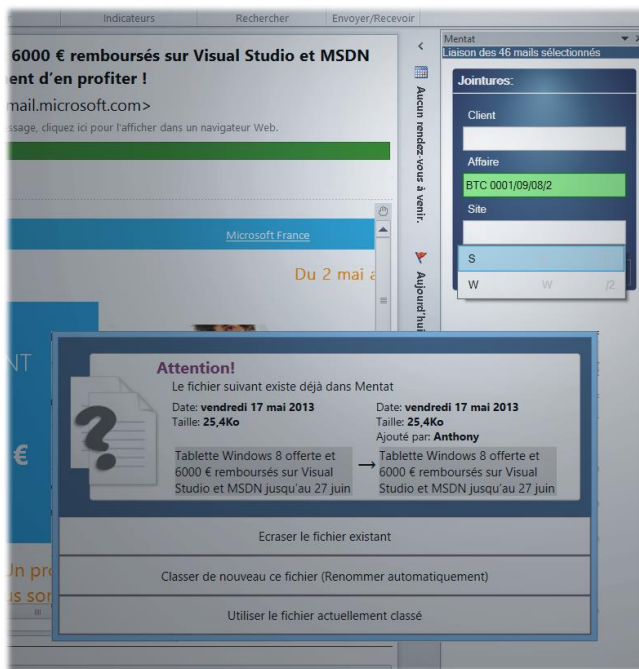
Ici, suivant l'arborescence de gauche, l'utilisateur peut lier un document depuis le client jusqu'au site de l'affaire.

N.B : je n'ai pas mentionné le rangement par catégorie qui permet de réutiliser les même mappages mais dans un ordre différents, de même que les « applications externes » qui servent à illustrer à l'utilisateur à partir de quelle base externe les éléments sont récupérés (ici, le logiciel de gestion Arrakys, mais aussi le logiciel de facturation Caladan).

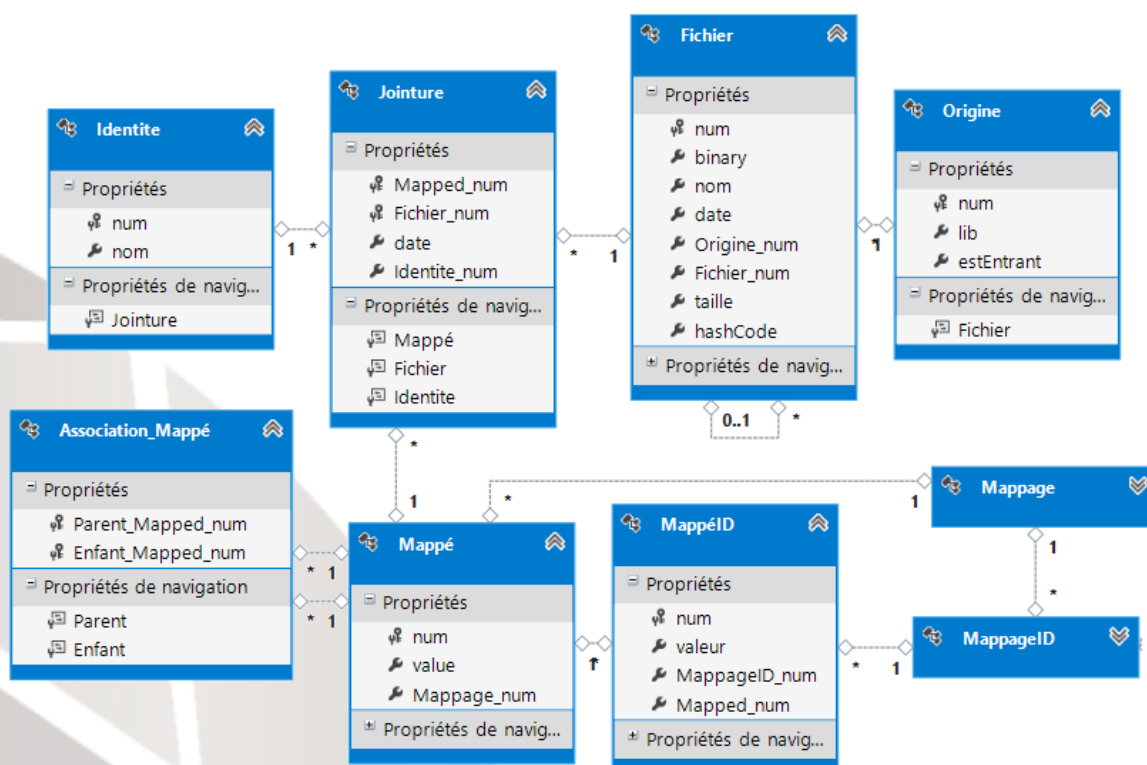
La liaison des fichiers aux éléments.

Une fois la configuration faite, Mentat en sait suffisamment sur la base extérieure pour pouvoir l'interroger. La jointure se fait alors depuis l'explorateur Mentat ou l'un des *addins*, suivant les arborescences précédemment créés, le tout en filtrant automatique chacun des champs.

Pour standardiser l'envoi des fichiers à Mentat, j'ai mis en place une bibliothèque .Net qui a été reprise dans les projets de Dune et dans les *addins* Office, qui propose directement un formulaire d'envoi réclamant uniquement le fichier à envoyer (les informations de configuration sont récupérées automatiquement depuis le fichier sur le poste client). Cela permet de ne pas répéter tous les contrôles nécessaires pour les jointures (calcul du *HashCode* du fichier, vérification des conflits...).



Derrière, pour chaque champ auquel le fichier (ici un mail à une affaire et un site) est joint, une copie (si elle n'existe pas déjà) simplifiée (clés primaires, libellé et relations) de l'élément du mappage est faite dans la base de Mentat... On parle d'un « **mappé** ».



Cette redondance m'est imposée pour simplifier et optimiser l'indexation des éléments des différentes tables/bases.

Dans le cas où le développeurs ne souhaite pas passer par l'interface proposé par la bibliothèque pour envoyer ses fichier dans Mentat (envoi automatique, ou charte graphique différente...), la bibliothèque propose de passer outre le formulaire, et permet travers des méthodes et un langage maison (« l'addin query ») de définir le(s) mappé(s) d'un fichier à joindre :

```
[Categorie.AddinID]Mappage.AddinID:Value;Mappage.AddinID;Mappage.AddinID:Value
```

Exemple : "[ARR_AFF]ARR_Cli:RM0001;ARR_AFF:KA00045"

Remarque : l'expression AddinID définit un champ librement choisi par le développeur de l'*addin* pour distinguer le mappage et ses identifiants. Il doit être renseigné dans la configuration de Mentat.

Le reste des informations sont contenues dans les paramètres des différentes méthodes d'ajout :

```
public JoinFileToMentat(string addinQuery, string Origine, bool isIncoming,
List<FileToJoin> files, FileCheckBehavior fileCheckBehavior, string header,
MentatContainer BDD, Window parentWindow)
```

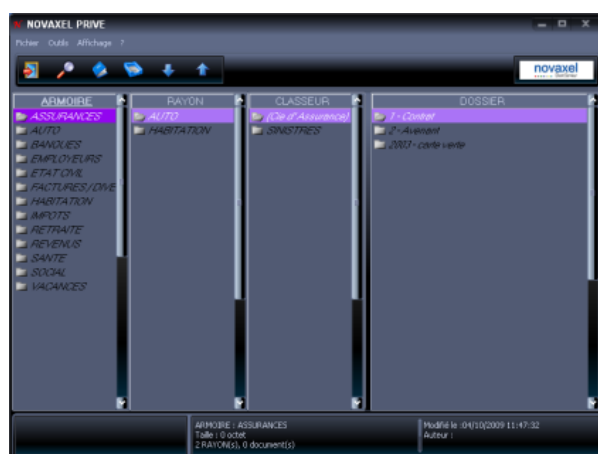
L'addin query est ensuite convertit en vMappage, et c'est à ce moment que l'échange a lieu entre la base externe et Mentat (en SQL brute) dans la méthode **QueryElementsFromTable** (Annexe 1)

Accès aux fichiers joints

Une fois les fichiers joints aux éléments des bases étrangères, il est possible de consulter la base de fichiers depuis l'explorateur Mentat.

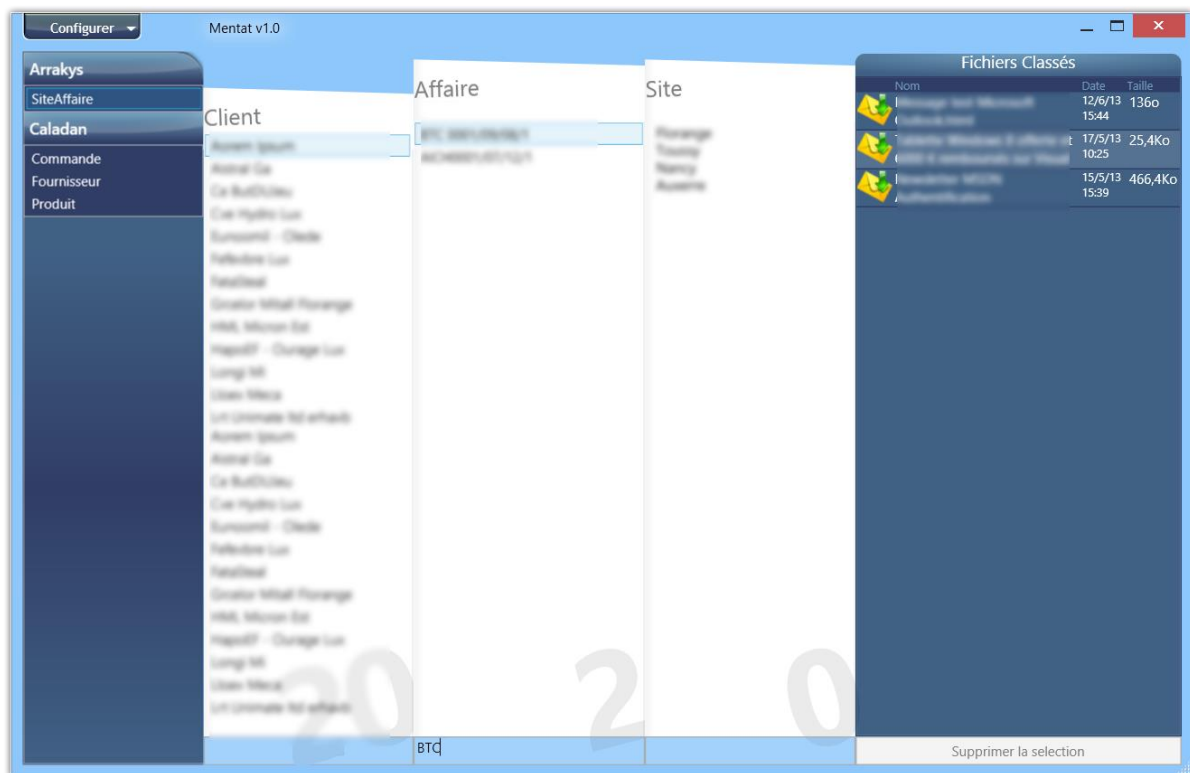
Pour cette partie je me suis librement inspiré de la GED française populaire, Novaxel.

L'accès aux fichiers se fait par des catégories fixes allant de l'armoire au classeur en passant par des rayons.



Aperçu de la GED lyonnaise Novaxel.

Dans mon cas, j'ai préféré jouer la carte de la flexibilité en reprenant ce fonctionnement mais en l'adaptant aux arborescences de mappage. L'utilisateur dispose d'un accès à tous les fichiers de la base (limité à 200 résultats), et se sert des panneaux pour affiner les résultats. Il peut ainsi consulter les fichiers sur plusieurs ensembles.



Ici l'utilisateur peut voir dans le panneau de droite l'ensemble des fichiers qui ont été joints à une affaire du client BTCXXXXX, tous site confondus

Limites du système

Malgré la remise en question perpétuel du cœur du système Mentat, il subsiste de lourds défauts :

- **La duplication des données** : C'est un souci d'espace, mais aussi de fidélité de la source : Si un mail est lié au Client « Mecalebeau », et que celui-ci est renommé en « McLebeau » dans la base externe, la modification ne seront pas propagés à la base de Mentat. De plus, la cohérence de la base de Mentat dépend énormément de celle sur laquelle elle se calque.
- **L'accès au données** : Microsoft envahit une grande part du projet, de la base de données à l'interface. S'il est vrai qu'une très grande majorité de nos clients utilisent Windows et SQL Server, mon choix d'accéder aux données par une bibliothèque .NET n'est pas des plus judicieux dans le cas de clients n'utilisant pas MS Office, car la majeure partie des équivalents gratuits ne peuvent en tirer parti. Dans le cas des extensions Mozilla par exemple (Thunderbird/Firefox), le langage mis en avant est le JavaScript, qui ne peut pas lire les bibliothèques .Net, mais uniquement envoyer des requêtes SQL brutes au serveur... Je regrette du coup de ne pas avoir tiré parti de WCF et de ses web services pour assurer un dialogue confortable avec tout type de langages.

Statut du projet/Evolutions possibles

Après 8 semaines de travaux, le projet a été mis en pause (fin 2012) suite à la réorganisation d'urgence sur le projet GestaAE. Le cœur de Mentat était opérationnel, ainsi que les *addin* de la GPAO et d'Outlook 2003+.

Parmi les évolutions envisagées, la priorité aurait été évidemment l'ajout d'autres *addin* (Word, Excel...), mais également le support d'autres types de base de données externes (pour l'instant réservé à *SQL Server*) comme *Oracle* ou *MySQL*.

Projet industriel : GestaAE

1-Contexte de création

Une mesure d'urgence

Fin 2012, l'annulation de certains gros projets conjointement avec des retards cumulés sur les paiements des organismes de formations amène la société à subir de très lourds problèmes de trésorerie.

C'est dans ce contexte d'urgence, et malgré les différents projets en place que la direction adopte une stratégie de redressement rapide dans la réalisation d'une application de facturation pour auto-entrepreneur en y mobilisant l'ensemble de l'effectif développeur.

Ce choix est justifié pour la direction par le fait qu'il s'agit des fondements de la spécialité des développeurs de DUNE (travaillant en général sur des solutions hautement plus complexes en termes de gestion).

Cette approche est également une tentative de forcer la transition vers une toute nouvelle solution GPAO de la société... En effet, il s'agit d'un réel leitmotiv tenu depuis plusieurs années par l'équipe : si le logiciel phare de la société est plus que jamais pourvu de fonctionnalités en tout genre, elle subit également les préjudices d'une lourde dette technique (développée en VB6, architecture de la base datant des premières versions d'Access...).

Choix préliminaire des technologies employées

Décidé dans l'urgence, le cahier des charges de l'application se résumera à une feuille Excel (Annexe 2).

Le temps est pris cotés développeurs de faire une étude de marché, et la concurrence se révèle assez lourde. En effet, plusieurs gros logiciels occupent là place depuis plusieurs années et proposent une très large gamme de fonctionnalités et dans certains cas, sont pleinement fonctionnels sans payer. Si cela remet complètement en question le choix de la direction qui comptait vendre une application bon marché à hauteur 30€/mois, cela nous impose surtout de trouver des moyens de faire face à cette concurrence, en faisant des choix technologiques que eux, ne peuvent plus faire...



On s'orientera sur une application native (et non web), afin de concurrencer plus facilement les solutions existantes nécessitant une connexion internet en continu. Attaché aux technologies Microsoft, et ayant abordé à plusieurs reprises le Framework .NET, il a été défini que nous utiliserions ces technologies et limitons ainsi le parc cible aux machines Windows (représentant toujours plus des ¾ des OS de bureau en 2013 selon Web AT).

Une réorganisation précoce

Si lors de sa mise sur pieds, l'ensemble de l'équipe de développeurs a été assignée au projet, cette mobilisation totale n'a cependant duré que très peu de temps. En effet, il a rapidement été mis en évidence que la vitesse de développement du projet n'était pas aussi bonne qu'escompté, notamment en raison des technologies choisies par le chef de projet (qui ne correspondait pas directement aux technologies utilisées jusque-là par l'équipe), mais également par une motivation en berne des membres de la société.

Aussi, au fur et à mesure des nouvelles offres clients, les différents membres de l'équipe ont été redirigés vers de nouveaux projets plus courts et plus rentables dans l'immédiat. A terme, l'ensemble de l'équipe n'aura été mobilisé que pour l'étude préliminaire et la modélisation de la base de données. Je suis donc rapidement revenu en autonomie totale sur ce projet.

2- Model

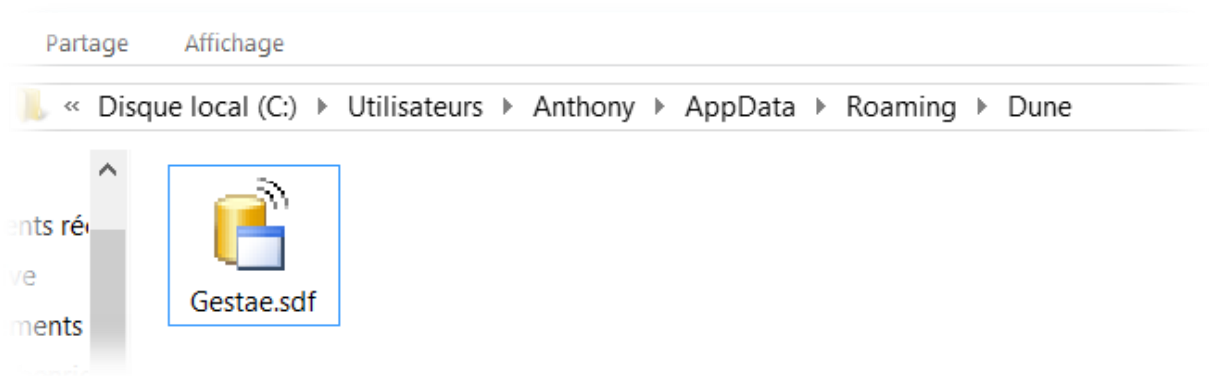
Il s'agit de la partir la plus simple pour aborder le projet. Déjà car elle exploite des techniques déjà maîtrisées par l'équipe, mais également, car c'est la suite logique d'une analyse. A terme, elle sert de moule pour les deux autres couches.

Etude préliminaire

L'application ciblant le grand public, nous ne disposons donc d'un contexte architectural minimal (à savoir un poste en local). Si certaines solutions commerciales existantes utilisent dans ce contexte directement le système de fichier de l'OS (via des fichiers .XML par exemple) pour la sauvegarde des données, dans notre cas, pour tirer pleinement partie des fonctionnalités de recherche et de statistiques, nous nous orienteront plutôt vers une base de données SQL embarquée. Dans ce cadre, la sélection se portera surtout sur des critères de rapidité et de simplicité d'installation.

Après étude, notre choix s'arrêtera sur l'utilisation d'SQL Compact Edition. En plus de respecter les contraintes précédentes, il apporte également quelques avantages propres :

- En tant que base de données Microsoft, elle est directement compatible avec les méthodes d'accès aux données (ORM) LinqToEntity de l'Entity Framework
- Elle est pensée pour être exécutée sur des postes clients ce qui lui permet entre autre de disposer d'une haute légèreté d'exécution.
- Même si elle se voit amputé de plusieurs fonctionnalités, ses fonctions de bases restent suffisantes dans le cadre de notre projet. On évite le surplus.
- Elle est déjà intégrée dans la majorité des systèmes Windows, et ne nécessite donc pas d'installation additionnelle.
- Sa structure peut être facilement mise à jour à travers un système de migrations.



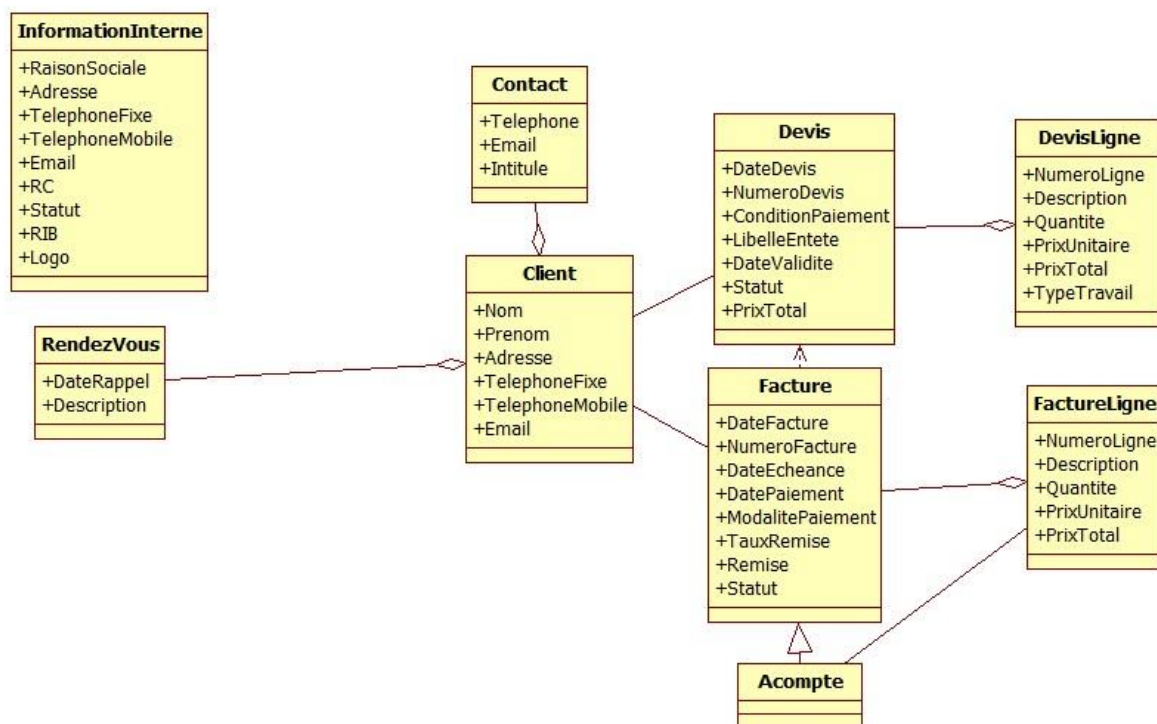
La base de données prend simplement la forme d'un fichier sur le poste « client »

On profitera également de cette phase d'études préliminaires pour définir quels seront nos méthodes d'accès. Si par le passé l'ensemble de l'équipe a surtout utilisé du requetage SQL en texte « plain », le risque est pris de passer par l'ORM de Microsoft Entity Framework. Ce choix est fait dans un souci d'économie du temps de production du code. Par ailleurs, si cette ORM est livré avec un concepteur graphique, on utilisera une autre méthode dite « code first » qui consiste à générer le contexte base de données depuis des classe C# plutôt que depuis des diagrammes ou une base existante, tout d'abord car cela permet l'utilisation du système de migration, mais également car en utilisation, il permet un degré de personnalisation supplémentaire et permet d'éviter certains bug liés à l'utilisation des outils lourds du concepteur graphique.

Construction de la base

Elle est réalisée au début de la phase développement, car on sait d'avance qu'il s'agira d'une structure simple sur laquelle pourra se baser pour l'ensemble des autres parties, qui seront plus riches.

Pour sa conception, on passera par un diagramme *UML* grossier, qui en plus de définir les différentes tables, permettra également de définir les différentes classes de la partie **Model** de l'application :



Remarque : GestaAE dans un souci de simplicité (elle ne s'adresse qu'à de petits entrepreneurs) n'intègre pas de gestion de produits.

Notre approche par l'Entity Framework code first comme méthode d'accès aux données nous impose de commencer la construction de la base de données par ses classes. Et c'est ensuite à partir de contexte objet qu'est générée la base de données relationnelle suivant le service choisis (ici, SQL server ce) lors d'une migration initiale.

L'ensemble des actions de migration sont réalisées directement à travers la console *nuget* de Visual Studio. On décide de construire la structure de la base de manière incrémentielle, c'est-à-dire qu'on part d'une base simple qui évoluera avant la première publication du projet, cela nous permettra de nous familiariser avec le system de migration (existant dans plusieurs autres ORM) qui permet de gérer les différentes versions de base données et les

mettre à jours (le cas échéant) directement depuis l'application sans intervention de l'utilisateur.

Une fois la migration initiale faite, on peut créer le contexte Entity Framework qui permettra d'accéder aux différentes tables de la base.



Le contexte Entity Framework de la base de GestaAE, un fois crée.

Accès à la base

Comme pour Hibernate et son HQL, Entity Framework permet de s'astreindre de la lourdeur et la mauvaise maintenabilité du SQL en utilisant LINQ depuis 2007 et le .NET Framework 3.5, un langage « unifié » (son utilisation est applicable autant aux BDD qu'à de simples collections génériques). Sa complète intégration dans l'IDE de Microsoft (à défaut de proposer une ouverture) apporte certaines fonctionnalités bien confortables tel que des aides à l'écriture (auto-complétion suivant les noms des tables/champs et des fonctions SQL utilisables) et au débogage, la requête n'est plus traitée comme une simple chaîne de caractères par le compilateur mais presque comme une suite d'instruction du langage.

Mais cela reste un langage converti dans un autre, aussi il peut y avoir certaine limite à une utilisation abusive et non surveillée du *linq to entities* car l'interprétation du traducteur peut différer de celle imaginée par le programmeur et en résulter des requêtes inutilement longues.

Aussi il est assez recommandé, (au moins au début de l'apprentissage du LINQ) de suivre les temps d'exécution des requêtes de l'application (avec le profiler de SQL Server par exemple), afin de relever celles qui sont trop lentes et afficher leur résultat SQL pour comprendre d'où vient le souci d'interprétation et essayer de trouver une requête plus propre.

Ce genre de situation, avec un minimum de raisonnement, n'est au final que peu fréquent... Et il faut peu de temps avant de ressentir le gain de temps procuré par l'utilisation de *linq to Entity*.

3- View

Pour la partie de l'interface utilisateur, il a été très rapidement décidé par le chef de projet (S. Delingette) d'utiliser le WPF suite à mes premier travaux réalisés du la GED Mentat.

Windows Presentation Foundation

Introduit en 2006 avec le .NET Framework 3.0, le WPF est la nouvelle spécification graphique de Microsoft en conception d'interface graphique. Elle se distingue notamment de la précédente spécification (appelée Winform), par son rendu exclusivement vectorielle, et son intégration d'un langage spécifique à la construction de l'interface et descendant du XML : le XAML (prononcé « zamel »).

```
<Grid>
    <TextBlock HorizontalAlignment="Stretch" FontSize="13" FontWeight="DemiBold"
Text="{Binding Facture.ModalitePaielement}" />

    <Rectangle HorizontalAlignment="Stretch" StrokeThickness="1"
StrokeDashArray="10" Stroke="{StaticResource bgood}" Margin="2" Visibility="{Binding
MainViewModel.IsEditable, Converter={StaticResource BoolToVisibility}}" />
</Grid>
```

Une organisation qui s'approche de ce qui existait dans le domaine du web, et qui a pour but, dans les grandes équipes, de scinder le travail des développeurs et des designers à travers des outils (MS Blend pour les designers, Visual Studio pour les développeurs) et des portions de programme dédiées (Interaction des données en C# pour le développeur, construction et animation de l'interface pour le designer).

Etude préliminaire

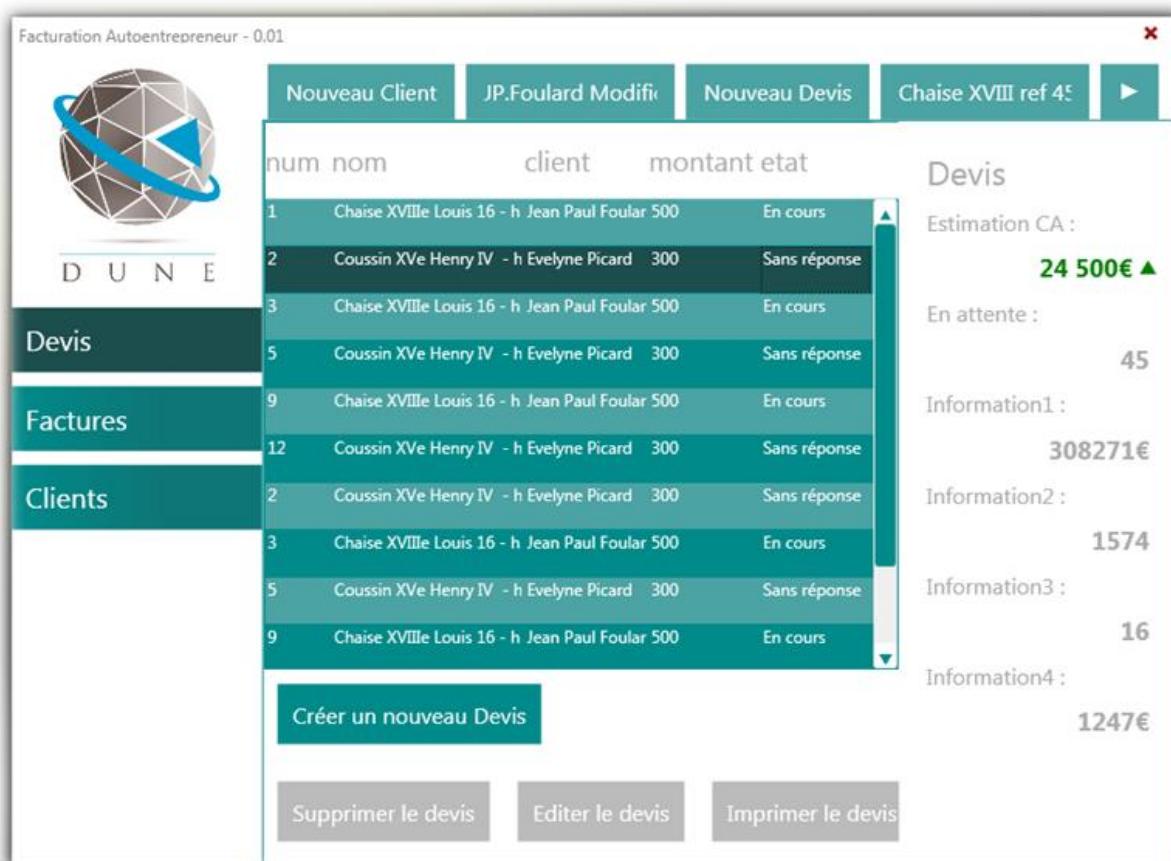
Le choix s'étant portée directement sur WPF, notamment pour son fort degré de personnalisation et son support assuré sur le long terme, l'étude préliminaire qui m'a été confié consistait surtout à proposer des croquis rapides de styles d'interfaces possibles (sans aucune interaction) afin de laisser à l'équipe le choix de la direction cosmétique à adopter.

J'ai donc proposé le lendemain les 4 essais suivants :



Les deux premiers se concentraient sur la partie ‘habillage’ en présentant les avantages esthétiques et fonctionnels d’opter pour du « borderless ». Tandis que le troisième reprenait plus les standards utilisés jusqu’à présent en Winform, avec la bibliothèque Infragistics.

Finalement, le choix de l’équipe s’est porté (à 7 contre 1) sur le 4ème exemple, qui mettait plus en avant les tendances récentes prises par Microsoft en termes de style :



Certaines remarques ont cependant été faites, notamment une demande de suppression des onglets (trop source d’incompréhension pour l’utilisateur, et améliorer la lisibilité de la liste)...

L’interface, un facteur important

Mis sur pied sur un contexte d’urgence, GestaAE n’a pas bénéficié d’une étude de marché approfondis, hors l’offre était déjà bien présente sur internet, avec des applications spécialisées qui tirent leurs richesses de leur ancienneté.

Ne pouvant directement rivaliser sur le plan des fonctionnalités, il m’a semblé important de tirer parti de cette jeunesse pour se démarquer. En effet, si les solutions existantes sont déjà bien implantées et pleinement fonctionnelles, elles souffrent peut être également de leur

âge. De par des technologies plus vieilles, et une conception qui ne suit plus les attentes d'aujourd'hui.

« Je veux du beau, je veux du simple »

Si jusqu'alors il était surtout demandé aux développeurs de s'en tenir à une interface riche et fonctionnelle dans le cadre de développements d'applications professionnelles, les dernières mouvances mettent l'accent sur la simplicité et le confort.

Un tournant pas toujours facile à comprendre pour des développeurs qui préfèrent avant tout le côté « pratique » et le côté savant des présentations brute.



Le site du W3C en 2009, avant sa refonte.

De plus, si GestaAE se veut être une application professionnelle, en s'adressant à des auto-entrepreneurs, elle s'adresse avant tout à une clientèle qui n'a que trop peu souvent de bonnes connaissances en matière de paperasse commerciale et administrative (un domaine qui plus est, souvent en contradiction totale avec leur métier de base).

Par la simplicité, GestaAE pourrait rendre triviale ces démarches, en n'hésitant pas à les alerter sur les événements annuels, ou simplement sur la signification de certains termes, et ce, toujours en mettant en avant les fonctions principales de l'application (pour éviter qu'elle soit noyée dans au milieu d'autres fonctions trop rarement utilisées).

Un bon moyen pour nous de renouveler le marché sans pouvoir proposer directement autant de fonctionnalité.

Exemple 1 : les DataTemplates

Le DataTemplate est un élément particulièrement puissant du WPF. Il donne la possibilité sur une large gamme de controls WPF d'intervenir sur la disposition des données qui y seront insérés. Tout ceci se fait exclusivement en injectant du XAML dans la propriété DataTemplate des controls :


```
<ListView ItemsSource="{Binding Source={StaticResource viewClients}}">
    <ListView.ItemTemplate>
        <DataTemplate>
            <!-- DEBUT DU DATATEMPLATE-->

            <Rectangle Fill="Red"/>

            <!-- FIN DU DATATEMPLATE-->
        </DataTemplate>
    </ListView.ItemTemplate>
</ListView>
```

Ici, pour chaque éléments de viewClients il y aura uniquement un rectangle rouge d'affiché.

Ainsi, si dans un premier temps l'ensemble de la personnalisation passait simplement par des couleurs comme ici :

num	nom	client	montant	etat
1	Chaise XVlle Louis 16 - h	Jean Paul Foular	500	En cours
2	Coussin XVe Henry IV - h	Evelyne Picard	300	Sans réponse
3	Chaise XVlle Louis 16 - h	Jean Paul Foular	500	En cours
5	Coussin XVe Henry IV - h	Evelyne Picard	300	Sans réponse

L'utilisation du *DataBinding* (l'action de lier une propriété du ModelView à la valeur d'un control) et l'utilisation de *Converters* a permis par la suite de dynamiser la disposition et l'interactivité des controls :

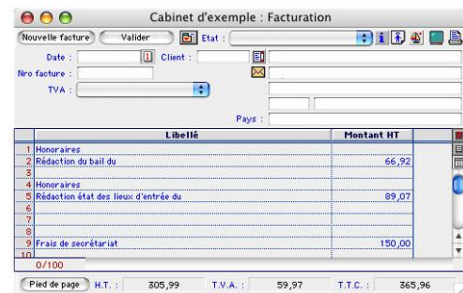
Devis				En-cours Tout		Rechercher	
	D201300014	Date de création: 04/06/13 Fin de Validité: 04/09/13	Davina Barthou	62,34 €			
	D201300015	Date de création: 04/06/13 Fin de Validité: 04/09/13	Davina Barthou	46,74 €			
	D201300016	Date de création: 04/06/13 Fin de Validité: 04/09/13	Antonella BelRillette	26,17 €			
	D201300017	Date de création: 04/06/13 Fin de Validité: 04/09/13	Cathou Herbier	38,76 €			
	D201300018	Date de création: 04/06/13 Fin de Validité: 04/09/13	Davina Barthou	31,04 €			

Désormais, l'affichage met en avant les données importantes de chaque devis et adapte ses couleurs suivant les statuts

Exemple 2 : l'édition WYSIWYG

Assez récurrente depuis le début des années 2000, l'expression *What You See Is What You Get* met en avant le concept skeumorphique qui préfère montrer l'équivalent réel à l'utilisateur pour qu'il prenne plus facilement conscience de ce qu'il est en train de faire et dans quel but. Par exemple, afficher un jukebox en fond sur l'interface d'un lecteur MP3.

Dans notre cas, si beaucoup de logiciels sont adeptes des champs sans sens à rallonge (pour généralement générer au final un état qui ne suit même pas l'ordre des propriétés remplies) j'ai préféré dans le cas de GestaAE d'assimiler la rédaction et le remplissage des documents de devis et de facturation à du traitement texte standard.



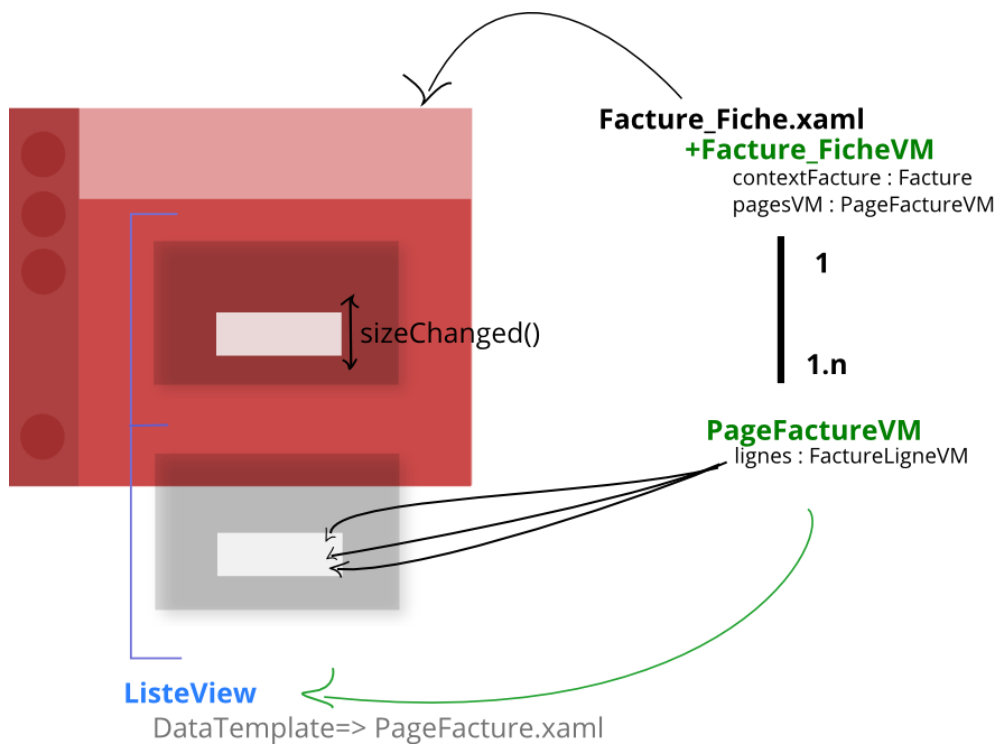
Pour Ciel compta, ceci est une facture.

L'utilisateur doit comprendre ce qu'il est en train de faire et ne doit pas avoir besoin d'attendre l'impression du document pour connaître son rendu.

En outre cette idée a également été exploitée car il fallait faire un choix sur l'outil à utiliser pour générer les documents. Et si jusqu'à présent, la société se servait de Crystal Report (un outil lourd de création d'états), son utilisation ici posait problème car cela nécessitait une installation supplémentaire pour l'utilisateur.

Toutefois, et si cela est très peu documenté sur internet, il est tout à fait possible d'imprimer un rendu WPF, et donc en conséquence, de construire ses documents directement depuis l'interface.

Une double aubaine qui a simplement nécessité que je construisse la logique de page multiple. Pour cela, je passe par un DataTemplate pour chaque page que je nourris avec un ViewModel de page :



Fonctionnement sommaire de l'éditeur de document

Facture F201300002

STATUT
 Rédaction
 Envoyée
 Payée

À propos

page 1/1

Dune
 25, rue de la poupée qui tousse
 54700 - Pont à Mousson
 06.118 218 06.118.000
 Allo@Jesuispasla.fr

FACTURE
 Du mardi 4 juin 2013 N° F201300002

Rocha Nina
 Pas la
 54321 - ARTE / MEURTHE

Modalités de paiement..?
 Échéance: **vendredi 19 juillet 2013**

Entête de la facture..?

Archiver?

LÉGENDE
 Champ optionnel

N°	DESCRIPTION	TYPE	QUANTITÉ	P.UNITAIRE	TOTAL
1	Rintintin Sex Machine		1	16,15 €	16,15 €
2	Bals de Feu		1	0,04 €	0,04 €
3	Documentaire animalier : Artic Fox		1	33,65 €	33,65 €

Rendu final de l'éditeur

Exemple 3 : Maximiser le confort d'utilisation

Améliorer le confort d'utilisation d'un logiciel non-ludique peut sembler anodin (dans des projets standards, on veut simplement minimiser le temps de production de l'utilisateur et réduire le coût de la livraison du logiciel), pourtant dans le cadre d'une application « grand public » il peut être intéressant de mieux travailler ce facteur pour se démarquer.

Cela passe par des points simples et rapides à mettre en place qui sont trop rarement appliqués dans des projets où les contraintes de temps sont de plus en plus importantes.

Par exemple, la saisie et la sélection automatique de champs dans un formulaire. Si il n'est pas rare de voir des champs date pré-remplis, beaucoup de

développeurs prennent rarement le temps de définir la focalisation initiale sur un élément du formulaire, même si celui-ci est susceptible d'être ouvert des milliers de fois par les différents utilisateurs.

Lors de la création d'un document, une sélection de clients fréquents est directement proposée

De plus, comme pour toute information de document, les boutons et autres menu doivent être mis en avant selon leur importance et leur fréquence d'utilisation. Dans le cas de GestaAE, le ComboBox de changement de statut était celui majoritairement utilisé dans l'éditeur de facture/devis : il suivait toutes les étapes des documents jusqu'à leur clôture, mais était noyé au milieu de bouton tel que l'impression, l'envoi par mail et l'ajout de ligne.



A gauche, le ComboBox tel qu'il était, alors noyé, à droite la version avec les boutons de statut mis en avant.

Lors du passage de l'édition des documents en WYSIWYG, j'ai factorisé les boutons d'envoi par mail, d'impression, et d'enregistrement dans une fenêtre qui s'affichait uniquement lorsque le statut passait en « envoyé » et ai mis en avant les boutons de statut.

4- ~~Controller~~ ViewModel

Parmi les patrons les plus utilisés, MVC se démarque par son large champ de langages sur lesquels il peut être appliqué. Toutefois, certaines contraintes inhérentes aux fonctionnalités du XAML, impose certaines modifications à ce patron.

Des difficultés rencontrées assez tard

L'un des principaux torts lors des prémices du projet qui s'orientait alors essentiellement sur la partie **Model**, et d'avoir déclaré les traitements de données dans cette dite partie, à travers de nombreuses méthodes associées à des objets DAO (Data Access Object).

Outre le fait que le modèle MVC n'est pas respecté, cette organisation pose également problème à l'échelle de l'interface en raison de certains caprices du WPF. Celui-ci est en effet conçu pour dialoguer directement avec les données (à l'instar du Template MVC) mais uniquement à travers des propriétés d'objet, et non des noms de méthodes.

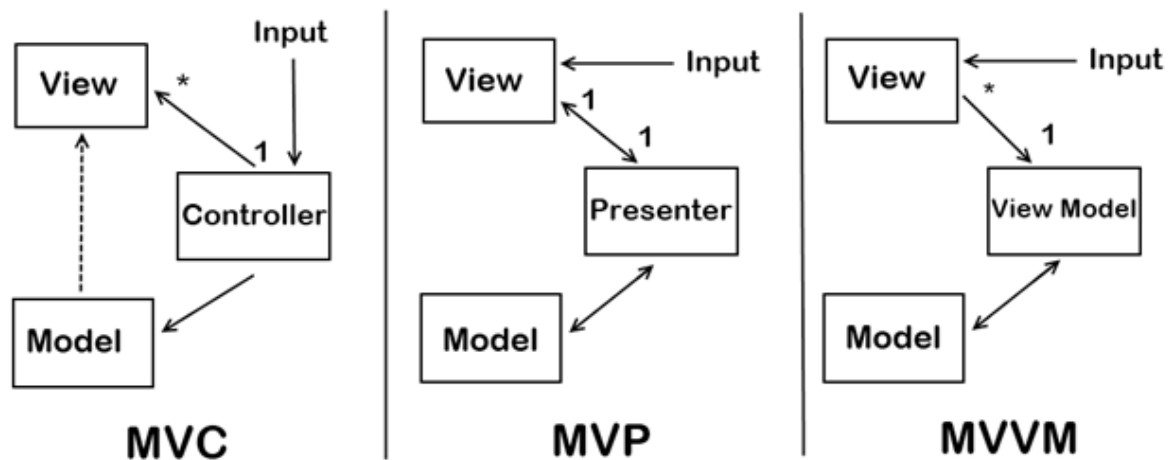
De plus, certains mécanismes automatiques liés à l'utilisation de l'Entity Framework, répercutaient les modifications sans intervention de notre part, dès lors que l'objet de la couche modèle était modifié... Il était donc nécessaire de passer par un objet intermédiaire équivalent.

De leur côté, les objets DAO, même si ils ont l'avantage de « standardiser » les accès aux données en les factorisant par méthodes, deviennent beaucoup trop superflus, l'accès aux données se voulant suffisamment trivial avec le contexte d'Entity Framework et l'utilisation de LINQ (pas besoin de gérer de connexion, ni de faire de conversion d'objet). Les accès DAO peuvent de plus se montrer plus rigides et moins clairs, voici un comparatif :

	cXEF = Requête LINQ	cXDao = Equivalent Dao
<code>var c1Ef = GestaeDb.Db.Clients;</code>	<code>var c1Dao = DaoClient.GetAllClients();</code>	
<code>var c2Ef = GestaeDb.Db.Clients.Find(42);</code>	<code>var c2Dao = DaoClient.GetClient(42);</code>	
<code>var c3Ef = from c in GestaeDb.Db.Clients where c.Contacts.Any(co => co.Telephone != null) select c;</code>		
<code>var c3Dao = DaoClient.<u>GetClientQuiOntAuMoinsUnTelDansContact()</u>;</code>		

Ajoutons à cela le fait que, par héritage du Winform (et du modèle MVP), les entrées de l'utilisateur interviennent directement sur l'interface (et non directement sur un quelconque Controller), et c'est elle qui se charge d'avertir le Controller d'une modification de valeur lorsqu'elle est faite par l'utilisateur, cela remet clairement en cause le modèle MVC et

apporte de lourdes difficultés en matière de traitement des données suivant les actions de l'utilisateur...



Le patron architectural MVVM introduit par Martin Fowler en 2005, largement promu par Microsoft et massivement adopté par la communauté WPF et Silverlight corrige l'ensemble des problèmes cités plus haut.

La partie **Controller** est remplacée par la partie **ViewModel**, qui tire son nom du réel pont qu'elle crée entre la vue et les données, elle devient le passage obligatoire pour tous les échanges et les traitements.

On l'écrit sur un fichier à part de la partie vue (qui possède sa propre page de code brut, mais qu'on réserve uniquement aux traitements cosmétique) et de la partie model.

Si le model semble assez anarchique dans un premier temps, il sied pourtant parfaitement aux contraintes du WPF, et son apprentissage est presque naturel. Pour simplifier son utilisation (qui requiert malgré tout pas mal de code), il est recommandé de passer par des *toolkits* spécialisés, qui se chargent d'ajouter le code nécessaire (suivant des *snippets*) pour assurer les liaisons entre les différentes couches.

Passage à MVVM Light

Les *toolkits* sont assez nombreux et si certains proposent une très large gamme de fonctionnalités, tous recommandent d'utiliser leurs outils sur un nouveau projet (car certaines mécaniques doivent être mises en place dès le début pour être pleinement fonctionnelles), aussi proposent-ils généralement des assistants de création de projet s'occupant d'alourdir chaque nouveau projet d'une quantité de code automatique.

Dans notre cas, le projet étant déjà bien avancé, et l'utilisation de MVVM devenue inévitable, j'ai préféré opter pour le plus léger et le plus flexible pour être certains que l'adaptation progressive à ce nouveau Template se passe dans les meilleures conditions. J'ai donc choisi **MVVM Light**.

S'il est vrai que, ne pas avoir utilisé MVVM Light au début du projet nous a retiré certaines fonctionnalités telles que le *Messenger*, un objet centralisant l'ensemble des messages envoyé de la View au ViewModel (on parle de « **command** »), nous avons quand même pu exploiter une large partie de ses générations automatiques de code et ses classes abstraites :

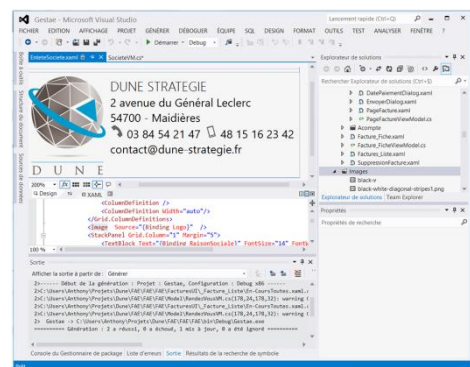
```
string _RaisonSociale;
public string RaisonSociale {
    get { return _RaisonSociale; }
    set { _RaisonSociale = value; RaisePropertyChanged("RaisonSociale"); }
}
```

La création d'une propriété bindable est une tâche redondante, son automatiser est un réel bénéfice.

```
public class SocieteVM : ViewModelBase
{
```

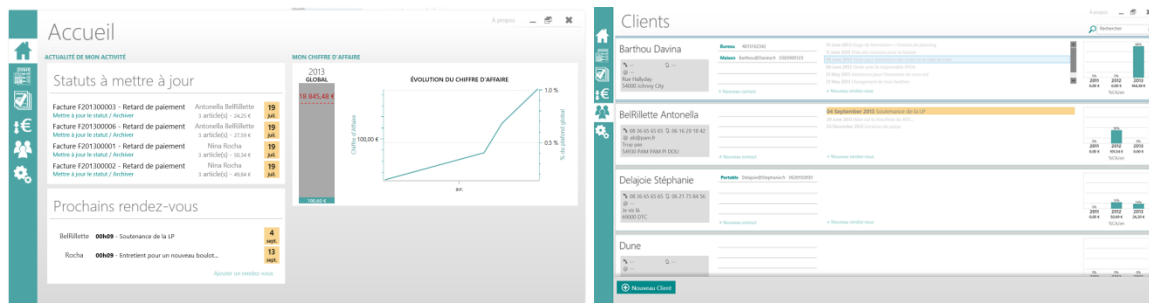
```
    public SocieteVM()
    {
        if (IsInDesignMode)
        {
            this.Adresse = "2 avenue du Général Leclerc";
            this.CP = "54700";
            this.Email = "contact@dune-strategie.fr";
            this.NumeroLicence = "qvny45ex7pdx4";
            this.RaisonSociale = "DUNE STRATEGIE";
            this.TelephoneFixe = "0384542147";
            this.TelephoneMobile = "4815162342";
            this.Ville = "Maidières";
            this.DateCreation = DateTime.Now.AddMonths(-2);
            this.Logo = new BitmapImage(
                new Uri("../Images/logo_dune_cmjn.png",
                    UriKind.RelativeOrAbsolute));
            this.StatutExclusif = false;
        }
        else
        {
            ...
        }
    }
}
```

Une propriété héritée de la *ViewModelBase* de MVVM Light permet de définir des valeurs d'exemple dans le designer.



5- Conclusion

Stopper en même temps que la déclaration de liquidation judiciaire par la direction (mi-juin 2013), le développement de GestaAE que j'entretenais seul depuis plusieurs mois pouvait être juger comme étant en moitié de chemin, car si une grande partie de l'application était fonctionnelle, le tout manquait encore de stabilité pour être encore vendu (je manque d'expérience pour la partie test), de même que le système de distribution (une licence renouvelable qui bloquait l'application après expiration) n'avait pas été terminé en raison de la démission d'un collègue plus tôt dans l'année.



Une fin assez triste donc pour une application qui bénéficiait malgré tout de certains atouts sur le marché, tel que son intuitivité et son fonctionnement local.

Mais à défaut d'avoir déboulé sur quelque chose de réellement concret, ce projet m'aura quand même permis d'améliorer grandement mes compétences en matière de conception et d'interaction d'interface, ainsi que de gestion de bases de données embarquées des technologies Microsoft. Acquis qui peuvent sembler bien spécifiques mais qui serviront sans doute comme une bonne base dans mes projets futurs qu'ils soient ou non construits sur la plateforme .NET.

En outre, ça restera également pour moi une excellente expérience de travail sous la contrainte du temps, et du manque d'éléments fourni pour débiter, de même qu'une énorme opportunité de réfléchir sur les possibilités de pénétration d'une application grand public sur un marché qui peut sembler saturé au premier regard.

Conclusion de l'année

Ces derniers mois marquent la fin de mon année à l'université mais également la fin de mes trois années enrichissantes de collaborations avec Dune.

Si d'un premier abord il peut sembler assez dure de corréler ma formation et l'activité professionnelle que j'ai exercée, la réalité est tout autre. En effet en plus d'apporter une légère polyvalence à mes compétences, une grande partie des notions des cours étudiées sous Java ou Oracle m'ont permis de mieux comprendre certains mécanismes de la plateforme .NET.

De plus, désormais libre de toute contrainte technologique et en attendant de trouver un nouvel employeur, je prends de nouveau plaisir à découvrir de-ci de-là les différentes technologies qui font appel aux dits cours dans le but de satisfaire ma curiosité et d'améliorer mes compétences.

Aussi, cette formation aura également été pour moi l'occasion de découvrir de nouveaux domaines tels que la gestion de projet et les avantages des méthodes agiles, ainsi que les web services à travers SOAP et xml-rpc (que j'ai regretté ne pas avoir pu/su implémenté au début de projet Mentat).

De même en plus d'avoir acquis une expérience non-négligeable sur le contexte assez bancal dans lequel mon dernier projet a pu être mené, la situation critique de l'entreprise cette dernière année m'aura également permis d'être témoins des étapes et conflits précédant la liquidation, et d'y être mieux préparé dans le futur.

Pour conclure, même si j'en tire les avantages cités plus haut, je regrette tout de même de ne pas savoir la société continuer à croître, l'ayant rejoint alors qu'elle était toute jeune (moins de deux ans d'existence), et avec les promesses d'embauche qui m'avaient été faites perdues...

Annexes

Annexe 1 : L'accès aux données étrangères par Mentat (p.19)

```

/// <summary>
/// Recupere les MAX_QUERIED_ELEM éléments de la table de ce mappage suivant
/// le filtre defini ainsi qu'en utilisant les affinements
/// </summary>
/// <param name="vMappages">Filtre enfants à appliquer</param>
/// <param name="matchAllvMap">Vrais si les elements retourné doivent validé
Tous les conditions dans les vMappage </param>
/// <param name="filtrePrincipal">Filtre à appliquer sur le champ value du
mappage</param>
/// <returns>Elements Tiers du mappage filtrés</returns>
/// <example>
/// --Recupéré depuis DefineStandardQuery_QueryElementFromTable()
/// USE [BDD]
/// SELECT DISTINCT top [MAX_QUERIED_ELEMENT]
[Mappage.table].[Mappage.champValue] , [Mappage.table].[Mappage.MappageID1.champ] ,
[Mappage.table].[Mappage.MappageID2.champ] (...)
/// FROM [Mappage.table]
///
/// --Filtrage principal appliqué sur le champValue du mappage
/// WHERE [Mappage.champValue] LIKE %[filtrePrincipal]%
///
/// --Filtrages annexes avec les MappageValue (sur les tables autours ou la
meme), soit en les matchant tous, soit en n'en matchant un ou plusieurs
/// AND ( [VMappage1.Mappage.champValue] LIKE %[VMappage1.Value]% OR|AND
[VMappage2.Mappage.champValue] LIKE %[VMappage2.Value]% )
///
/// --Groupage de la réponse
/// GROUP BY [Mappage.champValue]
/// </example>
public List<BDDTiersElement> QueryElementsFromTable(List<VMappage> vMappages,
bool matchAllvMap = true, string filtrePrincipal="")
{
    string query = DefineStandardQuery_QueryElementFromTable();
    string whereStatement = "";
    foreach (var vMappage in vMappages)
    {
        if (!vMappage.mappage.IsMappageDeValeur && !this.IsMappageDeValeur )
        {
            //L'association n'existe pas
            var ass = this.Enfants.Where(e => e.Enfant ==
vMappage.mappage).FirstOrDefault();
            if (ass == null)
                ass = this.Parents.Where(p => p.Parent ==
vMappage.mappage).FirstOrDefault();
            if (!this.Enfants.Any(e => e.Enfant == vMappage.mappage) &&
!this.Parents.Any(e => e.Parent == vMappage.mappage))
                throw new InvalidDataException("L'association entre [" +
vMappage.mappage + "] et [" + this + "] n'existe pas dans la configuration Mentat.");

            var t = vMappage.mappage.table;

```



```

//TODO: Optimiser cette partie
var idsAsParent = vMappage.mappage.MappageID.Where(vid =>
vid.JointureSur.Any(j => j.Association_Mappage == ass));
var idsAsEnfant = MappageID.Where(vid => vid.JointureSur.Any(j =>
j.Association_Mappage == ass));

query += " INNER JOIN " + t + ' ' + t +
vMappages.IndexOf(vMappage) + " ON ";
foreach (var id in idsAsParent)
{
    var j = id.JointureSur.Where(_j => _j.Association_Mappage ==
ass).First();
    query += t + vMappages.IndexOf(vMappage) + '.' + id.champ + "
= " + this.table + '.' + j.champParent + " AND ";
}
foreach (var id in idsAsEnfant)
{
    var j = id.JointureSur.Where(_j => _j.Association_Mappage ==
ass).First();
    query += this.table + '.' + id.champ + " = " + t +
vMappages.IndexOf(vMappage) + '.' + j.champParent + " AND ";
}
if (query.Substring(query.Length - 5, 5) == " AND ")
    query = query.Substring(0, query.Length - 5);
// fin de partie à optimiser
whereStatement += ((matchAllvMap || whereStatement=="")?" AND ": "
OR ")
    + ((!matchAllvMap && whereStatement == "") ? "(" : "")
    + t + vMappages.IndexOf(vMappage) + '.' +
vMappage.mappage.champValue + " LIKE '%" + vMappage.value + "%'";
}
else
{
    whereStatement += (matchAllvMap || whereStatement == "") ? " AND "
: " OR " + table + '.' + vMappage.mappage.champValue + " LIKE '%" +
vMappage.value + "%'";
}
}
query += " WHERE " + this.table+'.'+this.champValue + " LIKE '%" +
filtrePrincipal + "%' "
+ whereStatement + ((!matchAllvMap && (vMappages.Count!=0) ) ? ")" :
""");

if (IsMappageDeValeur)
    query += " GROUP BY " + this.champValue + ' ';

return Execute_QueryElementsFromTable(query);
}

```

Annexe 2 : Cahier des charges de GestaAE (p.22)

Etapes

1 Trouver 1 base

XML
SQL Compact
SQLite

Developpement de l'application

Gestion des licences et des dates limites

Ensembles des documents administratifs

2 Site internet

Paieement internet + renouvellement des licences

Documentation utilisateur

3 Référencement du site

Lobying auprès de la chambre des métiers