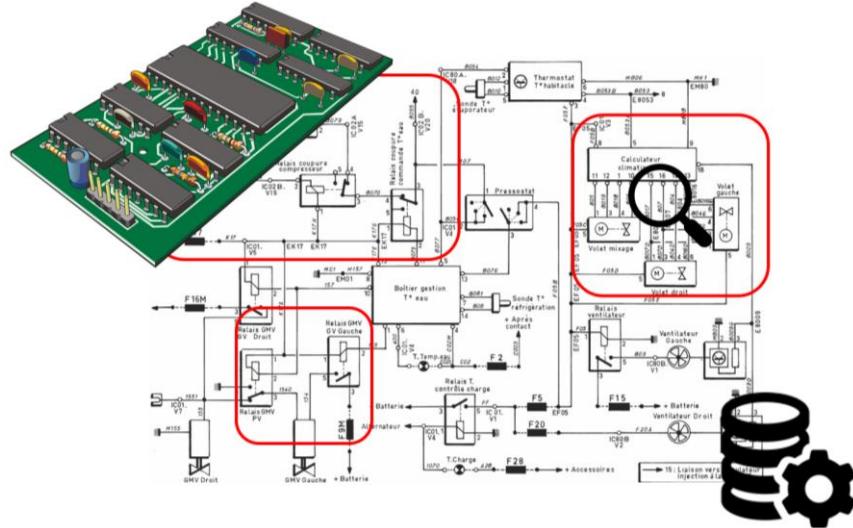


RAPPORT DE STAGE

CREATION D'UNE BANQUE DE SCHEMAS ELECTRONIQUES



Tuteur ou Maitre de Stage :

Vinot Nicolas

Signature

RAPPORT DE STAGE

CREATION D'UNE BANQUE DE SCHEMAS ELECTRONIQUES



Tuteur ou Maitre de Stage :
Vinot Nicolas

Remerciements

Je tiens à remercier toutes les personnes qui ont contribué au succès de mon stage et qui m'ont accompagné tout au long de celui-ci.

Tout d'abord, je remercie tout le corps enseignant de l'IUT Paul Sabatier pour cette année un peu spéciale. Une année qui restera gravée dans ma mémoire tant par la qualité des enseignants que par l'originalité des camarades de promotions. Merci à mon tuteur de stage M Broisin Julien pour les différents conseils lors de la conception du rapport de stage.

Merci à mon tracteur et mes parents, sans les fous, je n'aurai jamais vu Héricourt et la société A2E, le stage n'aurait pas eu lieu.

Je note l'accueil chaleureux par l'ensemble du personnel d'A2E. Je remercie Jean-Michel Miclo, Cédric Dubois, Raphaël Marc et le disciple Florian Remy pour m'avoir supporté dans l'équipe et pour avoir testé la super application. Je remercie encore plus particulièrement Nicolas Vinot, mon maître de stage et copain dans la vie. Merci pour la Leffe, les biscuits, les matchs de football, ... J'espère avoir été à la hauteur de tes attentes.

Merci à mon copain Polo pour mettre du beurre sur les épinards. Sans lui, j'habiterai encore parking Auchan.

Table des matières

1.	Introduction	1
2.	Présentation de la société	2
2.1.	Structure de l'entreprise	2
2.2.	L'atelier de production	3
2.2.1.	Cartes électroniques.....	3
2.2.1.	Câbles.....	3
2.2.2.	Intégration	3
3.	Le cahier des charges.....	4
3.1.	Mise en contexte du stage	4
3.2.	Analyse du besoin.....	4
3.2.1.	Objectifs du projet et enjeux.....	4
3.2.2.	Expression du besoin et validation	4
3.3.	Organisation mise en place	5
3.3.1.	Méthodologie AGILE (SCRUM)	5
3.3.2.	Equipe Agile	6
3.3.3.	Planification	6
3.4.	Analyse fonctionnelle	7
3.4.1.	Recherche des fonctions de service	7
3.4.2.	Caractérisation des fonctions de service.....	8
4.	Conception du produit demandé	9
4.1.	Base de données.....	9
4.1.1.	Informations à exploiter	9
4.1.2.	Modèle conceptuel de données (MCD).....	10
4.1.3.	Création de la BDD.....	11
4.2.	Conception UML.....	11
4.2.1.	Architecture mise en place (N-tiers).....	11
4.2.2.	Cas d'utilisation	11
4.2.3.	Diagramme de séquence	14
4.2.4.	Diagramme de classe métier (couche BEL)	16
4.2.5.	Description de la dynamique de l'interface : Diagramme d'état transitions	16
4.3.	Modélisation de l'Interface Homme Machine (IHM)	17
5.	Résultats obtenus	19
5.1.	Algorithme de la fonction de recherche.....	19
5.2.	L'application	19
5.3.	Les tests	25
5.4.	La documentation.....	25

6.	Bilan	26
6.1.	Bilan technique	26
6.2.	Bilan humain.....	26
7.	Conclusion	27
8.	Bibliographie.....	28
	Table des illustrations.....	29
	Annexe 1 – Audit de fonctionnement	1
	Annexe 2 – Modèles conceptuels de données.....	2
	Annexe 3 – Requêtes de la création des tables de la base de données.....	3
	Annexe 4 – Procédures stockées et fonctions pour le calcul du taux de correspondance	5
	Annexe 5 – Code C# de la couche métier BEL	9
	Annexe 6 – Code C# de la couche DAL pour la classe FonctionElectronique.....	12
	Annexe 7 – Code C# de la couche BAL pour la classe FonctionElectronique	17
	Annexe 8 – Enchainement des fenêtres de l'IHM	19

1. Introduction

Après une certaine expérience professionnelle dans le domaine de la logistique dans une entreprise de fabrication de meuble, j'ai décidé de me spécialiser dans le domaine informatique. Dans cette optique, j'ai eu l'opportunité de préparer un DUT informatique orienté développement, en formation accélérée (communément appelée « année spéciale ») à l'IUT Informatique Paul Sabatier situé à Toulouse. L'année aboutissant, un stage doit être effectué en entreprise afin de valider le diplôme d'université technologique en informatique.

J'ai de nouveau la chance de connaître un contexte d'entreprise dans un domaine que je ne connaissais pas encore. La société A2E, 90 employés environ, situé à Héricourt (70) fournit depuis 1992 des services de conception, d'assemblage et de test de systèmes électroniques. Cette PME conçoit de nombreuses cartes électroniques dans des milieux très vastes, et pour différentes applications chaque année.

J'ai travaillé avec une équipe d'ingénieur développement de carte électronique. J'ai pu me rendre compte que leur travail était fastidieux et long. L'objectif du stage est de réaliser un produit capable de capitaliser les projets déjà réalisés pour transmettre le savoir mais aussi pour réduire les délais de conception. La société aimerait développer une base de données regroupant les principales fonctions électroniques développées pour les projets précédent. L'idée est donc de découper un projet en fonctions susceptibles d'être réutilisées pour d'autres projets.

Le sujet de stage m'a fortement plu par son caractère libre et complet. En effet, j'ai participé à la réalisation du cahier des charges. Nous sommes donc partis d'une feuille blanche. La compréhension du métier, les auditions, l'analyse, l'organisation et la création sont des aspects très importants et intéressants pour moi, plus que le développement en lui-même. Ce stage est parfait pour mes objectifs.

A la demande du client, nous répondrons à la problématique : Comment transmettre le savoir afin de réduire les temps de développement de carte électronique ? Nous nous limiterons à la création d'une application répondant à cette problématique et correspondant aux attentes du client, toutes autres solutions ne sont pas envisagées dans cette étude.

L'architecture du rapport commencera par une présentation de la société A2e, nous détaillerons ensuite la phase d'analyse nécessaire à l'élaboration du cahier des charges puis nous présenterons le produit fini, nous établirons également un bilan technique et humain d'un point de vue personnel.

2. Présentation de la société

Depuis 1992 A2E (Applications Electroniques Européennes) propose la conception, l'assemblage et le test de systèmes électroniques. L'entreprise agit dans différents secteurs d'activité, tels que des appareils médicaux, des systèmes industriels ou bien des tests et mesures. L'entreprise a mis en place un concept de solution globale agile, afin de former des équipes plus polyvalentes.

Actuellement le directeur général de l'entreprise est Manuel Da Silva, depuis le départ du précédent directeur général Alain Gavois en 2015. La société est alors devenue membre du groupe R2V Corporate, de nos jours Synov, dont le président directeur général Nordine Mazari présidait déjà les entreprises JFI et CARV en France et TUNELEC en Tunisie. Ainsi, elle porte le statut de Société par Actions Simplifiées (SAS), une forme juridique flexible dans son fonctionnement.

Le chiffre d'affaire d'A2E est d'environ 15 M€ par an.

2.1. Structure de l'entreprise

A2E désormais dirigé par M. Manuel DA SILVA (voir figure ci-dessous) appartenant maintenant au groupe R2V composé de quatre sociétés dont trois en France : A2E, JFI et CERV ; et une située en Tunisie : TUNELEC.

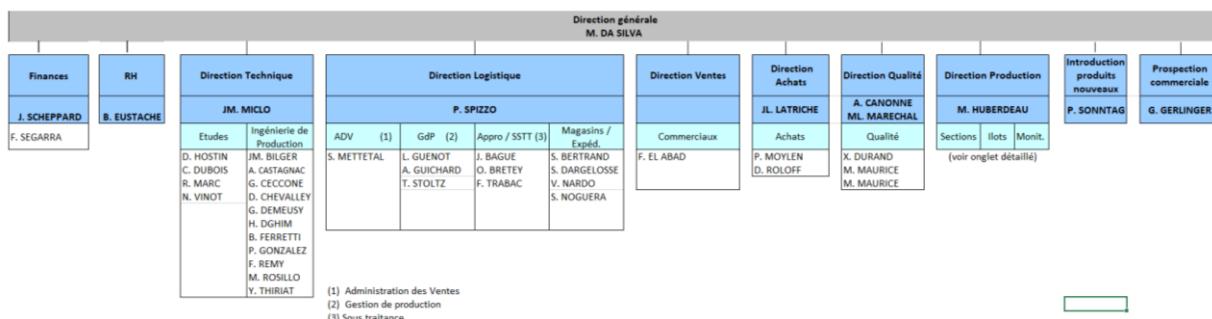


Figure 1 : Hiérarchie globale de l'entreprise

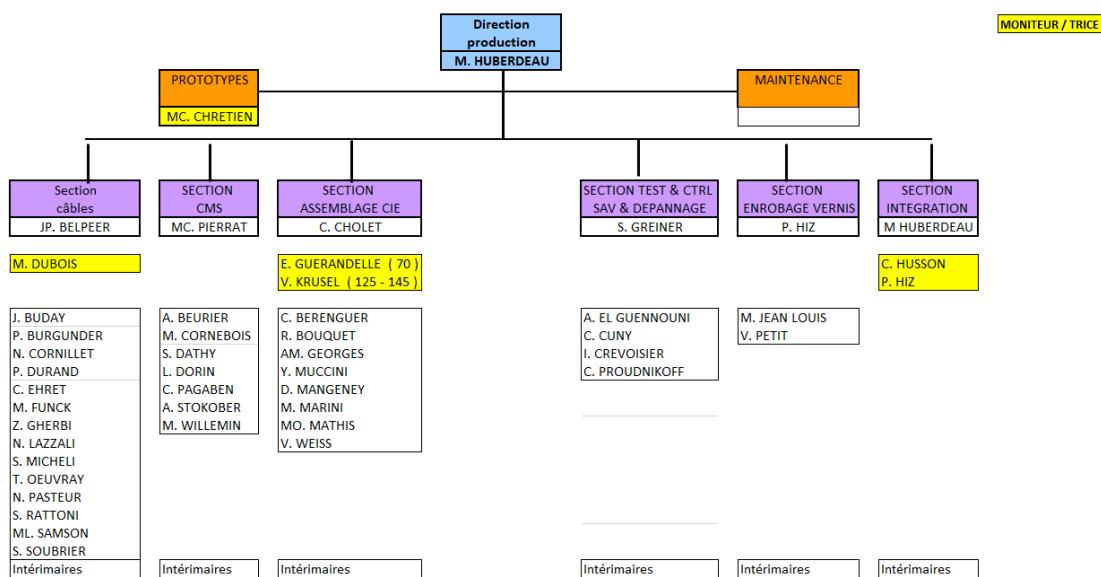


Figure 2 : Organigramme de la production

L'entreprise se divise principalement en deux grandes parties : une partie bureautique, avec les services financiers et logistiques (achats, ventes, ressources humaines, logistique) et les services techniques (méthode et production, études, qualité) et une partie atelier qui comporte un secteur dédié aux cartes, un aux câbles et un dernier à l'intégration.

Le service méthode et production est chargé d'étudier et réaliser l'industrialisation du produit. Il s'agit d'une logistique de pré-industrialisation (planification d'achats, identification des outillages nécessaires,) et la gestion de la production ensuite, notamment par la mise au point d'outillages spécifiques si nécessaire ou l'écriture de modes opératoires de test.

Le service étude s'occupe de la conception électronique hardware et software de produit à partir d'un cahier des charges fourni par le client. C'est dans ce service que mon stage se situe.

2.2. L'atelier de production

L'atelier est l'endroit où ont lieu toutes les fabrications commandées. L'atelier est organisé en trois parties distinctes :

2.2.1. Cartes électroniques

- CMS (Composants Montés en Surface)

Les CMS sont des composants montés en surface, c'est à dire des composants qui sont posés sur un circuit imprimé (PCB) et soudés. Les CMS stockés dans une étagère appelée Kardex (armoire de stockage rotative) sont ensuite, selon leur format, soit mis sur des cassettes, dans des tubes ou sur des plateaux et chargés dans une machine qui vient poser les composants sur le circuit imprimé. Ils sont ensuite soudés et le circuit imprimé est contrôlé.

- Traversants

Les traversants sont des composants mis à la main sur la carte électronique ; ces composants traversent, comme leur nom l'indique, le circuit imprimé et sont soudés soit manuellement, soit en passant dans une machine appelée « Vague ». Ces composants sont apportés dans l'atelier par les personnes du magasin et placés, soudés et contrôlés.

- Test

La zone de test est l'endroit où la carte électronique est contrôlée pour vérifier si elle est complète. Le contrôle est visuel sur des petites commandes ou des petites cartes, il est effectué par une machine pour les grandes quantités et les cartes compliquées. Si la carte est complète, elle est testée sur le produit final pour vérifier si le produit sera fonctionnel.

- Vernissage/Enrobage

Certaines cartes sont vernies. Le vernis assure une isolation ainsi qu'une protection efficace contre l'encrassement et l'humidité. Il est appliqué de manière automatique ou manuelle.

Certains produits fabriqués sont également enrobés avec de la résine. Cette opération est effectuée lorsque le produit final est utilisé dans des environnements difficiles ou pour des raisons de confidentialité ou pour l'aspect final du produit.

2.2.1. Câbles

Les câbles sont débités (coupés) et dénudés avec une machine. Après avoir été dénudé et coupé, le câble est repris sur une presse de sertissage dont le but est de mettre un contact au bout du fil destinés à être inséré dans un connecteur. Ensuite, le câble est préparé (montage du câble) manuellement puis contrôlé à 100% pour être emballé et envoyé au client.

2.2.2. Intégration

La zone d'intégration est l'endroit où sont réalisés des produits finis destinés à être vendus directement. L'intégration consiste à assembler des cartes, câbles, pièces mécaniques et plasturgies afin de restituer au client un produit directement utilisable.

3. Le cahier des charges

Avec l'équipe de développement, nous avons établi, ensemble, un cahier des charges détaillé. Un audit de fonctionnement a été pratiqué afin de comprendre les méthodes de fonctionnement dans le but de mieux cerner le problème et les attentes.

3.1. Mise en contexte du stage

La société A2E conçoit de nombreuses cartes électroniques dans des milieux très vastes, et pour différentes applications chaque année. Afin de capitaliser les projets déjà réalisés, la société aimerait développer une base de données regroupant les principales fonctions électroniques développées pour les projets précédant.

Cette base de données permettra de réutiliser des fonctions déjà validées afin de réduire les coûts et le temps de développement pour l'élaboration des devis.

La société A2E n'a que très peu de connaissances en base de données. J'interviens donc dans le but de réaliser cette banque de données avec l'interface homme machine permettant l'utilisation de celle-ci.

3.2. Analyse du besoin

Une analyse du besoin a été réalisée afin de déterminer les enjeux et les objectifs du projet. L'expression du besoin est également nécessaire pour connaître si le projet est viable ou non.

3.2.1. Objectifs du projet et enjeux

L'enjeux consiste à développer une application (client lourd) permettant de stocker les différents projets de développement de carte électronique. Aujourd'hui, l'équipe de développement ne possède aucune fonction de recherche sur des cartes déjà réalisées antérieurement. De plus, ils ne sont pas parfaitement au courant de ce qu'il a déjà été réalisé par les uns et les autres. Le but est donc de centraliser ces données afin d'avoir une fonction de recherche pour réutiliser des fonctions dites « génériques » (l'idée est de décomposer un projet entièrement en fonctions plus simples, potentiellement réutilisables, et de les stocker).

L'objectif du projet à court terme est de conserver le savoir de l'entreprise concernant les projets de développement de cartes électroniques. Les objectifs à long terme sont de lui permettre de lui faire gagner du temps de développement en réutilisant des fonctions dites génériques ainsi que gagner du temps sur la validation des fonctions. En effet, il est inutile de tester une fonction déjà validée par le bureau d'étude.

3.2.2. Expression du besoin et validation

Le Figure 3 : Diagramme Bête à corne met en exergue l'expression du besoin du produit demandé.

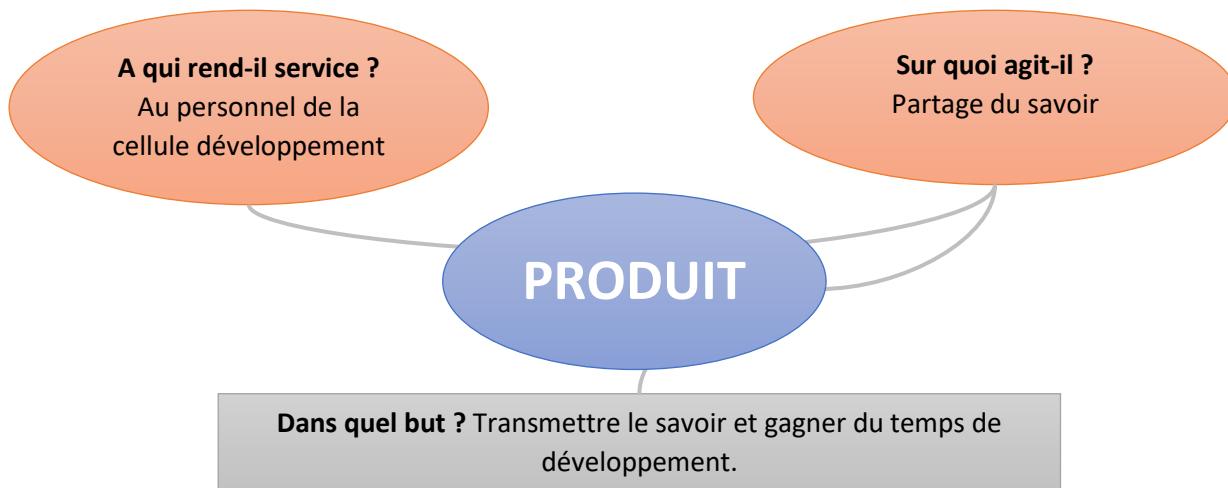


Figure 3 : Diagramme Bête à corne

3.2.2.1. Pour quelles raisons le besoin existe-t-il ?

Le besoin existe car il est important de transmettre le savoir faire de l'entreprise plus efficacement pour les nouveaux employés. A long terme avec une grande base de connaissance, l'équipe pourra gagner du temps de développement ainsi que sur les PV¹ de tests des fonctions déjà validée.

3.2.2.2. Qu'est-ce qui pourrait faire disparaître le besoin ? Ou le faire évoluer ?

La disparition de la cellule de développement peut nuire à ce besoin. Le développement de projets qui n'ont strictement rien en commun, peut également impacter de manière négative celui-ci.

La demande croissante, la redondance des fonctionnalités que demandent les clients et la fortification de l'équipe de développement sont des raisons qui peuvent faire évoluer le besoin.

3.2.2.3. Quelle est la probabilité ?

Les raisons citées ci-dessus qui nuisent au besoin ont une très faible probabilité d'apparaître, contrairement à celles qui peuvent le faire évoluer.

En conclusion, en fonction des objectifs cités et des évolutions possibles, nous pouvons affirmer que le projet est viable et aura un réel impact sur la conception de futurs cartes électroniques.

3.3. Organisation mise en place

Il a été convenu de mettre en place la méthode agile et plus particulièrement le cadre de travail SCRUM.

3.3.1. Méthodologie AGILE (SCRUM)

« L'approche Agile propose au contraire de réduire considérablement voire complètement cet effet tunnel en donnant davantage de visibilité, en impliquant le client du début à la fin du projet et en adoptant un processus itératif et incrémental. Elle considère que le besoin ne peut être figé et propose au contraire de s'adapter aux changements de ce dernier. Mais pas sans un minimum de règles. »²

Nous allons utiliser le cadre de travail SCRUM qui repose sur les différents principes AGILE. Le principe de fonctionnement est donné dans la figure ci-dessous.

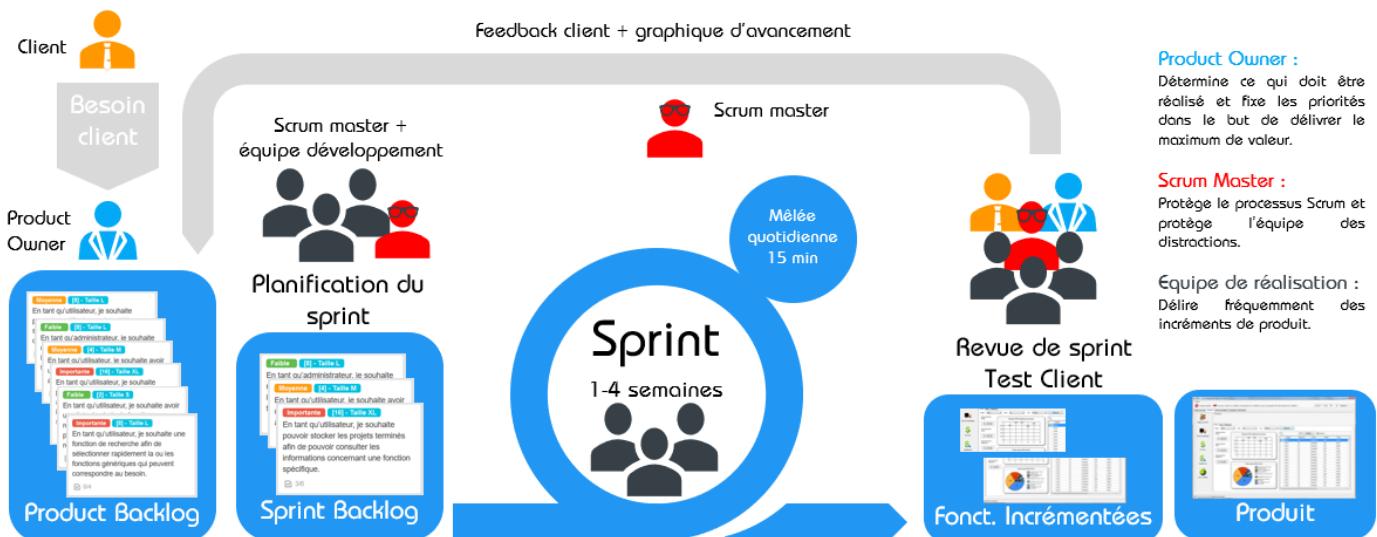


Figure 4 : Principe de fonctionnement SCRUM

Dans un premier temps, nous allons définir les récits utilisateurs concernant le projet. Nous avons établi l'importance de chacun ainsi que sa taille relative. Nous nous basons sur des revues de Sprint d'une

¹ Procès-Verbal de test

² (Lothon, 2015)

semaine afin d'avoir un retour très rapide du client. A la suite des tests clients, le backlog (liste de fonctionnalités ou de tâches) du produit sera alimenté par de nouveaux récits utilisateurs.

3.3.2. Équipe Agile

Notre cas est un peu spécial, dans la mesure où le product Owner est aussi notre client, à savoir monsieur Nicolas Vinot :



Figure 5 : Équipe Scrum

3.3.3. Planification

Un diagramme de Gantt a été réalisé. Les livrables attendus pour le client sont la documentation et l'application finalisée. En voici une succincte présentation de la planification :

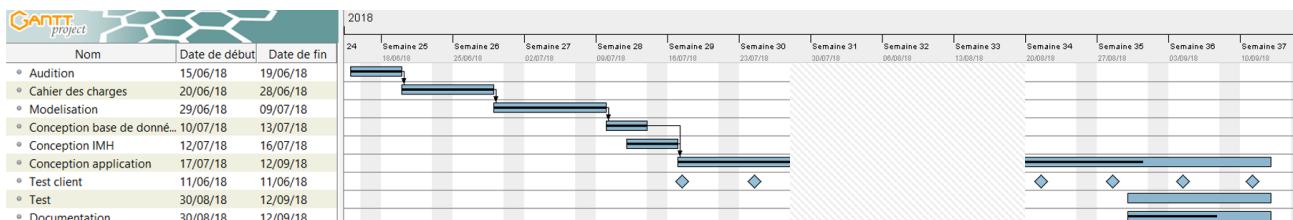


Figure 6 : Planification sous GanttProject

Tous les mardis, un test client a été effectué lors de réunion, j'ai présenté l'avancement de l'application et le backlog du produit a été alimenté au fur et à mesure. Pour la partie conception de l'application, la méthodologie agile a grandement aidé au dimensionnement temporel de l'application. Nous avons utilisé Trello³ pour la réalisation des récits utilisateurs et gérer le backlog du produit.

Figure 7 : Organisation Agile sous Trello

³ <https://trello.com/b/Ye8OgEnL/a2e-bdd-devis>

3.4. Analyse fonctionnelle

Une fois le besoin du projet validé, nous pouvons réaliser le cahier des charges en analysant le plus en détails possible les fonctions de services du produit répondant à notre problématique.

3.4.1. Recherche des fonctions de service

Nous avons recherché les fonctions du logiciel attendu. Le diagramme "pieuvre" met en évidence les relations entre les différents éléments du milieu environnant et le produit.

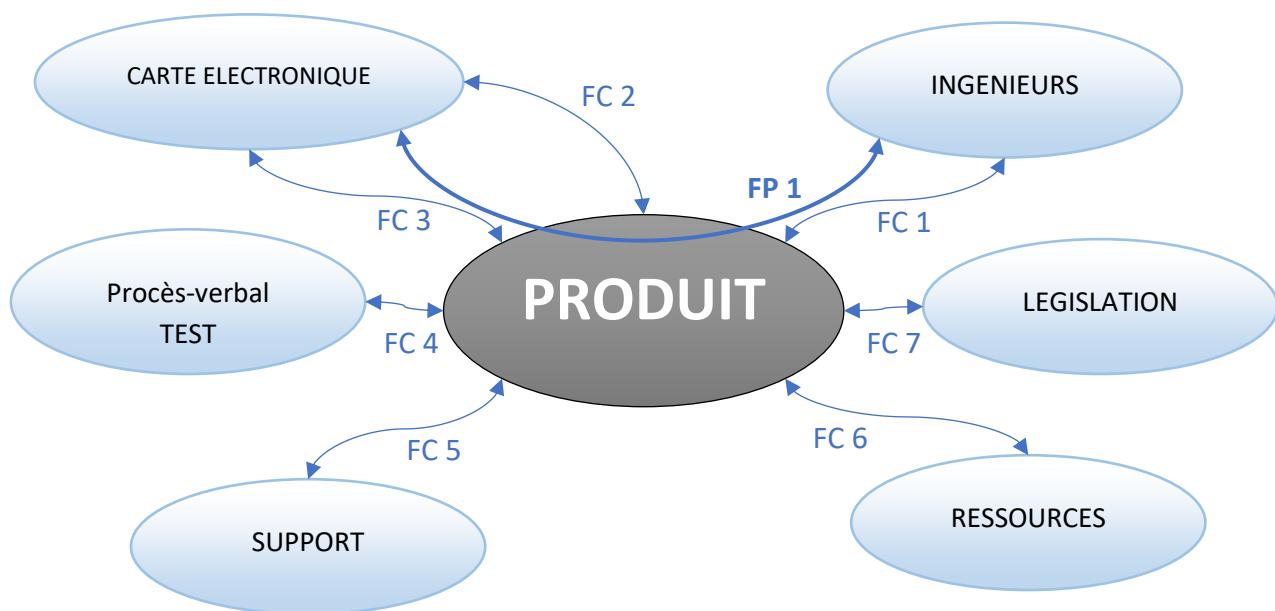


Figure 8 : Diagramme pieuvre

Voici la liste de la fonction principale et des fonctions contraintes en référence au diagramme pieuvre ci-dessus :

Identifiant	Désignation
FP 1	Le produit doit permettre aux ingénieurs de faciliter leur travail de développement de carte électronique
FC 1	Le produit doit permettre aux ingénieurs d'administrer la base de données
FC 2	Le produit doit permettre de retracer entièrement un projet d'une carte électronique
FC 3	Le produit doit être capable de stocker des fonctionnalités génériques d'une carte électronique
FC 4	Le produit doit permettre l'accès direct au dossier des procès-verbaux de test.
FC 5	Le produit doit fonctionner sous Windows Seven et Windows 10.
FC 6	Le produit doit être maintenable par le personnel de l'entreprise.
FC 7	Le produit doit respecter les lois de la CNIL et de la charte informatique interne à l'entreprise

Tableau 1 : Tableau des fonctions de service

3.4.2. Caractérisation des fonctions de service

A présent, il est nécessaire de définir les fonctions de service (Tableau 1 : Tableau des fonctions de services) avec un plus haut niveau de précision. Le tableau ci-dessous répertorie ces fonctions ainsi que le/les critères caractérisant celles-ci en appliquant également un niveau et flexibilité pour les différents critères.

Id	Critères	Niveau	Flexibilité
FP 1	Création nouveau projet	10 000 projets	Au moins
	Gestion des fonctions génériques	CRUD ⁴ 1 000 fonctions	Au moins
	Gestion liste des composants physiques	CRUD 1 000 000 composants	Au moins
	Fonctionnalité de recherche	5 critères (date, projet, fonctions, composants, puissance entrée/sorties)	Au moins ces 5 critères Choix plusieurs composants
FC 1	Utilisation de leur savoir-faire (PhpMyAdmin) et former à l'autonomie	Formation administration BDD PhpMyAdmin	
FC 2	Indiquer si la fonction a été validée ou non Pointage vers lien SNV projet complet	CRUD validation ou non CRUD lien	
FC 3	Décomposition carte en fonctions génériques	CRUD 1 000 fonctions par carte (projet)	Au moins
	Stockage image de la fonction	Stocker l'adresse de l'image dans la BDD et stocker l'image sur le serveur	
	Stockage de toutes les informations utiles	1 description fonction dédiée au projet 1 ligne puissance IO (4 données) 10 IO physique + type 10 critères génériques	= = Au plus Au plus
FC 4	Pointage vers lien dossier SNV PV test	1 lien par fonction	Lien ou cellule <i>null</i>
FC 5	Développement avec Framework .NET	Langage C#	.NET 3.5
FC 6	Environnement de développement (IDE) Utilisation technologie SNV (Gestion des versions)	Visual studio 2008, PhpMyAdmin	
FC 7	Lois informatiques Inaccessible depuis l'extérieur	CNIL, Charte informatique ent.	CNIL

Tableau 2 : Fonctions de service, critères, niveau et flexibilité

⁴ CRUD – Create Read Update Delete

4. Conception du produit demandé

Dans cette partie, nous détaillerons la modélisation de notre produit sous forme du langage de modélisation unifié (UML). Par mesure de clarté, le code ne sera pas détaillé et sera fourni uniquement en annexe. Bien que la méthode agile prévoie de réaliser la conception dirigée par les tests, nous avons adopté une approche plus classique car les tests seront rédigés en aval du projet. A l'heure de la rédaction de ce document, les tests ont été planifiés mais n'ont pas encore été effectués dues aux contraintes temporelles de mon stage (décalage des trois semaines de vacances imposées, voir Figure 6 : Planification sous GanttProject: Planification sous GanttProject).

4.1. Base de données

Au préalable, afin de réaliser le modèle conceptuel de données, il est important de réaliser un audit (voir Annexe 1 – Audit de fonctionnement) auprès de l'équipe de développement afin de comprendre leur manière de fonctionner ainsi que de connaître avec autant de précision possible les fonctionnalités qu'ils attendent.

4.1.1. Informations à exploiter

Cette base de données doit contenir un minimum d'informations pour être exploitée de façon rapide et optimale par toute l'équipe du bureau d'études. Le but n'est pas de contenir le projet complet mais les informations nécessaires pour savoir si la fonction peut correspondre à nos besoins. Si le chef de projet a besoin de plus d'informations, il peut aller consulter les projets sur les différents points de sauvegarde. Cette liste doit pouvoir être dynamique et modifiée à terme en cas d'oubli de champs.

Voici une première liste des principaux éléments qu'elle doit contenir :

Numéro	Intitulé	Exemples	Format	Type
1	Nom	Nicolas	VARCHAR(31)	Primitif
2	Prénom	Vinot	VARCHAR(31)	Primitif
3	Mail	nicolas.vinot@a2e.fr	VARCHAR(63)	Primitif
4	Nom Projet	Projet lampe auto	CHAR(10)	Primitif
5	Date Projet	18/06/2018	DATE	Primitif
6	Cout projet	4 500 €	FLOAT	Primitif
7	Ancienneté projet	1 an		Calculé
8	Lien SVN projet	Svn://A2E-FICHIERS/E*****/RFID	VARCHAR(255)	Primitif
9	Rubrique fonction	Alimentation	VARCHAR(63)	Primitif
10	Fonction générique	BOOST	VARCHAR(63)	Primitif
11	Schéma fonction	../Aima-lgestionalimentation.png	BLOB,VARCHAR(255)	Primitif
12	Description fonction	Gestion de l'alimentation pour projet AIMA ...	VARCHAR(1047)	Primitif
13	Coût fonction	120 €	FLOAT	Primitif
14	Lien SVN dossier PV test	Svn://A2E-FICHIERS/E*****/RFID/...	VARCHAR(255)	Primitif
15	Validation	Oui, non	BOOLEAN	Primitif
16	Courant entrée	1,2 A	FLOAT	Primitif
17	Tension entrée	12 V	FLOAT	Primitif
18	Puissance entrée	14 Watts		Calculé
19	Courant sortie	0,5A	FLOAT	Primitif

Numéro	Intitulé	Exemples	Format	Type
20	Tension sortie	24 V	FLOAT	Primitif
21	Puissance sortie	12 Watts		Calculé
22	IO physique v	BUS CAN, VBAT_MEAS, MESURE_VIN	VARCHAR(255)	Primitif
23	Quantité IO	3	INT	Primitif
24	Type IO physique	Analogique, numérique, pwm	VARCHAR(31)	Primitif

Tableau 3 : Champs base de données

Une liste déroulant contenant l'intitulé des fonctions sera définie par les utilisateurs, (par exemple : charge, régulation courant, Buck, Boost, Infrarouge, Montage relais...). Cette liste sera agrandie par l'aval de l'ensemble de l'équipe. A chaque fin de projet, cette base de données doit être mise à jour par le chef du projet.

4.1.2. Modèle conceptuel de données (MCD)

A présent, nous allons définir la structure de notre base de données en réalisant un modèle conceptuel de données. La solution définitive est un mixte entre les solutions 1 et 3 définies en Annexe 2 – Modèles conceptuels de données, qui est parfaitement en accord avec le Tableau 3 : Champs base de données.

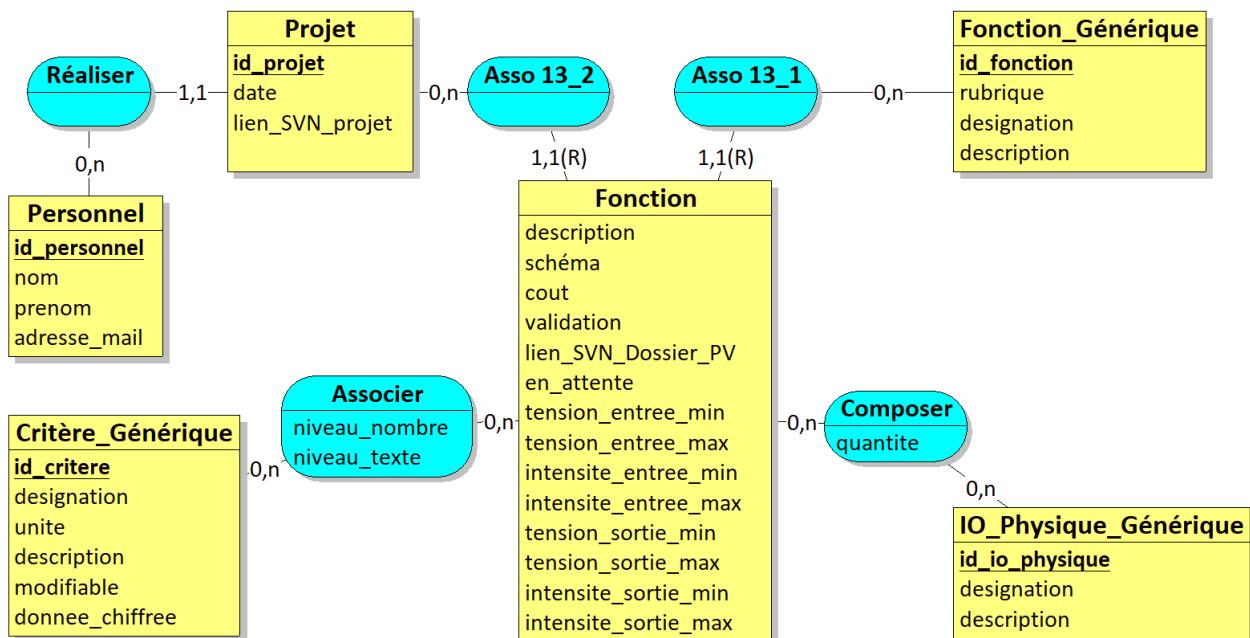


Figure 9 : MCD produit final

D'autres colonnes ont été ajoutées comme niveau_nombre et niveau_texte dans Associer car en fonction du critère nous renseignerons soit une valeur numérique (ex : fréquence 23 Hz) soit une valeur alphanumérique (ex : driver : drivers 2.0 pour Linux Debian). Le champ modifiable dans Critère_Générique autorise ou non la modification du critère générique.

Les colonnes tension_entree_min, ... ont été ajoutées car elles sont inhérentes à chaque fonction que l'utilisateur renseignera dans le système. Cette conception facilitera grandement le travail de recherche plus tard. Ces plages correspondent aux plages pour lesquelles la fonction a été validée par les PV de tests, si la fonction peut fonctionner sur une plage plus large (mais pas contrôlée par les PV) alors l'utilisateur pourra la renseigner par un critère générique en doublon (ex : crit_tension_entree_min = -12V, tension_entree_min = -6V). À terme, cette information sera utile pour la fonction de recherche car on calculera un taux de correspondance (voir Algorithme de la fonction de recherche).

4.1.3. Création de la BDD

La base de données a été initialement administrée sur phpMyAdmin mais, ayant trouvé l'outil très limité (impossibilité, par exemple, de créer une fonction SQL), j'ai travaillé avec MySQL Workbench 6.3 CE. Le système de gestion de base de données est MariaDB, édité sous licence GPL. Il s'agit d'un fork communautaire de MySQL.

Les différentes requêtes sont données en Annexe 3 – Requêtes de la création des tables de la base de données.

4.2. Conception UML

4.2.1. Architecture mise en place (N-tiers)

En ce qui concerne la modélisation, l'architecture du produit a du rapidement être mis en place. Après plusieurs recherches, j'ai choisi un modèle en plusieurs couches ; l'architecture N-Tiers.

« Cette séparation par couches de responsabilités sert à découpler au maximum une couche de l'autre afin d'éviter l'impact d'évolutions futures de l'application. Par exemple : si l'on est amené à devoir changer de base de données relationnelle, seule la couche d'accès aux données sera impactée, la couche de service et la couche de présentation ne seront pas concernées car elles auront été découpées des autres. »⁵

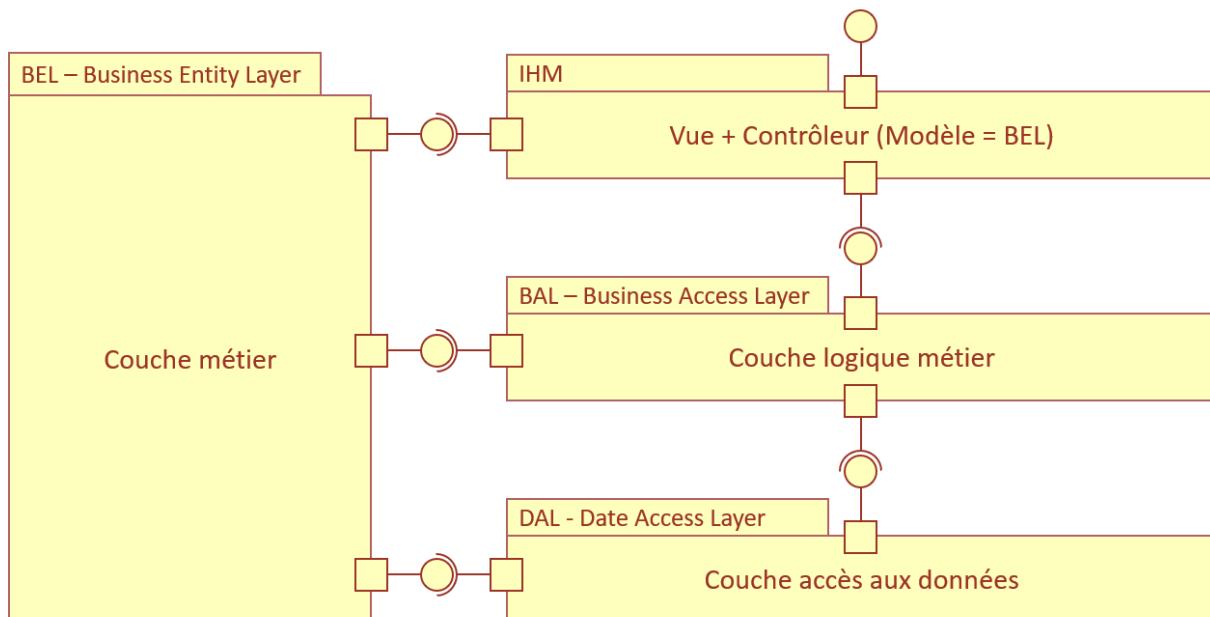


Figure 10 : Diagramme de composant N-Tiers

Comme le montre la figure ci-dessus, on constate que la couche métier est transverse aux trois autres couches. La couche d'accès aux données communiquera avec la base de données MariaDB. La couche supplémentaire à une architecture 3-tiers est la couche logique métier. Cette couche est très utile, par exemple, pour contrôler et vérifier avant l'insertion des données dans la base.

4.2.2. Cas d'utilisation

Notre système comportera un seul utilisateur disposant des droits de lecture et d'écriture dans la base de données.

⁵ (Farinone, 2011)

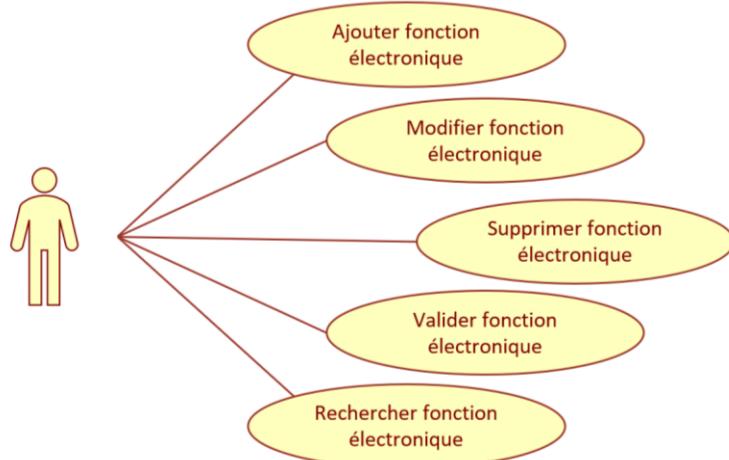


Figure 11 : Diagramme cas d'utilisation

Voici le descriptif des fonctionnalités du diagramme cas d'utilisation :

Nom du cas d'utilisation	Ajouter fonction électronique.
Rôle du CU	Ce cas d'utilisation permet à l'utilisateur de créer de nouvelles fonctions électroniques avec les données telles que le nom du projet, le nom de la fonction... Ce cas devra permettre également l'insertion d'une image correspondante au schéma électronique de la fonction concernée.
Début du CU	Ce cas d'utilisation se produit lorsque l'utilisateur veut ajouter une fonction.
Fin du CU	A la fin de l'ajout lorsque celui-ci s'est effectué avec succès (aucun code erreur provenant du moteur de bdd).
Précondition	Aucune
Postcondition	Sauvegarde des données dans la bdd puis passe en mode modification de la fonction avec la clé primaire de la table Fonction non modifiable. Lors de l'ajout, la fonction n'est pas validée par défaut. Elle passe dans une file d'attente (liste) de validation.
Données en entrées	La liste des fonctions génériques, la liste des projets, la liste des IO physiques génériques et la liste des critères génériques.
Données en sorties	Mise à jour de la table Fonction dans la bdd et la sauvegarde d'une image sur le serveur.
Exception	Code erreur moteur de bdd.

Tableau 4 : Cas d'utilisation - Ajouter fonction électronique

Nom du cas d'utilisation	Modifier fonction électronique.
Rôle du CU	Ce cas d'utilisation permet à l'utilisateur de modifier une fonction électronique avec les données. Seuls le projet et la fonction générique ne peuvent pas être modifiés.
Début du CU	Ce cas d'utilisation se produit lorsque l'utilisateur veut modifier une fonction ou après l'ajout d'une fonction.
Fin du CU	Lorsque l'utilisateur quitte le mode de modification.
Précondition	La sélection d'une fonction
Postcondition	Aucune
Données en entrées	La liste des fonctions génériques, la liste des projets, la liste des IO physiques génériques et la liste des critères génériques.

Données en sorties	Mise à jour de la table Fonction dans la bdd et la sauvegarde d'une image sur le serveur.
Exception	Code erreur moteur de bdd.

Tableau 5 : Cas d'utilisation - Modifier fonction électronique

Nom du cas d'utilisation	Supprimer fonction électronique.
Rôle du CU	Ce cas d'utilisation permet à l'utilisateur de supprimer une fonction électronique.
Début du CU	Ce cas d'utilisation se produit lorsque l'utilisateur veut supprimer une fonction.
Fin du CU	Lorsque la suppression a été effectuée.
Précondition	La sélection d'une fonction
Postcondition	Le mode de la fenêtre passe en mode ajouter.
Données en entrées	Une liste de fonction
Données en sorties	Mise à jour de la table Fonction dans la bdd
Exception	Code erreur moteur de bdd.

Tableau 6 : Cas d'utilisation - Supprimer fonction électronique

Nom du cas d'utilisation	Supprimer fonction électronique
Rôle du CU	Ce cas d'utilisation permet à l'utilisateur de valider une fonction électronique.
Début du CU	Ce cas d'utilisation se produit lorsque l'utilisateur veut modifier une fonction puis veut valider la fonction.
Fin du CU	Lorsque la validation a été effectuée.
Précondition	La sélection d'une fonction puis sélection du mode de modification
Postcondition	Aucune
Données en entrées	Une liste des fonctions en liste d'attente.
Données en sorties	Mise à jour de la table Fonction dans la bdd
Exception	Code erreur moteur de bdd.

Tableau 7 : Cas d'utilisation - Valider fonction électronique

Nom du cas d'utilisation	Rechercher une fonction électronique
Rôle du CU	Ce cas d'utilisation permet à l'utilisateur de rechercher une fonction électronique. Un tri sera effectué en fonction du pourcentage de correspondance avec les critères voulus. Toutes les fonctions seront affichées et triées.
Début du CU	Ce cas d'utilisation se produit lorsque l'utilisateur veut rechercher une fonction électronique.
Fin du CU	A la fin de la recherche.
Précondition	Une fonction générique sélectionnée, tout autres critères seront exceptionnels.
Postcondition	Aucune
Données en entrées	Une liste de fonctions génériques.
Données en sorties	Une liste de fonctions électroniques triées par son taux de correspondance.
Exception	Code erreur moteur de bdd.

Tableau 8 : Cas d'utilisation - Rechercher fonction électronique

4.2.3. Diagramme de séquence

Les diagrammes de séquences sont les interactions entre les acteurs et le système à étudier (SAE). Les interactions sont représentées dans un ordre chronologique (du haut vers le bas).

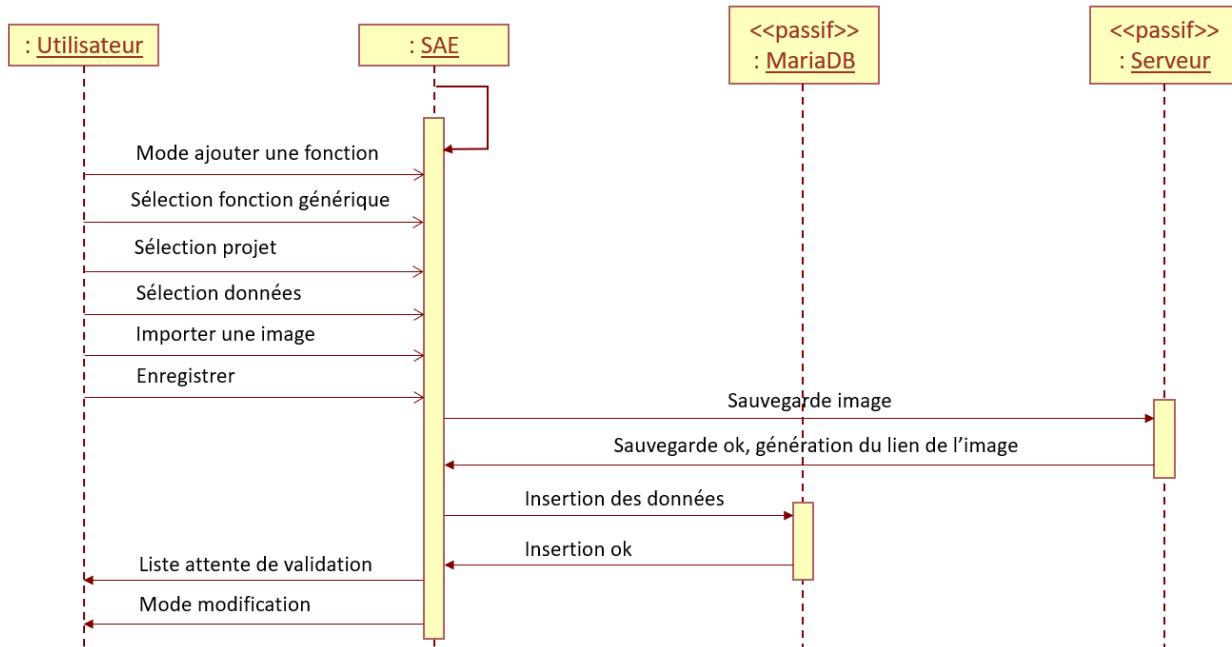


Figure 12 : Diagramme de séquence – Ajouter une fonction électronique

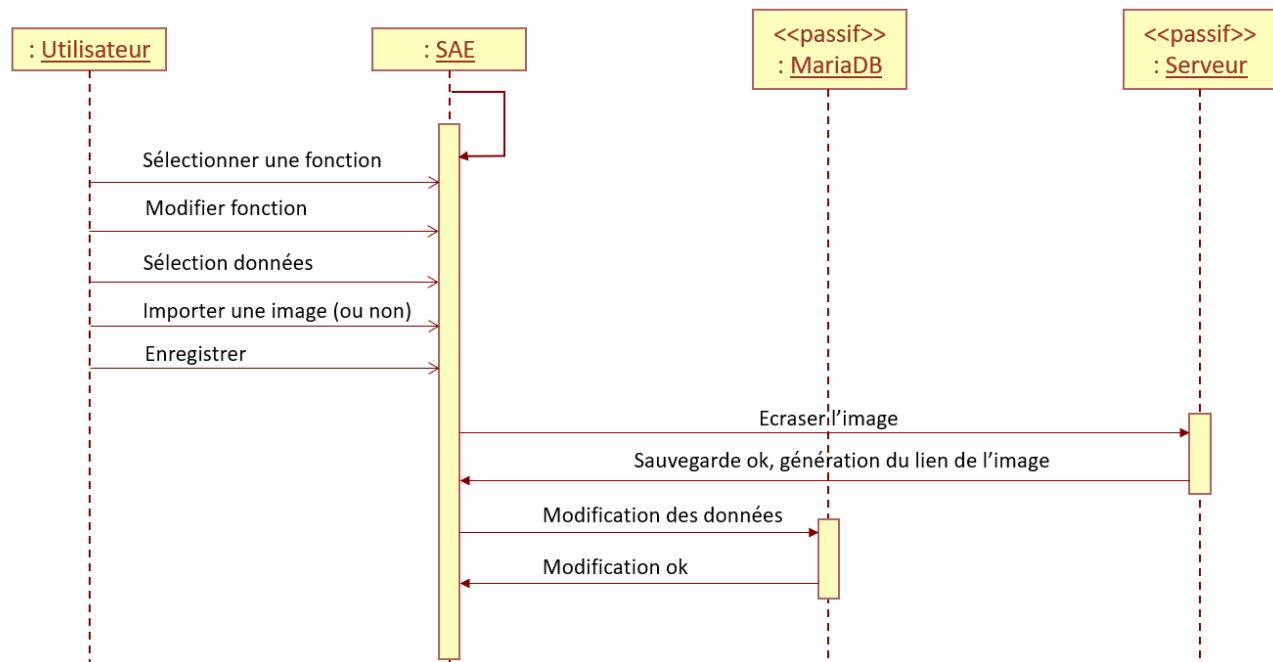


Figure 13 : Diagramme de séquence – Modifier une fonction électronique

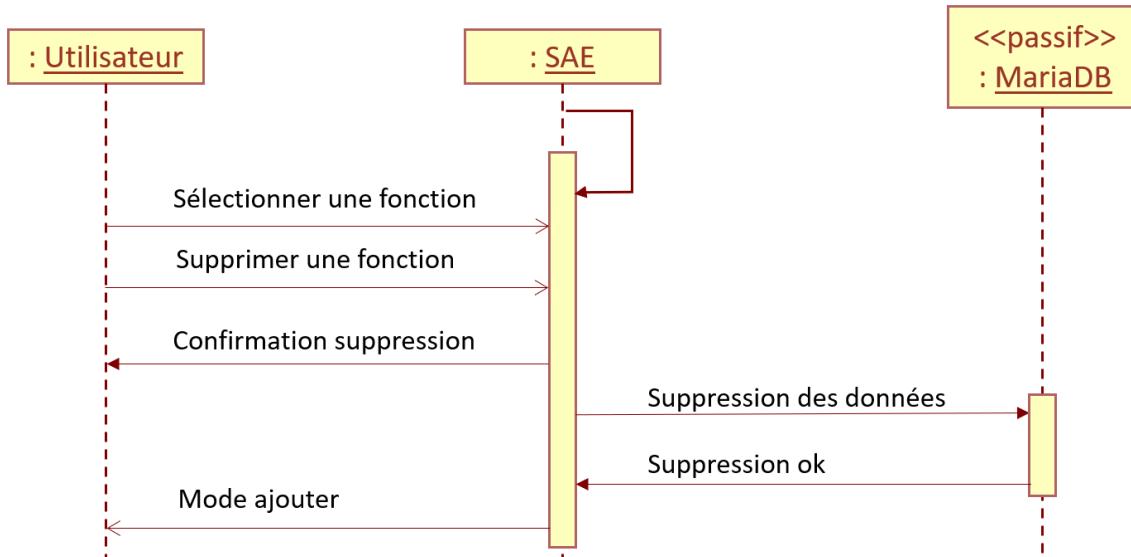


Figure 14 : Diagramme de séquence – Supprimer une fonction électronique

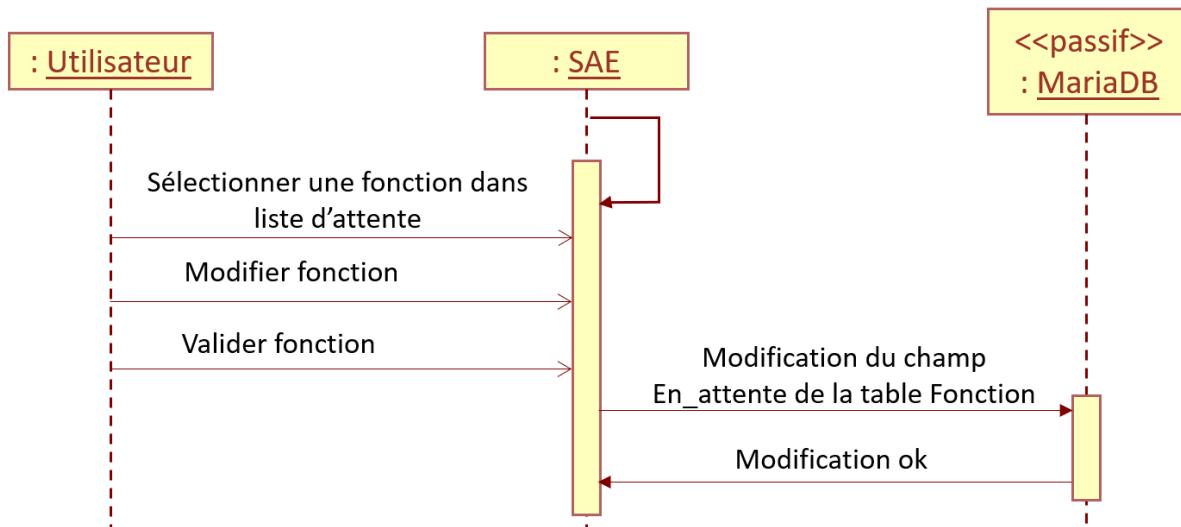


Figure 15 : Diagramme de séquence – Valider une fonction électronique

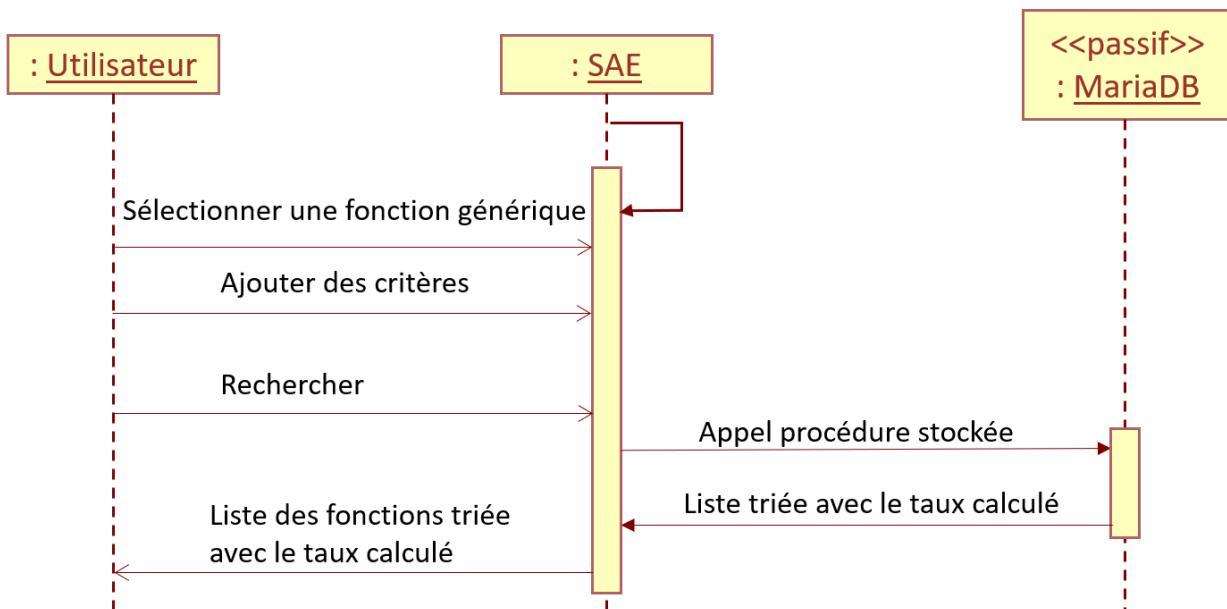


Figure 16 : Diagramme de séquence – Rechercher une fonction électronique

4.2.4. Diagramme de classe métier (couche BEL)

Nous avons établi le diagramme de classe de notre produit qui présente les différentes classes du système de la couche BEL ainsi que les relations entre celles-ci. Une classe décrit les responsabilités, le comportement et le type d'un ensemble d'objets. Les éléments de cet ensemble sont les instances de la classe. Dans ce rapport, nous ne détaillerons pas les classes des autres couches.

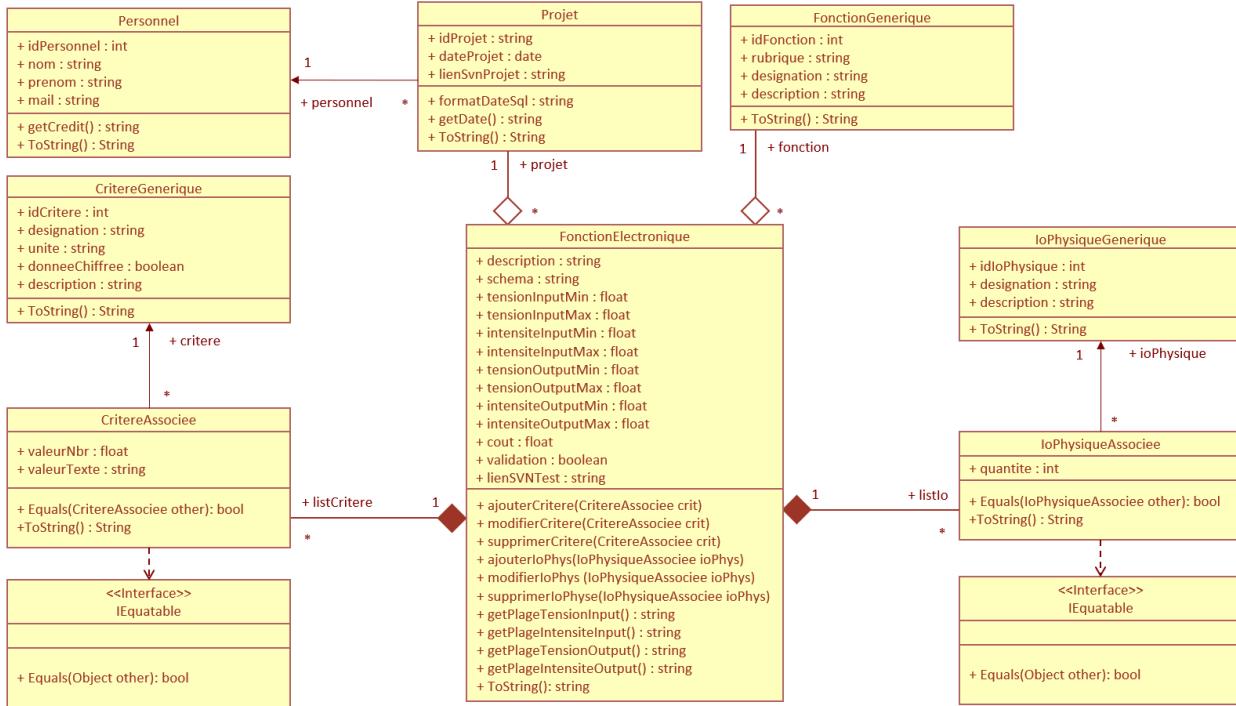


Figure 17 : Diagramme de classe métier

4.2.5. Description de la dynamique de l'interface : Diagramme d'état transitions

Les diagrammes d'états-transitions UML décrivent le comportement de la dynamique de notre interface graphique (voir Modélisation de l'Interface Homme Machine (IHM)). Ils présentent les enchainements entre les différentes fenêtres que nous utiliserons ensuite. Seul le diagramme d'état transition de la fenêtre ajouter/modifier une fonction est représenté dans ce rapport. Il s'agit de la fonction principale (voir 5.2 pour plus de compréhension).

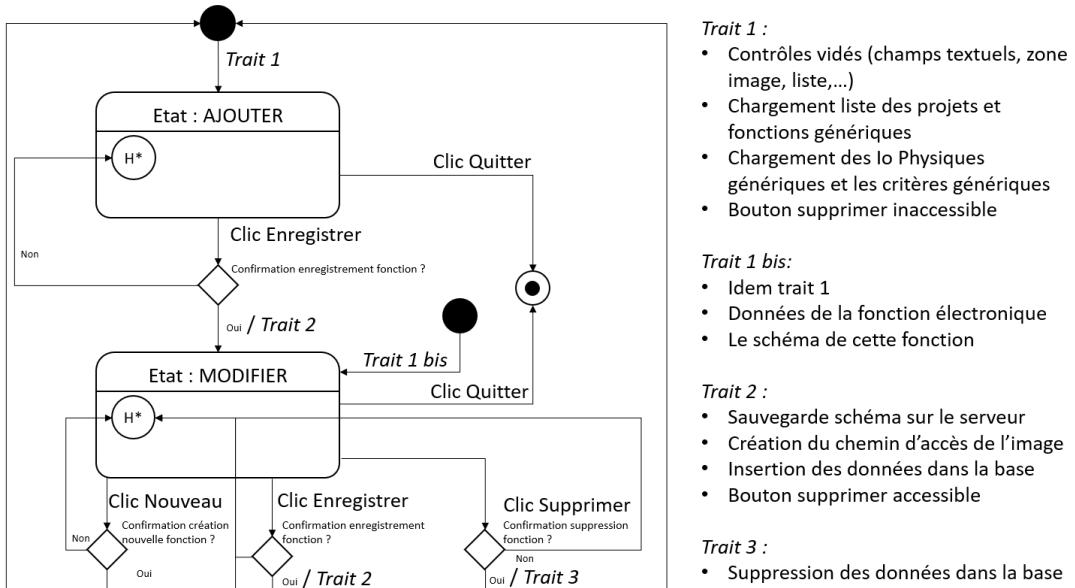


Tableau 9 : Diagramme état transition de la fenêtre ajouter/modifier une fonction

4.3. Modélisation de l'Interface Homme Machine (IHM)

Une première partie du travail est d'esquiver rapidement une maquette de l'interface homme machine. Ce travail est nécessaire pour valider rapidement avec le client si l'orientation choisie est la bonne. Nous allons présenter de manière succincte ce travail. Au terme de cette étude, l'application a rapidement été ébauchée avec l'outil Visual studio (C#) en respectant les maquettes ci-dessous. L'application terminée est présentée dans la partie 5.2 et en Annexe 8 – Enchainement des fenêtres de l'IHM.

Hypothèse : les maquettes qui vous sont présentées ici nous servirons lors de la programmation informatique du logiciel. Cependant, il se peut qu'elles évoluent suivant les éventuelles problèmes ou solutions de réalisation que nous serions amenés à rencontrer à travers les échanges avec le client.

On se base sur une application fenêtre sous le système d'exploitation Windows.

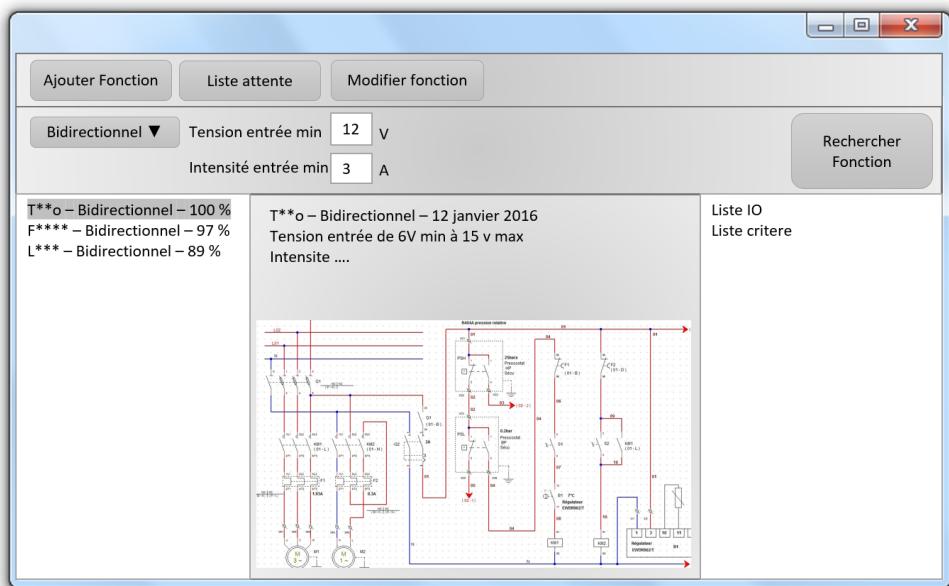


Figure 18 : IHM fenêtre principale

La figure ci-dessus montre notre fenêtre d'accueil. C'est la première chose que l'utilisateur verra. On peut la décomposer en trois parties, le bandeau du haut permet l'ajout, la gestion de la file d'attente (détalée en 5.2) et la modification de la fonction. Le bandeau du milieu correspondra au filtre de recherche. Enfin le bandeau bas présente la fonction sélectionnée dans la liste à droite.

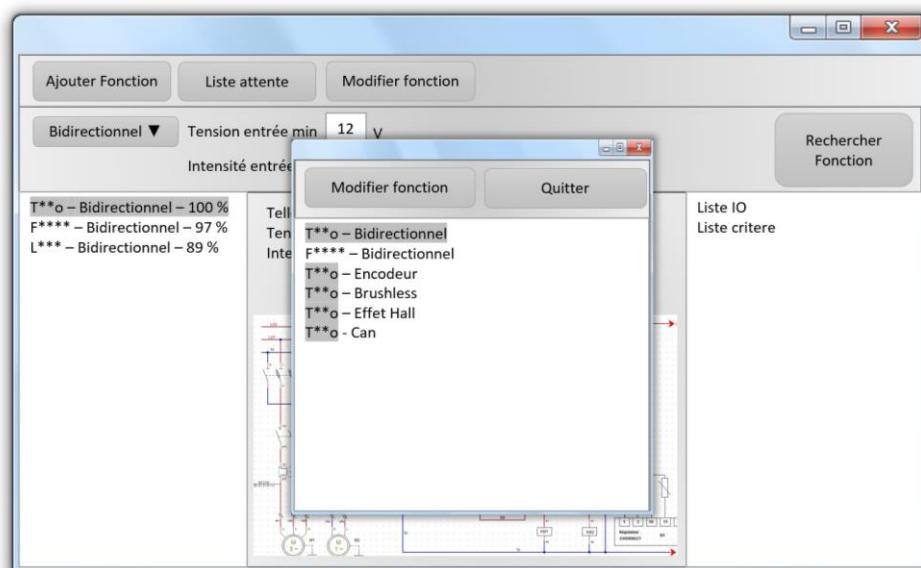


Figure 19 : IHM liste d'attente

La fenêtre liste d'attente apparaît lors du clic sur le bouton « Liste attente ». On sélectionne ensuite une fonction pour la modifier (et donc potentiellement pour la valider).

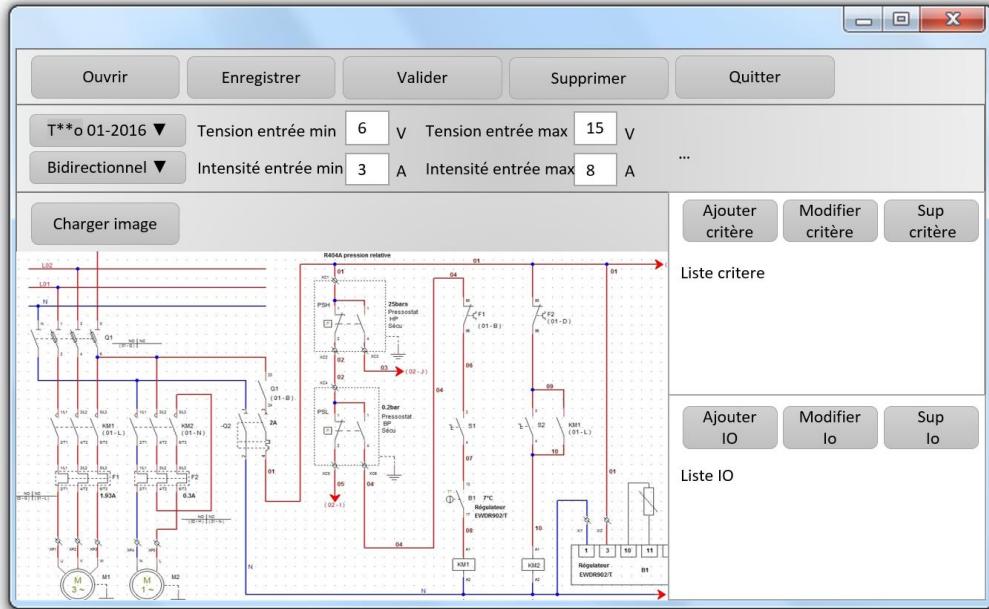


Figure 20 : IHM ajouter/modifier/supprimer et valider une fonction

Cette fenêtre sera unique pour l'ajout, la modification, la suppression et la validation d'une fonction électronique. On jouera sur la gestion des événements pour bloquer les contrôles.

Cette ébauche a rapidement été validé par le client. J'ai ensuite travaillé avec Visual Studio pour épurer encore plus l'interface homme machine. Visual Studio est un outil très pratique car il s'agit d'un système glisser/déposer pour la conception d'interface. Grâce à la méthode agile, l'IHM a rapidement évolué lors des revues tous les mardis.

5. Résultats obtenus

Dans cette partie, nous présenterons, les résultats finaux concernant l'application, les tests et la documentation. Pour rappel, nous avons réalisé l'application sous l'IDE Visual Studio en C# (à la demande du client, voir 3.4.2) avec le Framework .NET.

5.1. Algorithme de la fonction de recherche

Un point plutôt compliqué, et selon moi le plus intéressant, était de réaliser la fonction de recherche en se focalisant sur l'idée première du projet qui était le gain de temps sur la conception. Imaginons que nous recherchons une fonction avec des critères bien précis, d'un point de vue purement SQL avec la clause WHERE, on risque malheureusement d'éliminer beaucoup trop de solutions. L'idée est donc de calculer un taux de correspondance à l'aide d'une fonction SQL plutôt que d'utiliser la clause WHERE. Nous aurons donc toutes les fonctions correspondantes classées dans l'ordre du taux calculé.

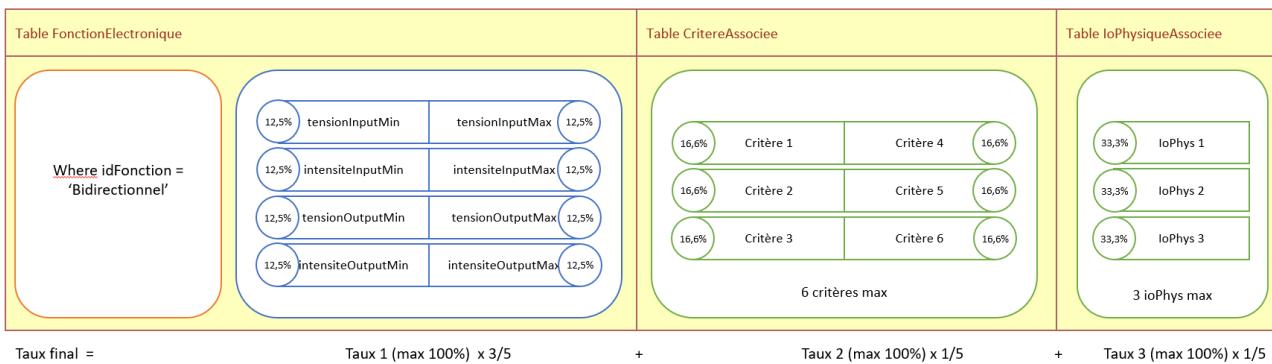


Figure 21 : Algorithme de la fonction de recherche

La figure ci-dessus montre l'algorithme pour le calcul du taux de correspondance. Si les champs ne sont pas renseignés alors le taux est directement incrémenté de la valeur maximale. Le taux 1, étant le plus important, est le plus prépondérant avec une valeur de 3 pour 5. Une petite subtilité réside dans le fait que les valeurs dans la table FonctionElectronique sont des valeurs pour lesquelles la fonction a été contrôlée mais nous pouvons ajouter un critère dans la table CritereAssociee pour ajuster ces valeurs si la fonction est aussi susceptible de fonctionner pour une plage plus grande (ex : le taux peut alors être amélioré si la tensionInputMin ne correspond mais correspond au critère).

Le code de cet algorithme est proposé en Annexe 4 – Procédures stockées et fonctions pour le calcul du taux de correspondance. Attention celui-ci est partiellement terminé et sera amélioré à la fin du stage.

5.2. L'application

Une partie du code de programmation de l'application est proposée dans les annexes (Annexe 5 – Code C# de la couche métier BEL, Annexe 6 – Code C# de la couche DAL pour la classe FonctionElectronique, Annexe 7 – Code C# de la couche BAL pour la classe FonctionElectronique) en suivant les différentes couches de l'architecture N-tiers. Des modules ont été ajoutés comme la possibilité de manipuler l'image ou la possibilité de réaliser des captures d'écran depuis l'application. Ses améliorations ont été ajoutées au backlog du produit à la demande du client et ont été très appréciées.

D'après ces annexes, nous pouvons constater que toutes les méthodes CRUD ont été rédigées pour toutes les classes métiers, nous pouvons donc insérer, modifier, supprimer et lire les données de toutes les tables dans la base de données.

L'architecture globale du projet est présentée ci-dessous. Par analogie au langage Java, on remarque bien que les différentes couches correspondent à différents « package ». L'outil de CaptureEcran a été conçu dans un projet à part. Il peut fonctionner de manière indépendante comme une application Windows.

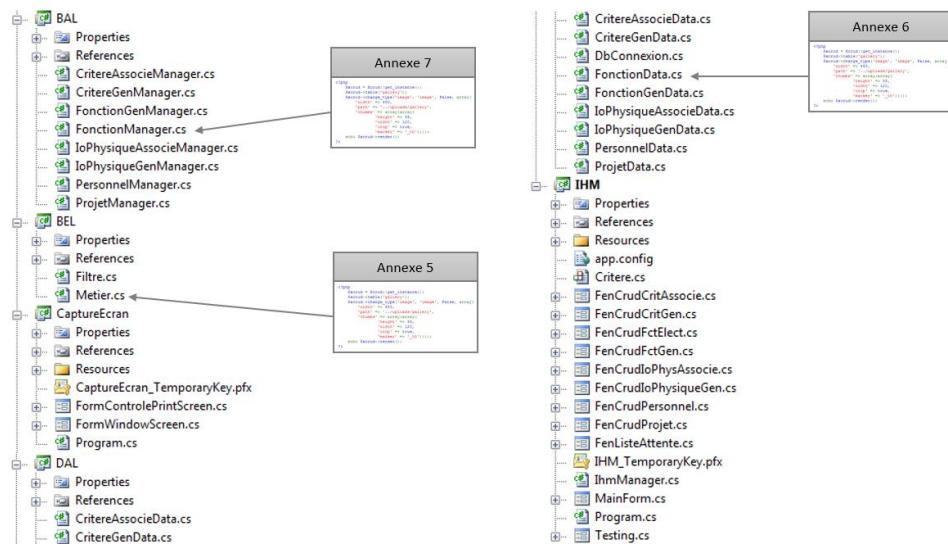


Figure 22 - Architecture général du projet

A présent, nous allons détailler l'interaction entre les différentes fenêtres graphiques de notre interface (voir Annexe 8 – Enchainement des fenêtres de l'IHM). La Figure 23 : Ecran d'accueil présente la première fenêtre que l'utilisateur verra au lancement de l'application.

Contrôles	Fonctionnalités
Bouton 1	Permet l'appel d'une fenêtre nécessaire à la création d'une fonction.
Bouton 2	Fait apparaître la liste des fonctions en attente de validation
Bouton 3	Permet l'appel de la même fenêtre que le bouton 1 (en mode modification) si une fonction est sélectionnée dans la zone liste des fonctions
Bouton 4	Déclenche l'algorithme de recherche et alimente la liste des fonctions
Bouton 5	Réinitialise (vide les zones de texte, ...) tous les filtres dans la zone de filtre de recherche.

Tableau 10 : Fonctionnalités de l'écran d'accueil

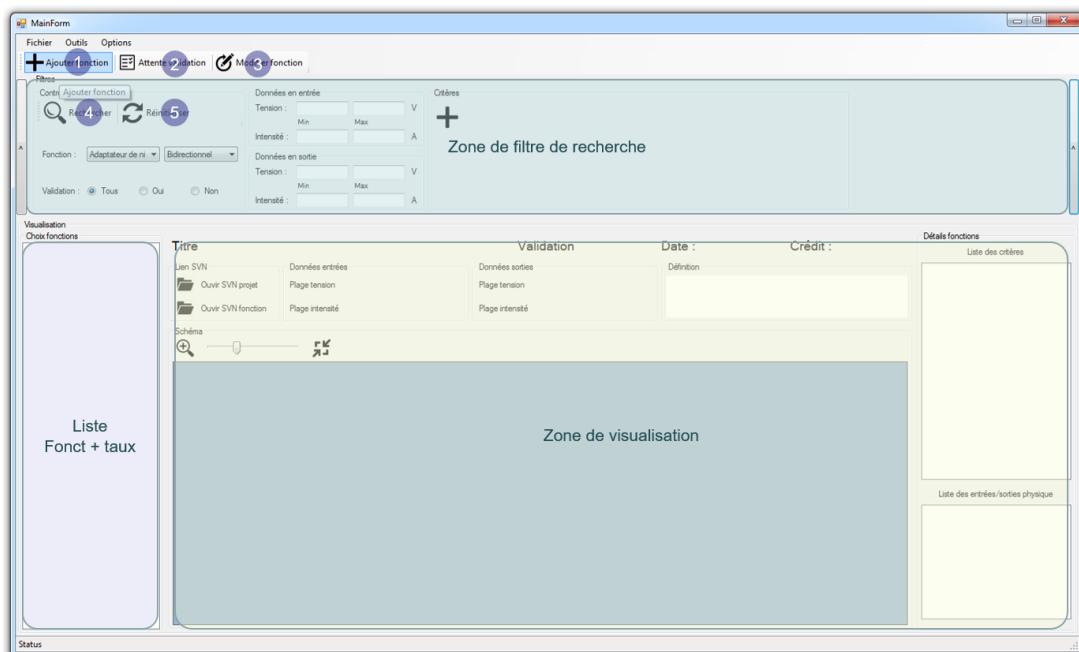


Figure 23 : Ecran d'accueil

Lors du clic du bouton 1 (voir Figure 23 : Ecran d'accueil), une fenêtre (voir figure ci-dessous) de création ou de modification d'une fonction apparaît. Nous pouvons modifier toutes les tables de la base de données depuis l'IHM à l'aide des boutons modifier, ajouter et supprimer (voir zones encadrées 1).

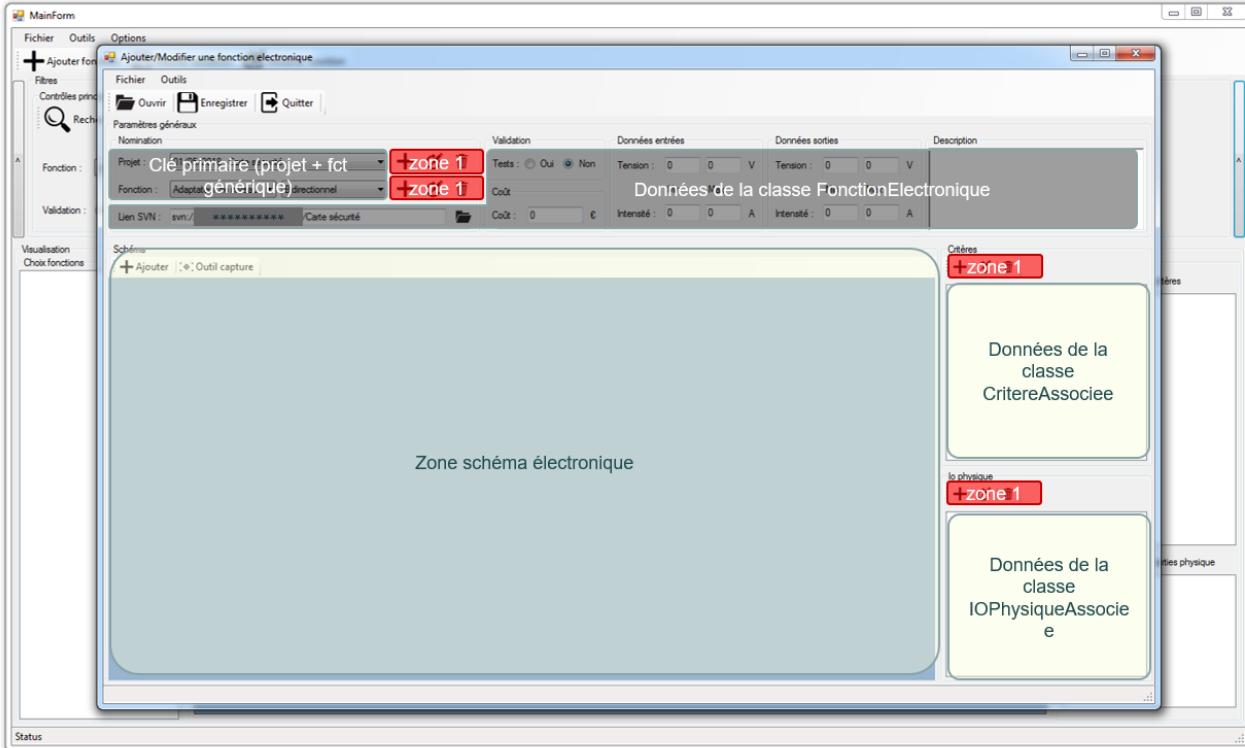


Figure 24 : Fenêtre ajouter ou modifier une fonction (mode ajouter)

Dans la Figure 25 : Modification d'un projet, nous présentons, en exemple, la fenêtre de modification d'un projet lors du déclenchement du bouton 1. Nous remarquons que nous pouvons également ajouter, modifier ou supprimer un responsable (zone encadrée 1).

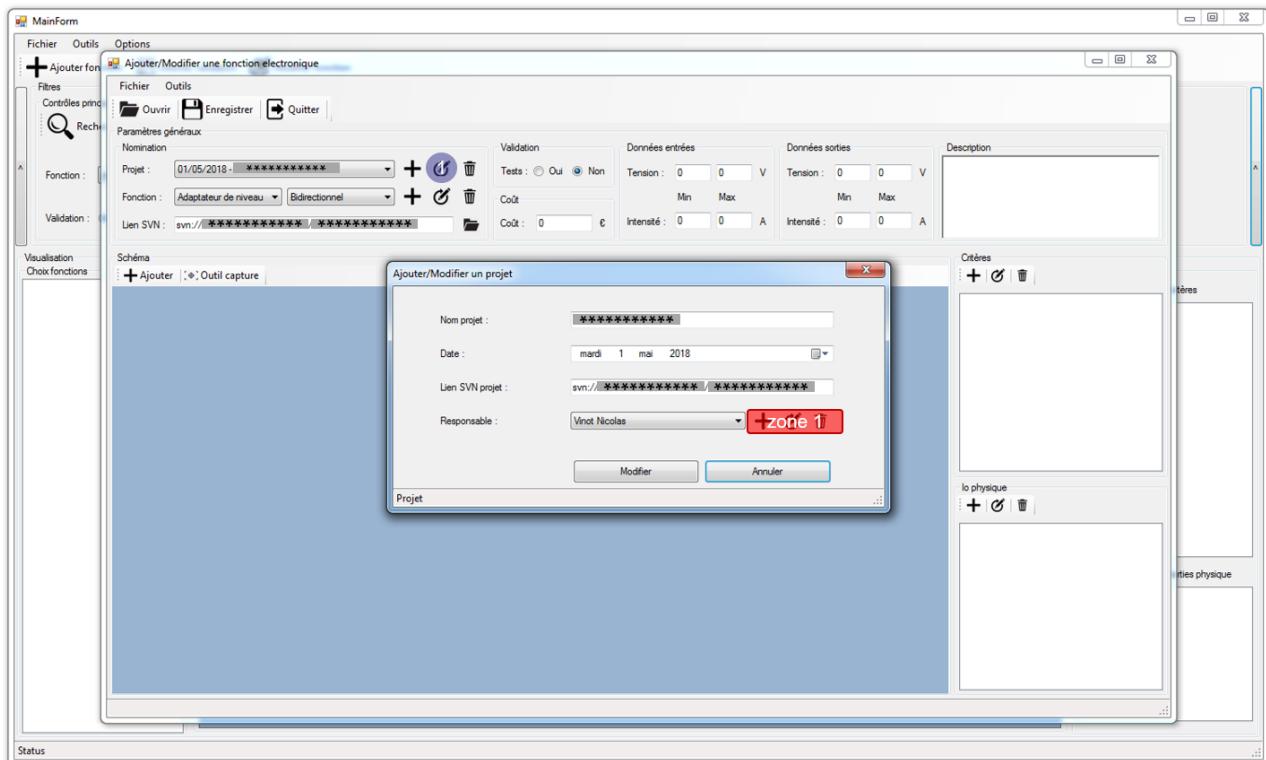


Figure 25 : Modification d'un projet

Un effort a été apporté à l'application par l'intégration et le développement d'un outil de capture d'image. Le bouton 1 de la figure ci-dessous fait appel à cette application.

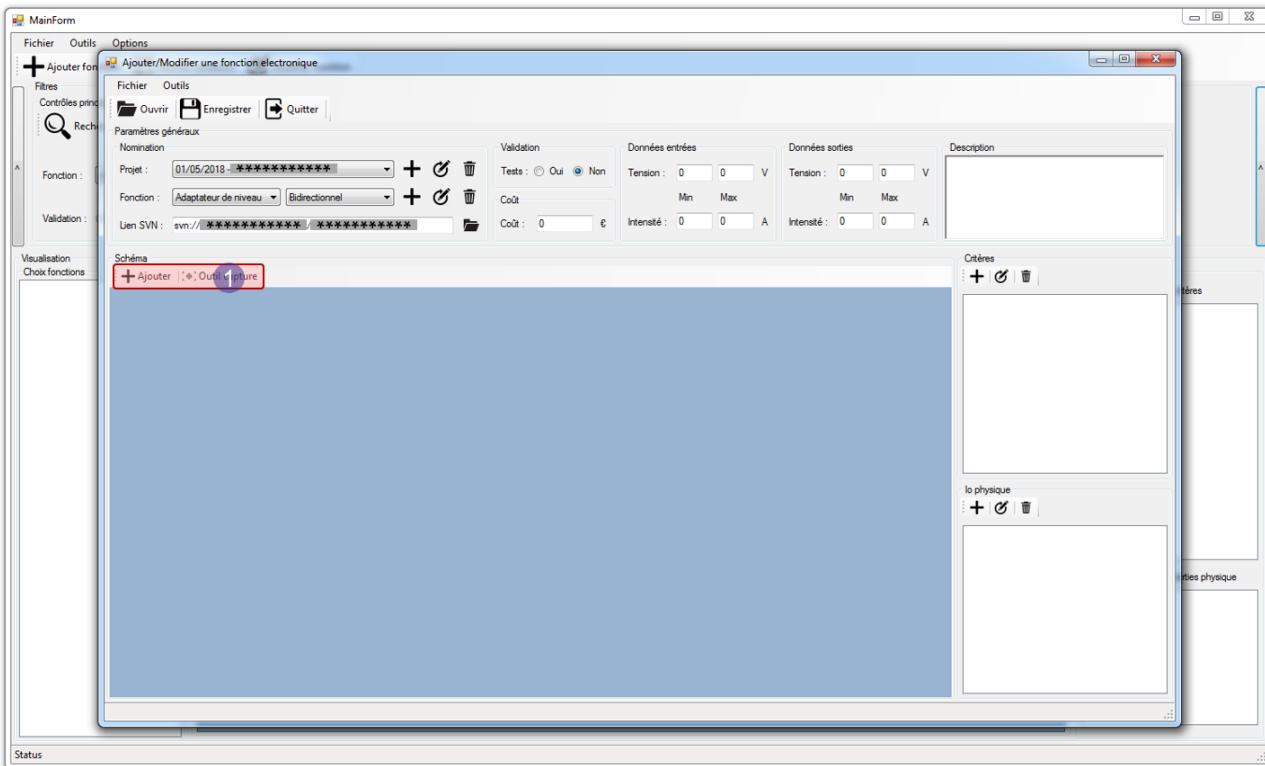


Figure 26 : Outil de capture d'écran

Lors de l'appel de cette fonction, l'application se met en arrière-plan et nous pouvons commencer la capture de l'écran de travail. La zone capturée est délimitée par le clic numéro 1 de la souris et par le clic numéro 2. L'image capturée se trouve à l'intérieur de l'encadré en pointillé.

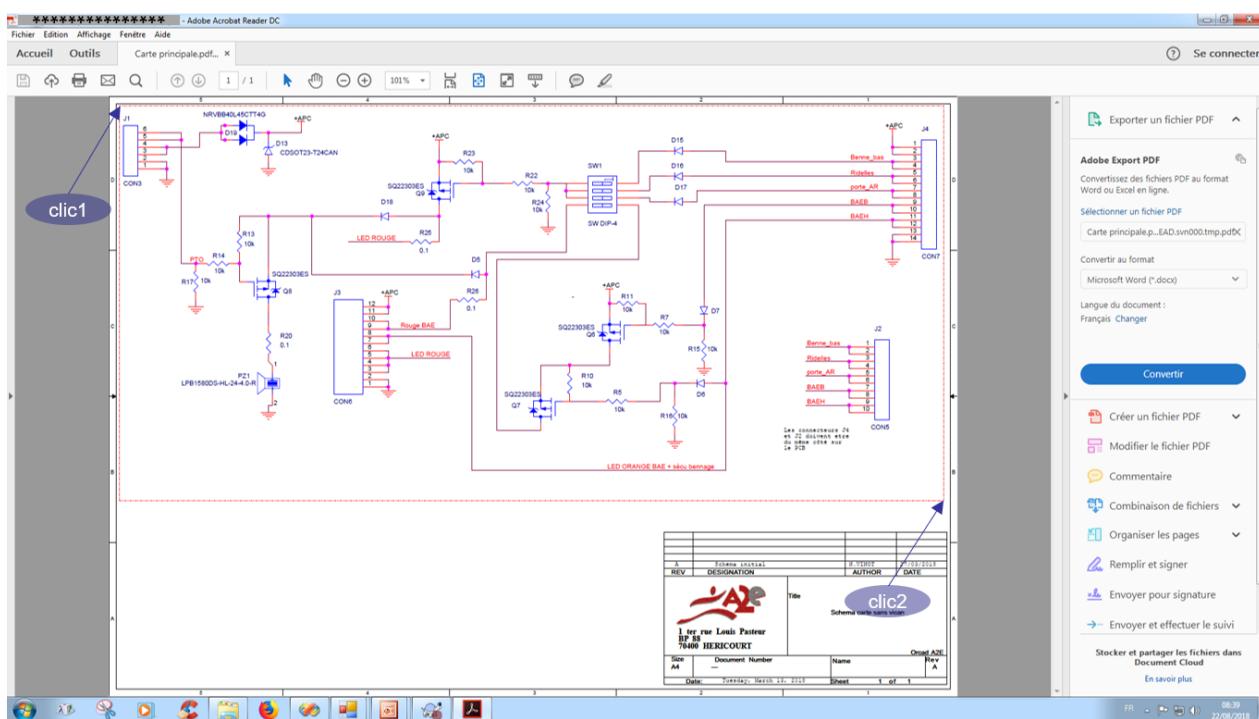


Figure 27 : Zone capturée

A la fin du deuxième clic, une fenêtre apparaît avec la zone capturée. Nous pouvons recommencer une capture avec le bouton 1 si celle-ci ne convient pas ou enregistre l'image à l'aide du bouton 2.

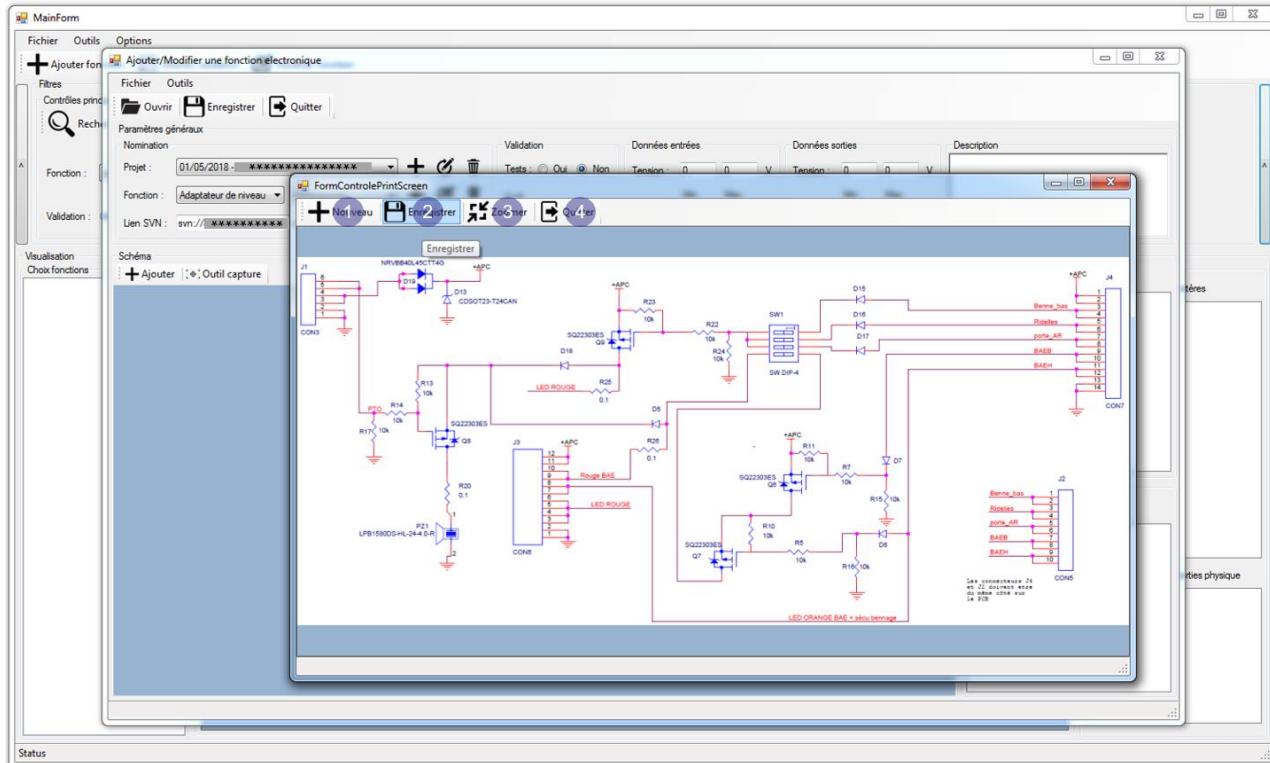


Figure 28 : Manipulation image capturée

Lors de l'enregistrement de l'image, la fenêtre se ferme et l'image est auto importée dans la zone schéma (voir Figure 24 : Fenêtre ajouter ou modifier une fonction (mode ajouter)). Nous pouvons déclencher l'insertion des données dans la base de données à l'aide du bouton 1 de la Figure 29 : Insertion des données.

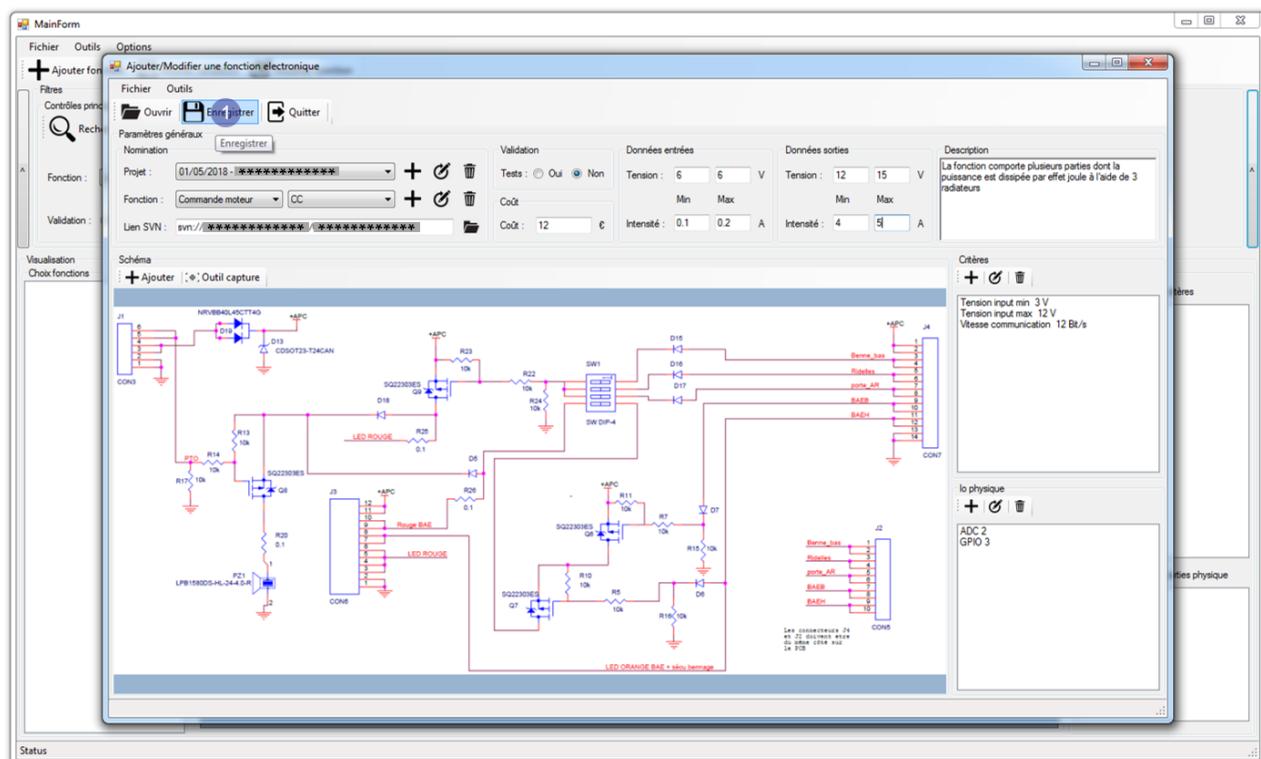


Figure 29 : Insertion des données

Lors de cet enregistrement, si nous étions dans le mode ajouter, alors la fenêtre passe en mode modification de la fonction (voir figure ci-dessous). Les contrôles de sélection de la fonction générique et du projet sont bloqués (car il s'agit de la clé primaire). Le bouton 1 de validation apparaît par défaut à la valeur en attente de validation (champs en_attente). Le menu déroulant (2) permet d'ajouter une nouvelle fonction (réinitialise tous les contrôles), d'ouvrir une fonction déjà créée ou de supprimer la fonction actuellement affichée.

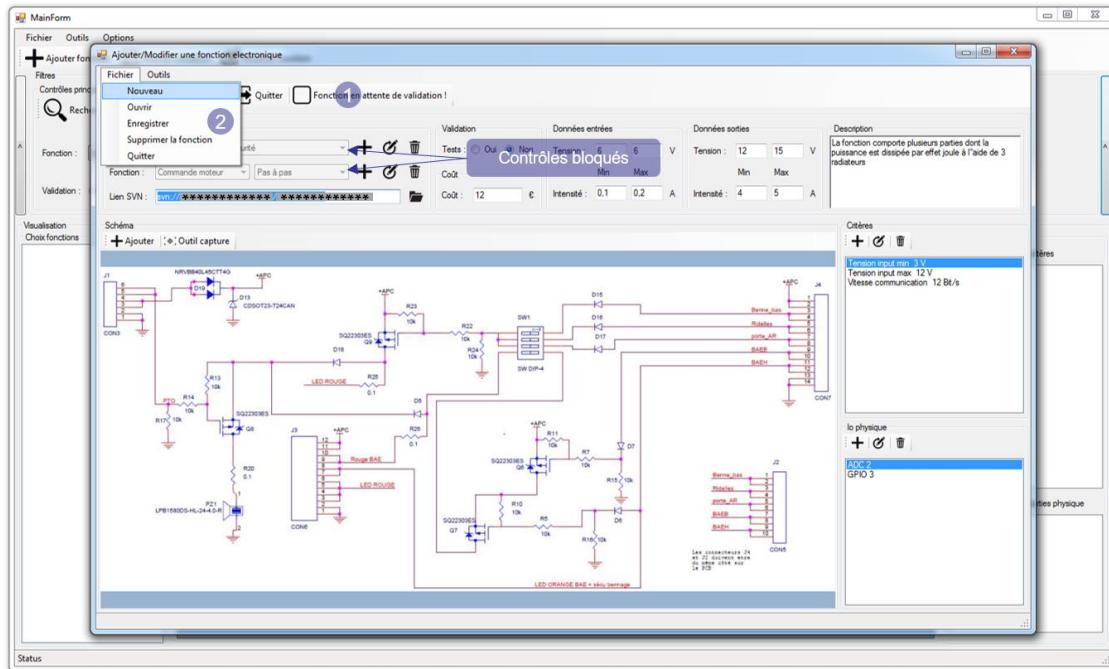


Figure 30 : Fenêtre ajouter ou modifier une fonction (mode modification)

A présent, nous allons rechercher une fonction à l'aide des filtres. Selon l'exemple ci-dessous, nous allons rechercher toutes les fonctions « Pas à pas » (liste déroulante 2). Les boutons 3 et 4 permettent l'ajout ou la suppression d'un critère (jusqu'à 6 critères affichés), le signe peut également être choisi. Le même principe sera adopté pour les Io physiques (jusqu'à 3). Après la déclenchement de la recherche (bouton 1), la liste des fonctions triées apparaît en zone 5.

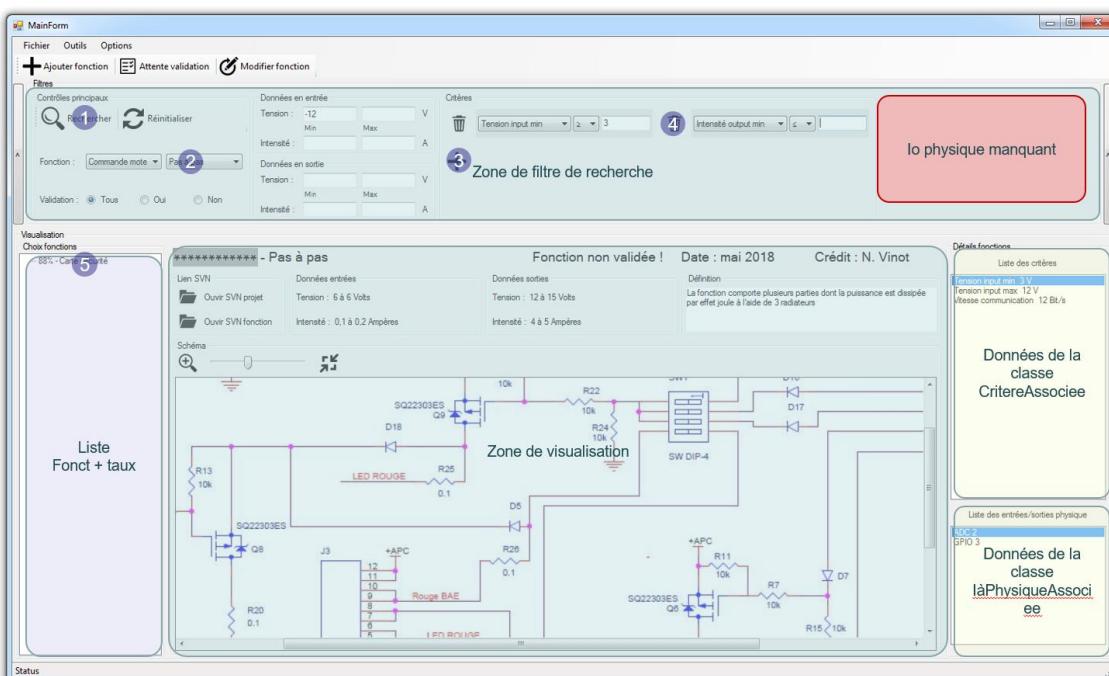


Figure 31 : Filtre de recherche

5.3. Les tests

Etant donné le planning proposé lors de la rédaction de ce rapport, je n'ai malheureusement pas encore eu l'opportunité de rédiger les tests. Ceux-ci seront effectués la première semaine de septembre. De plus, il me faut du temps pour l'apprentissage à la rédaction de test à l'aide du Framework .Net.

5.4. La documentation

La documentation du code est particulièrement utile lors des évolutions et de la maintenance du code. C# propose un mécanisme simple pour insérer dans le code des commentaires de documentation qui seront traités pour réaliser une documentation technique.

Il existe en C# quelques balises qui permettent de garantir une certaine cohérence entre les éléments de documentation et le code.⁶

Les couches BAL (Business Access Layer) et DAL (Data Access Layer) (voir les annexes : Annexe 6 – Code C# de la couche DAL pour la classe FonctionElectronique et Annexe 7 – Code C# de la couche BAL pour la classe FonctionElectronique), ont été commentées pour la génération d'une documentation automatique au format XML. Cette documentation automatique sera générée une fois le projet validé.

⁶ (Montantin, 2016)

6. Bilan

6.1. Bilan technique

D'un point de vue méthodologie, j'ai pu mettre en situation réelle la méthode agile vue en cours de mon année à l'université technologique. J'ai constaté que cette méthode était vraiment efficace et elle a très bien été reçue par le client. J'ai été frappé par l'évolution très rapide de l'interface homme machine notamment accentuée par l'outil de développement Visual Studio à l'aide du « glisser et déposer ». Finalement, l'application demeure très intuitive pour le client et les modifications apportées, même mineures, ont été appréciées.

Pour la modélisation, j'ai trouvé le langage UML très utile pour l'architecture des classes métiers. Cette étape a été pour moi fastidieuse car je ne possède encore qu'une trop faible expérience dans ce domaine. Cependant, j'ai bien conscience qu'une amélioration viendra avec le temps et l'habitude.

Le bilan technique est également très satisfaisant dans la mesure où j'ai pu mettre en pratique un patron de conception basé sur l'architecture N-Tiers. J'ai rapidement compris l'intérêt d'une telle conception même si c'est un peut-être trop détaillé pour notre cas simple.

Enfin, l'apprentissage du langage C# a été assez rapide. En effet cette variante du langage C créé par Microsoft dans les années 2000 possède certaines fonctions qui ressemblent à Java. Je suis ravi d'avoir eu l'opportunité d'avoir travaillé avec le Framework .NET car j'ai constaté un bon nombre d'offres d'emploi concernant ce langage et Framework sur le marché du travail.

6.2. Bilan humain

La découverte d'un secteur d'activité que je ne connaissais pas a été une expérience très enrichissante pour moi. J'aurai même personnellement voulu avoir plus de temps, pour apprendre quelques notions supplémentaires de programmation de bas niveau (driver, ...) auprès du bureau d'étude.

Les auditions auprès et par les différents protagonistes m'ont vraiment plu, et m'ont permis de m'immerger dans leur milieu plus rapidement. La conception de la base de données qui en débouche est un aspect que j'avais déjà exploré par le passé et confirme le fait que, pour moi, c'est l'un des aspects le plus intéressant d'un projet avec la phase d'analyse.

Pour ma part, j'ai trouvé ma première expérience, dans un espace de travail ouvert, plutôt réussit. On n'hésite moins à aller voir la personne concernée lors d'une demande spécifique et j'ai apprécié les pointes d'humour de certaines personnes.

Cependant, le trajet d'une heure pour l'aller et le retour a rapidement été une contrainte pour moi. Mais cet aspect a évidemment été gommé par l'intérêt que je portais au stage. Je n'ai pas vu le temps défiler.

7. Conclusion

Après un rapport d'audition nécessaire à la compréhension des méthodes de travail, nous avons effectué un cahier des charges aussi complet et exhaustif que possible. La conception de la base de données a rapidement été effectuée. Finalement, le modèle de départ était assez complet dans la mesure où il y a eu très peu de modifications de celui-ci au cours du stage.

Le backlog du produit a été alimenté au fur et à mesure des revues de sprint. Les récits utilisateurs ont été plutôt bien dimensionnés par la méthode agile. L'équipe projet a donc pu voir tous les bénéfices que pouvait apporter le cadre de travail SCRUM de la méthode agile.

Les objectifs du stage ont été atteints, et j'aurai, plus tard, un retour sur les bénéfices apportés par celui-ci. On espère un gain de 40% sur les temps de conception des devis qui peuvent s'étaler de trois semaines à 2 mois. On espère également gagner environ 10% sur le temps de conception global du projet (de 3 mois à 1 an) grâce à la réalisation des procès-verbaux de tests déjà créés. Des gains indirects pourront également impacter le projet, comme la réutilisation des composants en reprenant les mêmes références que les projets précédents, d'où des achats groupés et une potentielle meilleure gestion des stocks disponibles. Ces gains ne sont pas négligeables et ils pourront également avoir un fort impact sur l'image de marque auprès des clients futurs. En effet, plus de réactivité est un gage de satisfaction.

Il reste cependant un peu de développement à faire sur l'application. L'application est fonctionnelle et sera finalisée à la fin de mon stage. La documentation est également à finaliser. Enfin, la rédaction de test est prévue pour la suite. J'ai conscience de l'importance de ceux-ci et avec le temps et l'expérience, je serai plus habitué à les mettre en place rapidement. L'objectif sur le moyen terme est de faire du développement dirigé par les tests.

Bien qu'assez long et fastidieux, la modélisation UML a été une étape nécessaire et facilite grandement la conception du produit par la suite (tout particulièrement le diagramme de classe métier). En suivant, une ligne directrice, on évite toutes dérives. Ainsi il est plus facile de débugger l'application. Selon la loi de Pareto, j'ai pu constater que 20% des bugs étaient responsables de 80% des problèmes. Par extension, plus on essaye de trouver les bugs, plus les coûts engagés sont importants. Il faut savoir s'arrêter, tout l'art réside dans le fait du calcul subtil du gain par rapport à l'investissement, et ne dit pas-t-on que : la perfection n'existe pas.

8. Bibliographie

apognu. (2013, 10 30). *Utilisation de subversion*.

Récupéré sur OpenClassrooms: <https://openclassrooms.com/fr/courses/1129121-utilisation-de-subversion>

Doudoux, J.-M. (2010, 01 01). *La documentation du code en C#*.

Récupéré sur jmdoudoux: <https://www.jmdoudoux.fr/microsoft/doctechcsharp/doctechcsharp.htm>

Farinone, J.-M. (2011). *Architecture N-tier*.

Récupéré sur Cedric-Cnam: <http://cedric.cnam.fr/~farinone/SMB111/annee1011/architectureNTiers.pdf>

Hilaire, N. (2017, 08 14). *Apprenez à développer en c#*.

Récupéré sur OpenClassrooms: <https://openclassrooms.com/fr/courses/1526901-apprenez-a-developper-en-c/1527253-creez-un-projet-avec-visual-studio-express-2013-pour-windows-desktop>

Immobilis. (2010, Octobre 20). *L'architecture multicouche mise en œuvre sur une application Web ASP.Net*.

Récupéré sur Developpez.com: <https://immobilis.developpez.com/articles/dotnet/architecture-multicouche-asp-net/>

Lothon, F. (2015). *Introduction aux méthodes agiles et Scrum*.

Récupéré sur L'agiliste: <https://agiliste.fr/introduction-methodes-agiles/>

Montantin, M. (2016, 10 29). *Documentation du code C# en 1 min.*

Récupéré sur cdiese: cdiese.

Table des illustrations

Figure 1 : Hiérarchie globale de l'entreprise	2
Figure 2 : Organigramme de la production	2
Figure 3 : Diagramme Bête à corne	4
Figure 4 : Principe de fonctionnement SCRUM	5
Figure 5 : Equipe Scrum	6
Figure 6 : Planification sous GanttProject	6
Figure 7 : Organisation Agile sous Trello	6
Figure 8 : Diagramme pieuvre	7
Figure 9 : MCD produit final	10
Figure 10 : Diagramme de composant N-Tiers.....	11
Figure 11 : Diagramme cas d'utilisation	12
Figure 12 : Diagramme de séquence – Ajouter une fonction électronique	14
Figure 13 : Diagramme de séquence – Modifier une fonction électronique	14
Figure 14 : Diagramme de séquence – Supprimer une fonction électronique	15
Figure 15 : Diagramme de séquence – Valider une fonction électronique.....	15
Figure 16 : Diagramme de séquence – Rechercher une fonction électronique.....	15
Figure 17 : Diagramme de classe métier	16
Figure 18 : IHM fenêtre principale	17
Figure 19 : IHM liste d'attente	17
Figure 20 : IHM ajouter/modifier/supprimer et valider une fonction	18
Figure 21 : Algorithme de la fonction de recherche.....	19
Figure 22 - Architecture général du projet.....	20
Figure 23 : Ecran d'accueil	20
Figure 24 : Fenêtre ajouter ou modifier une fonction (mode ajouter)	21
Figure 25 : Modification d'un projet.....	21
Figure 26 : Outil de capture d'écran	22
Figure 27 : Zone capturée.....	22
Figure 28 : Manipulation image capturée	23
Figure 29 : Insertion des données	23
Figure 30 : Fenêtre ajouter ou modifier une fonction (mode modification)	24
Figure 31 : Filtre de recherche.....	24
Figure 32 : MCD - Solution 1.....	2
Figure 33 : MCD - Solution 2.....	2
Figure 34 : MCD - Solution 3	2
Tableau 1 : Tableau des fonctions de service.....	7
Tableau 2 : Fonctions de service, critères, niveau et flexibilité.....	8
Tableau 3 : Champs base de données	10
Tableau 4 : Cas d'utilisation - Ajouter fonction électronique.....	12
Tableau 5 : Cas d'utilisation - Modifier fonction électronique.....	13
Tableau 6 : Cas d'utilisation - Supprimer fonction électronique	13
Tableau 7 : Cas d'utilisation - Valider fonction électronique	13
Tableau 8 : Cas d'utilisation - Rechercher fonction électronique	13
Tableau 9 : Diagramme état transition de la fenêtre ajouter/modifier une fonction	16
Tableau 10 : Fonctionnalités de l'écran d'accueil.....	20

Annexe 1 – Audit de fonctionnement

Avec l'aide de Nicolas Vinot, nous avons réalisé un rapport d'audit afin de mieux comprendre les attentes liées au projet. Voici le résultat de cette audition :

« Lorsqu'un projet est terminé, le chef de projet veut inscrire dans une bdd, le nom du projet, sa date de création, le lien SVN (lien vers lequel pointe tout le contenu du projet) ~~ainsi que la nomenclature de tous les composants (BOM)~~.

A la suite de ceci, il veut décomposer l'ensemble du projet en plusieurs fonctions génériques (ex : gestion de l'alimentation. A cette fonction sera associée une image au format jpeg le descriptif de la fonction (descriptif dédié à la fonction du projet), le coût estimé (prix d'achat + montage) de celle-ci, ainsi qu'un booléen permettant de connaître si la fonction a été testée ou non. Un lien SVN des éventuelles PV de test devra également apparaître.

Cette fonction aura une UNIQUE entrée/sortie électrique en termes de tension et courant (de façon à calculer la puissance électrique en entrée et la puissance électrique en sortie). Pour une fonction d'un projet, on pourra lui associer plusieurs entrées/sorties physiques (BUS CAN, VBAT_MEAS, MESURE_VIN, ...). A chaque association correspondra un type de signal (analogique, numérique, pwn, ...). »

Annexe 2 – Modèles conceptuels de données

La solution 1 est la plus simple mais la plus rigide dans le temps. Les fonctions n'auront comme critères de détails que ceux définis en colonne (tension_input, ...). Cette solution facilite grandement le travail de recherche mais elle est trop rigide dans le temps. Elle n'a pas été retenue.

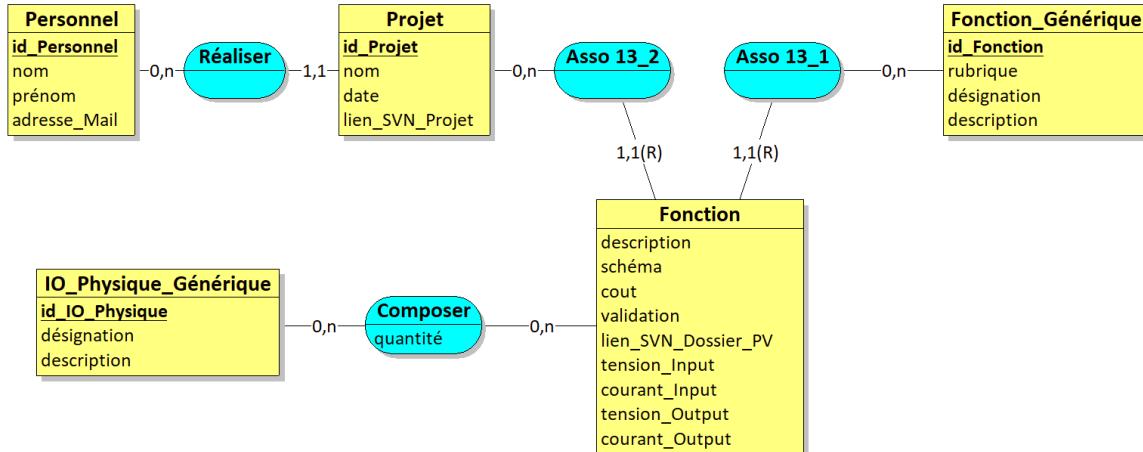


Figure 32 : MCD - Solution 1

Avec cette solution nous avons une liste de critères génériques qui pourront qualifier les fonctions. A chaque fonction nous définirons un niveau pour le critère générique. Le lien entre fonction générique et critères générique est un Template (modèle récurrent). Par exemple pour une fonction générique donnée, nous auront x critères à remplir.

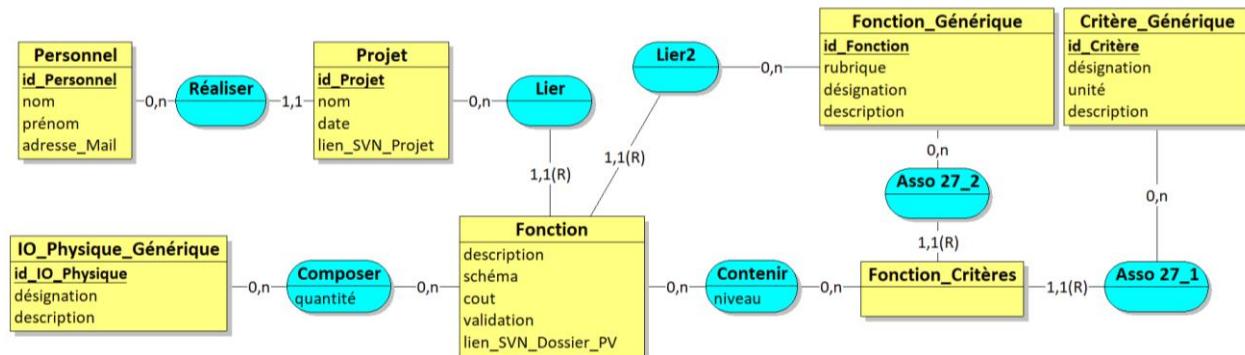


Figure 33 : MCD - Solution 2

La solution 3 est identique à la solution 2 mais sans le *Template*. Cette solution a été adoptée car c'est la plus souple et donc par conséquent la plus pérenne dans le temps.

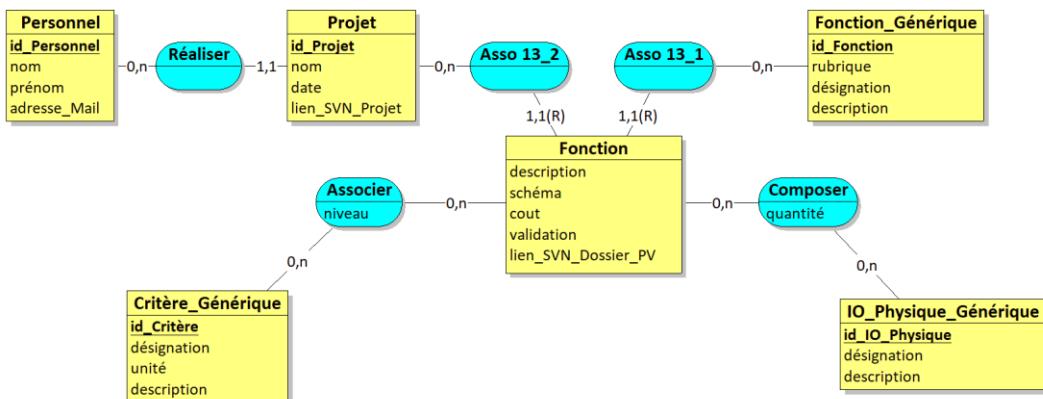


Figure 34 : MCD - Solution 3

Annexe 3 – Requêtes de la création des tables de la base de données

```

CREATE TABLE `Personnel` (
  `id_personnel` int(11) NOT NULL AUTO_INCREMENT,
  `nom` varchar(31) NOT NULL,
  `prénom` varchar(31) NOT NULL,
  `adresse_mail` varchar(63) DEFAULT NULL,
  PRIMARY KEY (`id_personnel`),
  UNIQUE KEY `nom` (`nom`,`prénom`)
) ENGINE=InnoDB AUTO_INCREMENT=52 DEFAULT CHARSET=utf8;

CREATE TABLE `Projet` (
  `id_projet` varchar(31) NOT NULL,
  `date_projet` date DEFAULT NULL,
  `lien SVN_projet` varchar(255) DEFAULT NULL,
  `id_personnel` int(11) NOT NULL,
  PRIMARY KEY (`id_projet`),
  KEY `fk_personnel` (`id_personnel`),
  CONSTRAINT `fk_personnel` FOREIGN KEY (`id_personnel`) REFERENCES `Personnel`(`id_personnel`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `Critère_Générique` (
  `id_critère` int(11) NOT NULL AUTO_INCREMENT,
  `désignation` varchar(31) NOT NULL,
  `unité` varchar(31) NOT NULL,
  `donnée_chiffrée` tinyint(1) NOT NULL,
  `description` varchar(63) DEFAULT NULL,
  `modifiable` tinyint(4) NOT NULL DEFAULT '1',
  PRIMARY KEY (`id_critère`),
  UNIQUE KEY `désignation_UNIQUE` (`désignation`)
) ENGINE=InnoDB AUTO_INCREMENT=16 DEFAULT CHARSET=utf8;

CREATE TABLE `Fonction_Générique` (
  `id_fonction` int(11) NOT NULL AUTO_INCREMENT,
  `rubrique` varchar(63) NOT NULL,
  `désignation` varchar(63) NOT NULL,
  `description` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id_fonction`),
  UNIQUE KEY `désignation` (`désignation`)
) ENGINE=InnoDB AUTO_INCREMENT=52 DEFAULT CHARSET=utf8;

CREATE TABLE `IO_Physique_Générique` (
  `id_io_physique` int(11) NOT NULL AUTO_INCREMENT,
  `désignation` varchar(31) NOT NULL,
  `description` varchar(255) DEFAULT NULL,
  PRIMARY KEY (`id_io_physique`)
) ENGINE=InnoDB AUTO_INCREMENT=4 DEFAULT CHARSET=utf8;

CREATE TABLE `Fonction` (
  `id_projet` varchar(31) NOT NULL,
  `id_fonction` int(11) NOT NULL,
  `description` text NOT NULL,
  `schéma` varchar(255) NOT NULL,
  `tension_entree_min` float NOT NULL,
  `tension_entree_max` float NOT NULL,
  `intensité_entree_min` float NOT NULL,
  `intensité_entree_max` float NOT NULL,
  `tension_sortie_min` float NOT NULL,
  `tension_sortie_max` float NOT NULL,
  `intensité_sortie_min` float NOT NULL,
  `intensité_sortie_max` float NOT NULL,
  `cout` double DEFAULT NULL,
  `validation` tinyint(1) NOT NULL,
  ...
)

```

```

`lien SVN_Dossier_PV` varchar(255) DEFAULT NULL,
`en_attente` tinyint(1) NOT NULL DEFAULT '1',
PRIMARY KEY (`id_projet`, `id_fonction`),
KEY `fk_fonction_générique` (`id_fonction`),
CONSTRAINT `fk_fonction_générique` FOREIGN KEY (`id_fonction`) REFERENCES
`Fonction_Générique` (`id_fonction`) ON DELETE NO ACTION ON UPDATE NO ACTION,
CONSTRAINT `fk_projet` FOREIGN KEY (`id_projet`) REFERENCES `Projet` (`id_projet`) ON
DELETE NO ACTION ON UPDATE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `Associer_Critère` (
`id_projet` varchar(31) NOT NULL,
`id_fonction` int(11) NOT NULL,
`id_critère` int(11) NOT NULL,
`niveau_nombre` float DEFAULT NULL,
`niveau_texte` varchar(63) DEFAULT NULL,
PRIMARY KEY (`id_projet`, `id_fonction`, `id_critère`),
KEY `fk_critère` (`id_critère`),
CONSTRAINT `fk_critère` FOREIGN KEY (`id_critère`) REFERENCES `Critère_Générique`(
`id_critère`),
CONSTRAINT `fk_fonction` FOREIGN KEY (`id_projet`, `id_fonction`) REFERENCES `Fonction`(
`id_projet`, `id_fonction`) ON DELETE CASCADE ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

CREATE TABLE `Associer_IO_Physique` (
`id_projet` varchar(31) NOT NULL,
`id_fonction` int(11) NOT NULL,
`id_io_physique` int(11) NOT NULL,
`quantité` int(11) NOT NULL,
PRIMARY KEY (`id_projet`, `id_fonction`, `id_io_physique`),
KEY `fk_iophysique` (`id_io_physique`),
CONSTRAINT `fk_fonction2` FOREIGN KEY (`id_projet`, `id_fonction`) REFERENCES
`Fonction` (`id_projet`, `id_fonction`) ON DELETE CASCADE ON UPDATE CASCADE,
CONSTRAINT `fk_iophysique` FOREIGN KEY (`id_io_physique`) REFERENCES
`IO_Physique_Générique` (`id_io_physique`) ON DELETE NO ACTION
) ENGINE=InnoDB DEFAULT CHARSET=utf8;

```

Annexe 4 – Procédures stockées et fonctions pour le calcul du taux de correspondance

Voici le code de la fonction permettant le calcul du taux avec les données principales :

```

CREATE DEFINER=`root_etudes`@`%` FUNCTION `calculerTauxCorrespondance` (
    tInputMinC FLOAT, tInputMinV FLOAT, tInputMinTarget FLOAT,
    tInputMaxC FLOAT, tInputMaxV FLOAT, tInputMaxTarget FLOAT,
    iInputMinC FLOAT, iInputMinV FLOAT, iInputMinTarget FLOAT,
    iInputMaxC FLOAT, iInputMaxV FLOAT, iInputMaxTarget FLOAT,
    tOutputMinC FLOAT, tOutputMinV FLOAT, tOutputMinTarget FLOAT,
    tOutputMaxC FLOAT, tOutputMaxV FLOAT, tOutputMaxTarget FLOAT,
    iOutputMinC FLOAT, iOutputMinV FLOAT, iOutputMinTarget FLOAT,
    iOutputMaxC FLOAT, iOutputMaxV FLOAT, iOutputMaxTarget FLOAT
) RETURNS int(11)
BEGIN
    DECLARE outpout FLOAT ;
    DECLARE valeurNull FLOAT;
    SET outpout = 0;
    SET valeurNull =9999;

    /* ****
    /*Calcul de la première occurrence concernant la plage de tension input */
    IF (tInputMinV <= tInputMinTarget or tInputMinTarget = valeurNull)
        THEN SET outpout = outpout + 12.5;
    ELSEIF tInputMinC <= tInputMinTarget
        THEN SET outpout = outpout + 5;
    END IF;
    IF (tInputMaxV >= tInputMaxTarget or tInputMaxTarget = valeurNull)
        THEN SET outpout = outpout + 12.5;
    ELSEIF tInputMaxC >= tInputMaxTarget
        THEN SET outpout = outpout + 5;
    END IF;
    /*Fin Calcul de la première occurrence concernant la plage de tension input*/
    ****

    /* ****
    /*Fin Calcul de la première occurrence concernant la plage d'intensité input*/
    IF (iInputMinV <= iInputMinTarget or iInputMinTarget = valeurNull)
        THEN SET outpout = outpout + 12.5;
    ELSEIF iInputMinC <= iInputMinTarget
        THEN SET outpout = outpout + 5;
    END IF;
    IF (iInputMaxV >= iInputMaxTarget or iInputMaxTarget = valeurNull)
        THEN SET outpout = outpout + 12.5;
    ELSEIF iInputMaxC >= iInputMaxTarget
        THEN SET outpout = outpout + 5;
    END IF;
    /*Fin Calcul de la première occurrence concernant la plage d'intensité input*/
    ****

    /* ****
    /*Calcul de la première occurrence concernant la plage de tension output */
    IF (tOutputMinV <= tOutputMinTarget or tOutputMinTarget = valeurNull)
        THEN SET outpout = outpout + 12.5;
    ELSEIF tOutputMinC <= tOutputMinTarget
        THEN SET outpout = outpout + 5;
    END IF;
    IF (tOutputMaxV >= tOutputMaxTarget or tOutputMaxTarget = valeurNull)
        THEN SET outpout = outpout + 12.5;
    ELSEIF tOutputMaxC >= tOutputMaxTarget
        THEN SET outpout = outpout + 5;
    END IF;
    /*Fin Calcul de la première occurrence concernant la plage de tension input*/
    ****

```

```

/*Fin Calcul de la première occurrence concernant la plage d'intensité input*/
IF (iOutputMinV <= iOutputMinTarget or iOutputMinTarget = valeurNull)
    THEN SET outpout = outpout + 12.5;
ELSEIF iOutputMinC <= iOutputMinTarget
    THEN SET outpout = outpout + 5;
END IF;
IF (iOutputMaxV >= iOutputMaxTarget or iOutputMinTarget = valeurNull)
    THEN SET outpout = outpout + 12.5;
ELSEIF iOutputMaxC >= iOutputMaxTarget
    THEN SET outpout = outpout + 5;
END IF;
/*Fin Calcul de la première occurrence concernant la plage d'intensité input*/
*****
```

RETURN outpout;

END

Voici le code pour le calcul du taux avec les critères sélectionnés (jusqu'à 6). Seuls les critères 1 et 2 sont montrés ici.

```

CREATE DEFINER=`root_etudes`@`%` FUNCTION `calculerTauxCrit`(
    idCrit1 int, signCrit1 int, valCrit1 float, valCibleCrit1 float
    idCrit2 int, signCrit2 int, valCrit2 float, valCibleCrit2 float
) RETURNS float
BEGIN
DECLARE outpout FLOAT ;
DECLARE valeurNull FLOAT;
SET outpout = 0;
SET valeurNull = 9999;

*****
```

*****Calcul de la première occurrence pour le critère 1 *****

```

IF (idCrit1 = valeurNull OR valCrit1 is null) THEN
    SET outpout = outpout + 100/6;
ELSE
    BEGIN
        IF (signCrit1 = -1) THEN
            BEGIN
                IF (valCrit1 <= valCibleCrit1) THEN
                    SET outpout = outpout + 100/6;
            END IF;
        END;
        ELSEIF (signCrit1 = 0) THEN
            BEGIN
                IF (valCrit1 = valCibleCrit1) THEN
                    SET outpout = outpout + 100/6;
            END IF;
        END;
        ELSEIF (signCrit1 = 1) THEN
            BEGIN
                IF (valCrit1 >= valCibleCrit1) THEN
                    SET outpout = outpout + 100/6;
            END IF;
        END;
    END IF;
*****Fin Calcul de la première occurrence pour le critère 1 *****
```

```

*****Calcul de la première occurrence pour le critère 2 *****
IF (idCrit2 = valeurNull OR valCrit2 is null) THEN
    SET outpout = outpout + 100/6;
ELSE
    BEGIN
        IF (signCrit2 = -1) THEN
            BEGIN
                IF (valCrit2 <= valCibleCrit2) THEN
                    SET outpout = outpout + 100/6;
            END
```

```

        END IF;
    END;
    ELSEIF (signCrit1 = 0) THEN
        BEGIN
            IF (valCrit2 = valCibleCrit2) THEN
                SET outpout = outpout + 100/6;
            END IF;
        END;
    ELSEIF (signCrit2 = 1) THEN
        BEGIN
            IF (valCrit2 >= valCibleCrit2) THEN
                SET outpout = outpout + 100/6;
            END IF;
        END;
    END IF;
END;

*****Fin Calcul de la première occurrence pour le critère 2 *****/
*****idem pour crit 3 4 5 6   */

RETURN outpout;
END

```

Afin de calculer le taux principal et fournir une liste des fonctions, nous avons défini une procédure stockée. On remarque qu'il manque encore le calcul du taux avec les lo physiques (idem que le calcul du taux précédent).

```

CREATE DEFINER=`root_etudes`@`localhost` PROCEDURE `filtrerPremier`(
    `fonction` INT,
    `tensionInMin` FLOAT,
    `tensionInMax` FLOAT,
    `intensiteInMin` FLOAT,
    `intensiteInMax` FLOAT,
    `tensionOutMin` FLOAT,
    `tensionOutMax` FLOAT,
    `intensiteOutMin` FLOAT,
    `intensiteOutMax` FLOAT,
    idCrit1 int, signCrit1 int, valCibleCrit1 float,
    idCrit2 int, signCrit2 int, valCibleCrit2 float
)
NO SQL
BEGIN

CASE fonction
WHEN 9999 THEN
    BEGIN
        CREATE TEMPORARY TABLE IF NOT EXISTS critereAss AS(
        SELECT id_fonction, id_projet,
        SUM(CASE id_critère WHEN 1 THEN niveau_nombre END) AS tensionInputCritMin,
        SUM(CASE id_critère WHEN 2 THEN niveau_nombre END) AS tensionInputCritMax,
        SUM(CASE id_critère WHEN 3 THEN niveau_nombre END) AS intensitéInputCritMin,
        SUM(CASE id_critère WHEN 4 THEN niveau_nombre END) AS intensitéInputCritMax,
        SUM(CASE id_critère WHEN 5 THEN niveau_nombre END) AS tensionOutputCritMin,
        SUM(CASE id_critère WHEN 6 THEN niveau_nombre END) AS tensionOutputCritMax,
        SUM(CASE id_critère WHEN 7 THEN niveau_nombre END) AS intensitéOutputCritMin,
        SUM(CASE id_critère WHEN 8 THEN niveau_nombre END) AS intensitéOutputCritMax,
        SUM(CASE id_critère WHEN idCrit1 THEN niveau_nombre END) AS valcrit1,
        SUM(CASE id_critère WHEN idCrit2 THEN niveau_nombre END) AS valcrit2
        FROM Projets_Etudes.Associer_Critère GROUP BY id_fonction, id_projet
    );
    SELECT *, 
    Projets_Etudes.calculerTauxCorrespondance(
        tensionInputCritMin, tension_entree_min, tensionInMin,
        tensionInputCritMax, tension_entree_max, tensionInMax,
        intensitéInputCritMin, intensité_entree_min, intensiteInMin,
        intensitéInputCritMax, intensité_entree_max, intensiteInMax,
        tensionOutputCritMin, tension_sortie_min, tensionOutMin,
        tensionOutputCritMax, tension_sortie_max, tensionOutMax,
    )
    
```

```

intensitéOutpuCrittMin,      intensite_sortie_min,      intensiteOutMin,
intensitéOutputCritMax,      intensite_sortie_max,      intensiteOutMax
) as Taux from Fonction LEFT JOIN critereAss
ON critereAss.id_fonction = Fonction.id_fonction AND critereAss.id_projet
= Fonction.id_projet
;

DROP TABLE critereAss;
END;
ELSE
BEGIN
    CREATE TEMPORARY TABLE IF NOT EXISTS critereAss as
    SELECT id_fonction, id_projet,
    SUM(CASE id_critère WHEN 1 THEN niveau_nombre END) AS tensionInputCritMin,
    SUM(CASE id_critère WHEN 2 THEN niveau_nombre END) AS tensionInputCritMax,
    SUM(CASE id_critère WHEN 3 THEN niveau_nombre END) AS intensitéInputCritMin,
    SUM(CASE id_critère WHEN 4 THEN niveau_nombre END) AS intensitéInputCritMax,
    SUM(CASE id_critère WHEN 5 THEN niveau_nombre END) AS tensionOutputCritMin,
    SUM(CASE id_critère WHEN 6 THEN niveau_nombre END) AS tensionOutputCritMax,
    SUM(CASE id_critère WHEN 7 THEN niveau_nombre END) AS intensitéOutpuCrittMin,
    SUM(CASE id_critère WHEN 8 THEN niveau_nombre END) AS intensitéOutpuCrittMax,
    SUM(CASE id_critère WHEN idCrit1 THEN niveau_nombre END) AS valcrit1,
    SUM(CASE id_critère WHEN idCrit2 THEN niveau_nombre END) AS valcrit2
    FROM Projets_Etudes.Associer_Critère WHERE id_fonction = fonction GROUP BY
id_fonction, id_projet
);

SELECT Fonction.*,
Projets_Etudes.calculerTauxCorrespondance(
tensionInputCritMin,      tension_entree_min,      tensionInMin,
tensionInputCritMax,      tension_entree_max,      tensionInMax,
intensitéInputCritMin,      intensité_entree_min,      intensiteInMin,
intensitéInputCritMax,      intensité_entree_max,      intensiteInMax,
tensionOutputCritMin,      tension_sortie_min,      tensionOutMin,
tensionOutputCritMax,      tension_sortie_max,      tensionOutMax,
intensitéOutpuCrittMin,      intensite_sortie_min,      intensiteOutMin,
intensitéOutpuCrittMax,      intensite_sortie_max,      intensiteOutMax
) * 3/5 +
Projets_Etudes.calculerTauxCrit(idCrit1,signCrit1,valcrit1, valCibleCrit1,
idCrit2,signCrit2,valcrit2, valCibleCrit2) *1/5 as Taux
from Fonction LEFT JOIN critereAss
ON critereAss.id_fonction = Fonction.id_fonction AND critereAss.id_projet
= Fonction.id_projet
WHERE Fonction.id_fonction = fonction
ORDER BY Taux1 desc;
DROP TABLE critereAss;
END;
END CASE;

END

```

Annexe 5 – Code C# de la couche métier BEL

```

using System; System.Collections.Generic; System.Linq; System.Text; System.Globalization;

namespace BEL
{
    public class Personnel
    {
        public int idPersonnel { get; set; }
        public string nom { get; set; }
        public string prenom { get; set; }
        public string mail { get; set; }
        public string getCredit()
        {
            return this.prenom.Substring(0, 1) + ". " + this.nom ;
        }

        public override string ToString()
        {
            return this.nom + " " + this.prenom;
        }
    }
    public class Projet
    {
        public string idProjet { get; set; }
        public DateTime dateProjet { get; set; }
        public string lienSnvProjet { get; set; }
        public Personnel personnel { get; set; }
        public string formatDateSql()
        {
            return this.dateProjet.Year + "-" + this.dateProjet.Month + "-" +
this.dateProjet.Day;
        }
        public string getDate()
        {
            CultureInfo ci = new CultureInfo("fr-Fr");
            string date;
            date = (ci.DateTimeFormat.GetMonthName(this.dateProjet.Month).Length > 7) ?
ci.DateTimeFormat.GetMonthName(this.dateProjet.Month).Substring(0, 3) :
ci.DateTimeFormat.GetMonthName(this.dateProjet.Month);

            return date + " " + this.dateProjet.Year;
        }

        public override string ToString()
        {
            return this.dateProjet.ToString("dd/MM/yyyy") + " - " + this.idProjet;
        }
    }
    public class IoPhysiqueGenerique
    {
        public int idIophysique { get; set; }
        public string designation { get; set; }
        public string description { get; set; }
        public override string ToString()
        {
            return this.designation;
        }
    }
    public class CritereGenerique
    {
        public int idCritere { get; set; }
        public string designation { get; set; }
        public string unite { get; set; }
        public Boolean donneeChiffree { get; set; }
        public string description { get; set; }
        public override string ToString()
        {
            return this.designation;
        }
    }
}

```

```

}

public class FonctionGenerique
{
    public int      idFonction      { get; set; }
    public string   rubrique       { get; set; }
    public string   designation    { get; set; }
    public string   description    { get; set; }
    public override string ToString()
    {
        return this.designation ;
    }
}
public class IoPhysiqueAssociee : IEquatable<IoPhysiqueAssociee>
{
    public IoPhysiqueGenerique ioPhysique { get; set; }
    public int                  quantite   { get; set; }

    public bool Equals(IoPhysiqueAssociee other)
    {
        if (other == null) return false;
        return (this.ioPhysique.idIophysique.Equals(other.ioPhysique.idIophysique));
    }
    public override string ToString()
    {
        return this.ioPhysique.designation + " : " + this.quantite;
    }
}
public class CritereAssociee : IEquatable<CritereAssociee>
{
    public CritereGenerique critere { get; set; }
    public string      valeurNbr   { get; set; }
    public string      valeurTexte { get; set; }

    public bool Equals(CritereAssociee other)
    {
        if (other == null) return false;
        return (this.critere.idCritere.Equals(other.critere.idCritere));
    }
    public override string ToString()
    {
        return this.critere.designation + " " + this.valeurTexte + " : " +
this.valeurNbr + " " + this.critere.unite;
    }
}

public class FonctionElectronique
{
    public Projet          projet      { get; set; }
    public FonctionGenerique fonction    { get; set; }
    public string           description { get; set; }
    public string           schema     { get; set; }
    public float            tensionInputMin { get; set; }
    public float            tensionInputMax { get; set; }
    public float            intensiteInputMin { get; set; }
    public float            intensiteInputMax { get; set; }
    public float            tensionOutputMin { get; set; }
    public float            tensionOutputMax { get; set; }
    public float            intensiteOutputMin { get; set; }
    public float            intensiteOutputMax { get; set; }
    public float            cout        { get; set; }
    public Boolean          validation  { get; set; }
    public string           lienSVNTest { get; set; }
    public List<IoPhysiqueAssociee> listIO    { get; set; }
    public List<CritereAssociee>  listCritere { get; set; }
    public Boolean          enAttente  { get; set; }
    public float             taux       { get; set; }

    public FonctionElectronique()
    {
        this.listIO = new List<IoPhysiqueAssociee>();
        this.listCritere = new List<CritereAssociee>();
    }
}

```

```

public void ajouterCritere(CritereAssociee crit)
{
    if (!this.listCritere.Contains(crit))
    {
        this.listCritere.Add(crit);
    }
}
public void modifierCritere(CritereAssociee crit)
{
    foreach (CritereAssociee c in this.listCritere)
    {
        if (c.critere.idCritere == crit.critere.idCritere)
            c.valeurNbr = crit.valeurNbr;
            c.valeurTexte = crit.valeurTexte;
    }
}
public void supprimerCritere(CritereAssociee crit)
{
    if (this.listCritere.Contains(crit))
    {
        this.listCritere.Remove(crit);
    }
}
public void ajouterIoPhys(IoPhysiqueAssociee ioPhys)
{
    if (!this.listIO.Contains(ioPhys))
    {
        this.listIO.Add(ioPhys);
    }
}
public void modifierIoPhys(IoPhysiqueAssociee ioPhys)
{
    foreach (IoPhysiqueAssociee io in this.listIO)
    {
        if (io.ioPhysique.idIophysique == ioPhys.ioPhysique.idIophysique)
            io.quantite = ioPhys.quantite;
    }
}
public void supprimerIoPhys(IoPhysiqueAssociee ioPhys)
{
    if (this.listIO.Contains(ioPhys))
    {
        this.listIO.Remove(ioPhys);
    }
}
public string getPlageTensionInput()
{
    return "Tension : " + this.tensionInputMin + " à " + this.tensionInputMax +
" Volts";
}
public string getPlageIntensiteInput()
{
    return "Intensité : " + this.intensiteInputMin + " à " +
this.intensiteInputMax + " Ampères";
}
public string getPlageTensionOutput()
{
    return "Tension : " + this.tensionOutputMin + " à " + this.tensionOutputMax +
" Volts";
}
public string getPlageIntensiteOutput()
{
    return "Intensité : " + this.intensiteOutputMin + " à " +
this.intensiteOutputMax + " Ampères";
}
public override string ToString()
{
    return this.projet.idProjet + " - " + this.fonction.designation;
}
}
}

```

Annexe 6 – Code C# de la couche DAL pour la classe FonctionElectronique

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using BEL;
using System.Data.Common;
using System.Data;
using MySql.Data.MySqlClient;

namespace DAL
{
    /*
     * //Création des méthodes CRUD pour la classe FonctionElectronique
     */
    /// <summary>
    /// <c>FonctionGenData</c> class.
    /// Contient les méthodes CRUD de la classe FonctionElectronique
    /// </summary>
    /// <remarks>
    /// <para>Cette classe peut ajouter, modifier et supprimer un objet FonctionElectroniques</para>
    /// <para>Cette classe permet d'obtenir la liste entière des FonctionElectronique</para>
    /// </remarks>
    ///
    public class FonctionData
    {
        private DbConnexion db = new DbConnexion();
        private ProjetData prjManager = new ProjetData();
        private FonctionGenData fctGenData = new FonctionGenData();
        private IoPhysiqueAssocieData ioPhysAssData = new IoPhysiqueAssocieData();
        private CritereAssocieData critAssData = new CritereAssocieData();

        /// <summary>
        /// Ajouter un <paramref name="fctElec"/> dans la table Fonction de la base de données.
        /// </summary>
        /// <returns>
        /// Nombre de lignes affectées
        /// </returns>
        /// <param name="fctElec">un objet FonctionElectronique</param>
        public int ajouterFonction(FonctionElectronique fctElec)
        {
            DbCommand cmd = db.CreerCommande();
            cmd.CommandType = CommandType.Text;
            cmd.CommandText = "INSERT INTO Fonction "
                + "(id_projet, "
                + "id_fonction , "
                + "description , "
                + "schéma , "
                + "tension_entree_min , "
                + "tension_entree_max , "
                + "intensite_entree_min , "
                + "intensite_entree_max , "
                + "tension_sortie_min , "
                + "tension_sortie_max , "
                + "intensite_sortie_min , "
                + "intensite_sortie_max , "
                + "cout , "
                + "validation , "
                + "lien SVN_Dossier_PV ) "
                + "VALUES (" + fctElec.projet.idProjet + ", "
                + fctElec.fonction.idFonction + ", "
                + fctElec.description + ", "
                + fctElec.schema + ", "

```

```

        + fctElec.tensionInputMin.ToString().Replace(", ", ".") +
        + fctElec.tensionInputMax.ToString().Replace(", ", ".") +
        + fctElec.intensiteInputMin.ToString().Replace(", ", ".") +
        + fctElec.intensiteInputMax.ToString().Replace(", ", ".") +
        + fctElec.tensionOutputMin.ToString().Replace(", ", ".") +
        + fctElec.tensionOutputMax.ToString().Replace(", ", ".") +
        + fctElec.intensiteOutputMin.ToString().Replace(", ", ".") +
        + fctElec.intensiteOutputMax.ToString().Replace(", ", ".") +
        + fctElec.cout.ToString().Replace(", ", ".") + ", "
        + fctElec.validation + ", "
        + fctElec.lienSVNTest + "'");
    return db.ExecuterRequete(cmd);
}

/// <summary>
/// Obtenir un objet FonctionElectronique par son <paramref name="idFonction"/>
/// </summary>
/// <returns>
/// Un objet FonctionElectronique
/// </returns>
/// <param name="idFonction">Identifiant de la fonction</param>
public FonctionElectronique getFonctionById(int idFonction, string idprojet)
{
    FonctionElectronique fctElect = new FonctionElectronique();
    DbCommand cmd = db.CreerCommande();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "Select * from Fonction where id_fonction=" + idFonction +
" and id_projet ='" + idprojet + "'";
    DataTable table = db.CreerDatatable(cmd);
    if (table.Rows.Count >= 1)
    {
        fctElect.projet = prjManager.getProjetById((string)table.Rows[0][0]);
        fctElect.fonction = fctGenData.getFonctionGenById((int)table.Rows[0][1]);
        fctElect.description = (string)table.Rows[0][2].ToString();
        fctElect.schema = (string)table.Rows[0][3].ToString();

        //les données io tension et intensité
        fctElect.tensionInputMin = float.Parse(table.Rows[0][4].ToString());
        fctElect.tensionInputMax = float.Parse(table.Rows[0][5].ToString());
        fctElect.intensiteInputMin = float.Parse(table.Rows[0][6].ToString());
        fctElect.intensiteInputMax = float.Parse(table.Rows[0][7].ToString());
        fctElect.tensionOutputMin = float.Parse(table.Rows[0][8].ToString());
        fctElect.tensionOutputMax = float.Parse(table.Rows[0][9].ToString());
        fctElect.intensiteOutputMin = float.Parse(table.Rows[0][10].ToString());
        fctElect.intensiteOutputMax = float.Parse(table.Rows[0][11].ToString());

        fctElect.cout = float.Parse(table.Rows[0][12].ToString());
        fctElect.validation = (Boolean)table.Rows[0][13];
        fctElect.lienSVNTest = (string)table.Rows[0][14].ToString();
        fctElect.enAttente = (Boolean)table.Rows[0][15];
        //fctElect.listIO =
    }
    fctIoManager.getListIOAssocieeByFonction((string)table.Rows[0][0],
    (int)table.Rows[0][1]);
    //fctElect.listCritere =
    fctCritManager.getListCritAssocieeByFonction((string)table.Rows[0][0],
    (int)table.Rows[0][1]);
}
return fctElect;
}

/// <summary>
/// Obtenir une liste d'objet FonctionElectronique
/// </summary>

```

```

/// <returns>
/// Une liste d'objet FonctionElectronique
/// </returns>
public List<FonctionElectronique> getListFonction()
{
    List<FonctionElectronique> listFonction = new List<FonctionElectronique>();

    DbCommand cmd = db.CreerCommande();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "Select * from Fonction";
    DataTable table = db.CreerDatatable(cmd);

    foreach (DataRow row in table.Rows)
    {
        FonctionElectronique fctElect = new FonctionElectronique();

        fctElect.projet = prjManager.getProjetById((string)row[0]);
        fctElect.fonction = fctGenData.getFonctionGenById((int)row[1]);
        fctElect.description = (string)row[2].ToString();
        fctElect.schema = (string)row[3].ToString();
        //les données io tension et intensité
        fctElect.tensionInputMin = float.Parse(row[4].ToString());
        fctElect.tensionInputMax = float.Parse(row[5].ToString());
        fctElect.intensiteInputMin = float.Parse(row[6].ToString());
        fctElect.intensiteInputMax = float.Parse(row[7].ToString());
        fctElect.tensionOutputMin = float.Parse(row[8].ToString());
        fctElect.tensionOutputMax = float.Parse(row[9].ToString());
        fctElect.intensiteOutputMin = float.Parse(row[10].ToString());
        fctElect.intensiteOutputMax = float.Parse(row[11].ToString());

        fctElect.cout = float.Parse(row[12].ToString());
        fctElect.validation = (Boolean)row[13];
        fctElect.lienSVNTest = (string)row[14].ToString();
        fctElect.enAttente = (Boolean)row[15];
        fctElect.listIO = ioPhysAssData.getListIoAssocieeByFonction(fctElect);
        fctElect.listCritere =
critAssData.getListCritAssocieeByFonction(fctElect);
        listFonction.Add(fctElect);
    }

    return listFonction;
}

/// <summary>
/// Obtenir une liste d'objet FonctionElectronique par son <paramref name="f"/>
/// </summary>
/// <returns>
/// Une liste d'objet FonctionElectronique
/// </returns>
/// <param name="f">Filtre</param>
public List<FonctionElectronique> getListFonction(Filtre f)
{
    List<FonctionElectronique> listFonction = new List<FonctionElectronique>();

    DbCommand cmd = db.CreerCommande();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "call filtrerPremier (" + f.fonction.idFonction + ", "
    + f.tensionInMin + ", " + f.tensionInMax + ", "
    + f.intensiteInMin + ", " + f.intensiteInMax + ", "
    + f.tensionOutMin + ", " + f.tensionOutMax + ", "
    + f.intensiteOutMin + ", " + f.intensiteOutMax + ", "
    + ((f.crit[0].crit != null) ? f.crit[0].crit.idCritere :
CritereRecherche.valeurParDefisNull) + ", "
    + ((f.crit[0].signe != null) ? f.crit[0].signe.valeur :
CritereRecherche.valeurParDefisNull) + ", "
    + f.crit[0].valeur + " ) ";
    DataTable table = db.CreerDatatable(cmd);
}

```

```

foreach (DataRow row in table.Rows)
{
    FonctionElectronique fctElect = new FonctionElectronique();

    fctElect.projet = prjManager.getProjetById((string)row[0]);
    fctElect.fonction = fctGenData.getFonctionGenById((int)row[1]);
    fctElect.description = (string)row[2].ToString();
    fctElect.schema = (string)row[3].ToString();
    //les données io tension et intensité
    fctElect.tensionInputMin = float.Parse(row[4].ToString());
    fctElect.tensionInputMax = float.Parse(row[5].ToString());
    fctElect.intensiteInputMin = float.Parse(row[6].ToString());
    fctElect.intensiteInputMax = float.Parse(row[7].ToString());
    fctElect.tensionOutputMin = float.Parse(row[8].ToString());
    fctElect.tensionOutputMax = float.Parse(row[9].ToString());
    fctElect.intensiteOutputMin = float.Parse(row[10].ToString());
    fctElect.intensiteOutputMax = float.Parse(row[11].ToString());

    fctElect.cout = float.Parse(row[12].ToString());
    fctElect.validation = (Boolean)row[13];
    fctElect.lienSVNTest = (string)row[14].ToString();
    fctElect.enAttente = (Boolean)row[15];
    fctElect.taux = float.Parse(row[16].ToString());
    fctElect.listIO = ioPhysAssData.getListIoAssocieeByFonction(fctElect);
    fctElect.listCritere =
critAssData.getListCritAssocieeByFonction(fctElect);
    listFonction.Add(fctElect);
}
return listFonction;
}

/// <summary>
/// Obtenir une liste d'objet FonctionElectronique avec une close where
/// </summary>
/// <returns>
/// Une liste d'objet FonctionElectronique
/// </returns>
public List<FonctionElectronique> getListFonction(string closeWhere)
{
    List<FonctionElectronique> listFonction = new List<FonctionElectronique>();

    DbCommand cmd = db.CreerCommande();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "Select * from Fonction where " + closeWhere ;
    DataTable table = db.CreerDatatable(cmd);

    foreach (DataRow row in table.Rows)
    {
        FonctionElectronique fctElect = new FonctionElectronique();

        fctElect.projet = prjManager.getProjetById((string)row[0]);
        fctElect.fonction = fctGenData.getFonctionGenById((int)row[1]);
        fctElect.description = (string)row[2].ToString();
        fctElect.schema = (string)row[3].ToString();

        //les données io tension et intensité
        fctElect.tensionInputMin = float.Parse(row[4].ToString());
        fctElect.tensionInputMax = float.Parse(row[5].ToString());
        fctElect.intensiteInputMin = float.Parse(row[6].ToString());
        fctElect.intensiteInputMax = float.Parse(row[7].ToString());
        fctElect.tensionOutputMin = float.Parse(row[8].ToString());
        fctElect.tensionOutputMax = float.Parse(row[9].ToString());
        fctElect.intensiteOutputMin = float.Parse(row[10].ToString());
        fctElect.intensiteOutputMax = float.Parse(row[11].ToString());

        fctElect.cout = float.Parse(row[12].ToString());
        fctElect.validation = (Boolean)row[13];
        fctElect.lienSVNTest = (string)row[14].ToString();
        fctElect.enAttente = (Boolean)row[15];
    }
}

```

```

        //fctElect.listIO =
fctIoManager.getListIOAssocieeByFonction((string)row[0], (int)row[1]);
        //fctElect.listCritere =
fctCritManager.getListCritAssocieeByFonction((string)row[0], (int)row[1]);
        listFonction.Add(fctElect);
    }
    return listFonction;
}

/// <summary>
/// Modifier un <paramref name="fctElec"/> dans la table Fonction_Générique de la
base de données.
/// </summary>
/// <returns>
/// Nombre de lignes affectées
/// </returns>
/// <param name="fctElec">un objet FonctionGenerique</param>
public int modifierFonction(FonctionElectronique fctElec)
{
    DbCommand cmd = db.CreerCommande();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "UPDATE Fonction " +
                    "SET description = '" + fctElec.description + "', "
                    + "schéma = '" + fctElec.schema + "', "
                    + "tension_entree_min = " +
fctElec.tensionInputMin.ToString().Replace(",",".") + ", "
                    + "tension_entree_max = " +
fctElec.tensionInputMax.ToString().Replace(",",".") + ", "
                    + "intensite_entree_min = " +
fctElec.intensiteInputMin.ToString().Replace(",",".") + ", "
                    + "intensite_entree_max = " +
fctElec.intensiteInputMax.ToString().Replace(",",".") + ", "
                    + "tension_sortie_min = " +
fctElec.tensionOutputMin.ToString().Replace(",",".") + ", "
                    + "tension_sortie_max = " +
fctElec.tensionOutputMax.ToString().Replace(",",".") + ", "
                    + "intensite_sortie_min = " +
fctElec.intensiteOutputMin.ToString().Replace(",",".") + ", "
                    + "intensite_sortie_max = " +
fctElec.intensiteOutputMax.ToString().Replace(",",".") + ", "
                    + "cout=" + fctElec.cout.ToString().Replace(",",".") + ", "
                    + "validation=" + fctElec.validation + ", "
                    + "lien SVN_Dossier_PV='"+ fctElec.lienSVNTest + "'"
                    + "en_attente = " + fctElec.enAttente + " "
                    + "WHERE id_projet = '" + fctElec.projet.idProjet + "'"
and id_fonction=" + fctElec.fonction.idFonction;
    cmd.Parameters.Add(new MySqlParameter("@cout", MySqlDbType.Float, 8));
    cmd.Parameters["@cout"].Value = fctElec.cout;
    return db.ExecuterRequete(cmd);
}

/// <summary>
/// Supprimer un <paramref name="fctGen"/> dans la table Fonction_Générique de la
base de données.
/// </summary>
/// <returns>
/// Nombre de lignes affectées
/// </returns>
/// <param name="fctGen">un objet FonctionGenerique</param>
public int supprimerFonction(FonctionElectronique fctElec)
{
    DbCommand cmd = db.CreerCommande();
    cmd.CommandType = CommandType.Text;
    cmd.CommandText = "DELETE FROM Fonction " +
                    "WHERE id_projet = '" + fctElec.projet.idProjet + "' and "
id_fonction=" + fctElec.fonction.idFonction;
    return db.ExecuterRequete(cmd);
}
}

```

Annexe 7 – Code C# de la couche BAL pour la classe FonctionElectronique

```

using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using DAL;
using BEL;
using System.Data.Common;
using System.Data;

namespace BAL
{
    /*
     * //Création des méthodes CRUD pour la classe Personnel
     */
    /// <summary>
    /// <c>PersonnelData</c> class.
    /// Contient les méthodes CRUD de la classe personnel
    /// </summary>
    /// <remarks>
    /// <para>Cette classe peut ajouter, modifier et supprimer un objet personnel</para>
    /// <para>Cette classe permet d'obtenir la liste entière des personnel</para>
    /// </remarks>
    public class FonctionManager
    {
        private FonctionData fctd = new FonctionData();

        /// <summary>
        /// Ajouter un <paramref name="fctElec"/> dans la table Fonction de la base de données.
        /// </summary>
        /// <returns>
        /// Nombre de lignes affectées
        /// </returns>
        /// <param name="fctElec">un objet FonctionElectronique</param>
        public int ajouterFonction(FonctionElectronique fctElec)
        {
            //Ici les conditions pour exécuter ajouterFonction()
            return fctd.ajouterFonction(this.echappementFctElec(fctElec));
        }

        /// <summary>
        /// Obtenir un objet FonctionElectronique par son <paramref name="idFonction"/>
        /// </summary>
        /// <returns>
        /// Un objet FonctionElectronique
        /// </returns>
        /// <param name="idFonction">Identifiant de la fonction</param>
        public FonctionElectronique getFonctionGenById(int idFonction, string idprojet)
        {
            return fctd.getFonctionById(idFonction,idprojet.Replace("'", 
"").Replace(@"\", "\\")); 
        }

        /// <summary>
        /// Obtenir une liste d'objet FonctionElectronique
        /// </summary>
        /// <returns>
        /// Une liste d'objet FonctionElectronique
        /// </returns>
        public List<FonctionElectronique> getListFonction()
        {
            return fctd.getListFonction();
        }

        /// <summary>
    }
}

```

```

    ///> Obtenir une liste d'objet FonctionElectronique par son <paramref name="f"/>
    ///</summary>
    ///<returns>
    /// Une liste d'objet FonctionElectronique
    ///</returns>
    ///<param name="f">Filtre</param>
    public List<FonctionElectronique> getListFonction(Filtre f)
    {
        return fctd.getListFonction(f);
    }

    ///<summary>
    /// Obtenir une liste d'objet FonctionElectronique avec une close where
    ///</summary>
    ///<returns>
    /// Une liste d'objet FonctionElectronique
    ///</returns>
    public List<FonctionElectronique> getListFonction(string closeWhere)
    {
        return fctd.getListFonction(closeWhere);
    }

    ///<summary>
    /// Modifier un <paramref name="fctElec"/> dans la table Fonction_Générique de la
    base de données.
    ///</summary>
    ///<returns>
    /// Nombre de lignes affectées
    ///</returns>
    ///<param name="fctElec">un objet FonctionGenerique</param>
    public int modifierFonction(FonctionElectronique fctElec)
    {

        return fctd.modifierFonction(this.echappementFctElec(fctElec));
    }

    ///<summary>
    /// Supprimer un <paramref name="fctGen"/> dans la table Fonction_Générique de la
    base de données.
    ///</summary>
    ///<returns>
    /// Nombre de lignes affectées
    ///</returns>
    ///<param name="fctGen">un objet FonctionGenerique</param>
    public int supprimerFonction(FonctionElectronique fctElec)
    {
        return fctd.supprimerFonction(this.echappementFctElec(fctElec));
    }

//controler les caractères d'échappement pour l'insertion SQL
private FonctionElectronique echappementFctElec(FonctionElectronique fctElec)
{
    FonctionElectronique f = new FonctionElectronique();
    f = fctElec;
    f.projet.idProjet = fctElec.projet.idProjet.Replace("'", "''").Replace(@"\", "");
    f.description = fctElec.description.Replace("'", "''").Replace(@"\", " ");
    f.lienSVNTest = fctElec.lienSVNTest.Replace("'", "''").Replace(@"\", " ");
    return f;
}
}

```

Annexe 8 – Enchainement des fenêtres de l'IHM

