



RAPPORT DE FIN DE PROJET

Cédric ALLEMAND – Quentin BONICHOT

Omar LATRECHE – Taha MERRIKA

Encadré par M. BOUJEDLI Said et M. FOUGERES Alain Jérôme

Année universitaire 2014-2015

Table des matières

[I) Introduction 3](#_Toc423531293)

[II) Processus et gestion de projet : 4](#_Toc423531294)

[III) Définition des objectifs 5](#_Toc423531295)

[1) Objectifs du projet 5](#_Toc423531296)

[2) Analyse « MALIN » de l’objectif 5](#_Toc423531297)

[3) Liste des tâches à faire : 6](#_Toc423531298)

[4) Justification et Méthode : Phase Définition du projet 8](#_Toc423531299)

[IV) Analyse des besoins et faisabilité 9](#_Toc423531300)

[1) CQQCOQP 9](#_Toc423531301)

[2) Liste des contraintes 10](#_Toc423531302)

[3) Liste des exigences 11](#_Toc423531303)

[4) Etat de l'art 12](#_Toc423531304)

[5) Analyse Faisabilité 12](#_Toc423531305)

[6) Justification et Méthode : Phase d’analyse des besoins et faisabilité 13](#_Toc423531306)

[V) Conception générale : 14](#_Toc423531307)

[1) Analyse technique 14](#_Toc423531308)

[2) Analyse fonctionnelle : 15](#_Toc423531309)

[VI) Conception détaillée : 19](#_Toc423531310)

[1) Phase 1 : Dictionnaire des données 19](#_Toc423531311)

[2) Phase 2 : Données par groupe 20](#_Toc423531312)

[3) Phase final : Diagramme de Classe. 21](#_Toc423531313)

[4) Développement 22](#_Toc423531314)

[1. SADT 23](#_Toc423531315)

[2. Modèle PAC 24](#_Toc423531316)

[3. Matrice R.A.C.I (Responsable Acteur Consulté Informé) 25](#_Toc423531317)

[4. L’analyse SWOT 26](#_Toc423531318)

[5. Planning / Diagramme de Gantt 27](#_Toc423531319)

[6. Suivi et pilotage (Daily Scrum) 27](#_Toc423531320)

[5) Les tests 28](#_Toc423531321)

[1) Tests unitaires 28](#_Toc423531322)

[2) Tests d’intégration 29](#_Toc423531323)

[6) Recettes ou tests de validation 30](#_Toc423531324)

[7) Application finale 31](#_Toc423531325)

[1) Ecran de connexion 31](#_Toc423531326)

[2) Ecran principal 32](#_Toc423531327)

[3) Création du graphe 33](#_Toc423531328)

[4) Le calcul de valeurs 34](#_Toc423531329)

[8) Bilan fin de projet 36](#_Toc423531330)

[1) L'objectif initial est-il respecté ? 36](#_Toc423531331)

[2) Quels sont les points positifs de ce projet ? 36](#_Toc423531332)

[3) Quels sont les points à améliorer ? 36](#_Toc423531333)

[4) Respect du planning ? 36](#_Toc423531334)

[9) Remerciements 36](#_Toc423531335)

# Introduction

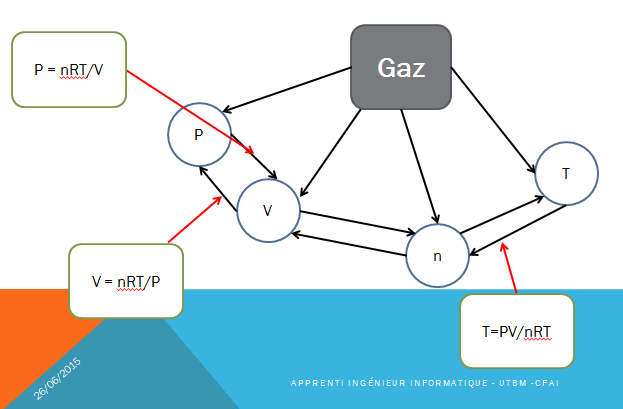
Dans le cadre de notre formation d’ingénieur en informatique et plus précisément durant les enseignements [LP74] - Interface Homme-Machine et [SI73] - Génie Logiciel, nous devions mener à bien un projet.

Dans le rôle du Client, M A.J. Fougères a spécifié le sujet et exprimé ses besoins, qu’ils soient d’un point de vue interface homme-machine ou fonctionnel. Ces besoins ont été repris ci-après. De plus, d’un point de vue gestion de projet et surtout méthodologie, M S. Boujedli nous a encadrés dans nos démarches analytiques, conceptuelles et de développement.

Dans ce rapport, vous retrouverez l’ensemble des phases de notre projet ainsi que les réflexions menées durant celle-ci. En fin du rapport, vous trouverez des écrans de l’application finale.

Aperçu du sujet de notre projet, j’ai nommé Modélisator :

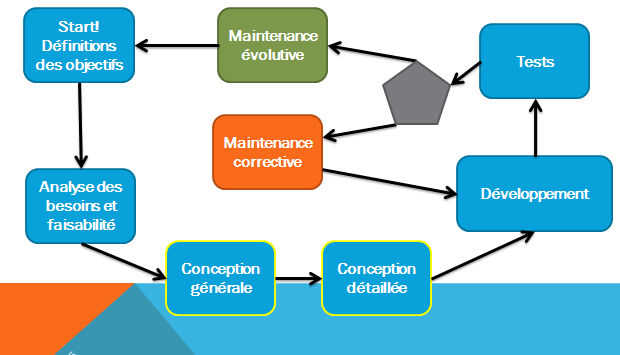
Nous souhaitons modéliser un objet physique (ici le gaz) à l’aide de ses grandeurs physiques. Elles seront représentées conceptuellement comme suit (exemple de grandeur physique de la formule PV=nRT).



On voit alors une représentation en graphe orienté ou les grandeurs représentées par des nœuds sont interconnectées par des arcs correspondants à leurs équations de dépendance.

# Processus et gestion de projet :

Dans le but de travailler de manière efficace et surtout de pouvoir réellement achever cet ouvrage en groupe, nous avons opté pour un fonctionnement en processus. Il s’inspire de méthodologies agiles telles que SCRUM ou le cycle en V. Voici donc notre cycle de vie de projet :



Chacune de ces phases est reprises dans la suite du rapport à l’exception des maintenances qui n’auront lieux qu’après livraison du projet, s’il poursuit son développement.

# Définition des objectifs

## Objectifs du projet

D’ici le 24/06/2015, réaliser une application/logiciel permettant de modéliser les propriétés physiques d’objets. La bibliothèque d’objets disponibles est limitée au contexte spécifié. La modélisation sera détaillée selon l’état d’avancement du projet.

## Analyse « MALIN » de l’objectif

Malin comme :

\* Mesurable (quels indicateurs chiffrés)

\* Atteignable (dans le sens réaliste)

\* Limité dans le temps (Quand)

\* Intelligent (il répond à une envie / problème)

\* Négociable ("gagnant/gagnant")

|  |  |  |
| --- | --- | --- |
| **Mesurable** :  Modelisator est mesurable par sa capacité à modéliser des objets de la liste spécifiée. | **Atteignable :**  Le projet Modelisator est réalisable de par la décomposition du niveau de modélisation d’objets partant du plus simple au plus détaillé. | **Limité** :  Le projet a pour date limite le 24/06/2015 |
| **Intelligent** :  Il répond aux besoins forts des entités à petit budget souhaitant modéliser leur catalogue. | **Négociable** :  Grâce au niveau détaillé progressif de la modélisation des objets, il est possible d’adapter l’objectif en réduisant le périmètre de modélisation des objets au profit de la qualité. |  |

## Liste des tâches à faire :

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **TODO list on 25/06/2015** | | | | | | | |
| **Mission** | **But** | **Objectif** | **Livrable** | **Référent** | **Date** | **Ressource** | **Etat** |
| **Définition des objectifs** | Objectifs |  | SMART/ MALIN | \* | 21/05/2015 | 2 |  |
| Choisir un nom |  | NOM | \* | 21/05/2015 |  |
| Choisir un Logo |  | Image du Logo | Q | 21/05/2015 | 0,5 |  |
| TO-DO list |  | ici | C + O | 21/05/2015 | 0,5 | C:\Users\Cedric\Pictures\taha.png |
| Documentation | Justification et méthode | Justification phase d'analyse | \* | 28/05/2015 | 0,5 | C:\Users\Cedric\Pictures\taha.png |
| **Analyse des besoins et faisabilité** | CQQCOQP | Analyse des besoins |  | Q + O | 28/05/2015 | 0,5 | C:\Users\Cedric\Pictures\taha.png |
| Liste des contraintes |  | Analyse contraintes | T | 28/05/2015 | 0,5 | C:\Users\Cedric\Pictures\taha.png |
| Liste des exigences |  | Analyse des requis | Q | 28/05/2015 | 0,5 | C:\Users\Cedric\Pictures\taha.png |
| Etat de l'art | Analyse de l'existant |  | T | 28/05/2015 | 0,5 | C:\Users\Cedric\Pictures\taha.png |
| Analyse Faisabilité | Faisabilité du projet | Analyse de faisabilité | C | 28/05/2015 | 0,5 | C:\Users\Cedric\Pictures\taha.png |
| CdC | Reprend toutes les analyses | CdC | Q |  | 0,5 | C:\Users\Cedric\Pictures\taha.png |
| Documentation | Justification et méthode | Justification phase de définitions | \* | 28/05/2015 | 0,5 | C:\Users\Cedric\Pictures\taha.png |
| **Conception** | Analyse Technique | Le choix technologique, la description des traitements, le nouveau planning | Rapport technique + spécifications | O |  | 1 | C:\Users\Cedric\Pictures\taha.png |
| Analyse Fonctionnelle | La description des fonctions; L’enchainement des écrans (avec un maquettage) | Rapport fonctionnel + maquettage écran | O + Q |  | 0,5 | C:\Users\Cedric\Pictures\taha.png |
| **Développement** | Planning |  | GANTT | T |  | 1 | C:\Users\Cedric\Pictures\taha.png |
| Responsabilités |  | RACI | T |  | 0,5 | C:\Users\Cedric\Pictures\taha.png |
| Gestion des risques |  | SWOT | C + Q |  | 1 | C:\Users\Cedric\Pictures\taha.png |
| Suivi et pilotage | Ensemble de réunions, méthode AGILE | Rapport réunion | O |  | 1,5 | C:\Users\Cedric\Pictures\taha.png |
| **Tests** | Prototype | Réaliser une version test du livrable souhaité | Prototype du logiciel | \* |  | 15 | C:\Users\Cedric\Pictures\taha.png |
| **Recette** | Finalisation du livrable | Tests d'intégration et tests unitaires | Cahier de test | C |  | 4 | C:\Users\Cedric\Pictures\taha.png |
| **Documentation** | Rapport fin de projet global | Analyse, objectif, planning, objectif, Ressources |  | \* |  | 1 | C:\Users\Cedric\Pictures\taha.png |
| Transfert du livrable | Remise du rapport de projet et prototype | Récépissé de livraison | \* |  | 2 | C:\Users\Cedric\Pictures\taha.png |
| **Accompagnement** | Formation MOA/Juge | Documentation pour la formation à l'utilisation du livrable | Documentation | \* |  | 3 |  |
| **Mise en production** | Réaffectation des ressources sur d'autres projets |  | Retour en entreprise | \* |  | 0,5 |  |
|  |  |  |  | **Total Jour/Homme:** | | **38** | 3 |
|  |  | légende | Omar | O | <10% |  | 0 |
|  |  | Quentin | Q | <50% |  | 1 |
|  |  | Cédric | C | >50% |  | 2 |
|  |  | Taha | T | 100% |  | 3 |
|  |  | Les 4 | \* |  |  |  |

## Justification et Méthode : Phase Définition du projet

Avant de commencer à étudier le projet en lui-même, nous avons jugé utile de définir certains points pratiques :

* Tout d’abord, nous avons eu besoin d’un outil de travail collaboratif pour gérer nos documents. Pour ce faire, notre choix s’est tourné vers Tortoise SVN, logiciel de versioning dont nous avions déjà une certaine expérience.
* Ensuite, nous avions besoin d’un bon cadre de travail, afin d’être plus efficaces. Pour cela, nous avons opté pour une salle de cours vide.
* Enfin, afin d’avoir une méthodologie qui nous permet de suivre correctement notre projet, nous avons choisi d’utiliser la méthode de cycle en V. L’avantage de cette méthode est que les différentes étapes sont bien définies et nous permettront de bien nous situer dans le projet au fur et à mesure de temps.

|  |  |  |
| --- | --- | --- |
| **Sujet** | **Comment** | **Pourquoi** |
| **Objectif** | Pour trouver notre objectif, nous avons jugé utile de nous mettre à la place du client. Pour cela, nous avons fait un jeu de rôles mettant en scène le client et le prestataire. | Pour définir ce que l’on recherche afin de parvenir à notre but. |
| **Choix du nom** | Pour choisir le nom, nous avons fait un brainstorming, qui consiste à écrire des noms que l’ensemble de l’équipe propose sur un tableau, puis de faire un choix parmi les propositions. | Le nom sert à donner une identité à un logiciel. |
| **Choix du logo** | Nous avons choisi délibérément un logo qui possède des formes plutôt abstraites, car le logiciel pourra à terme modéliser un nombre important d’objets différents, quel qu’ils soient. | Le logo sert à compléter le nom en lui donnant une identité visuelle supplémentaire. En effet, dans de nombreuses situations, on reconnaît le logo plus que le nom ou le logiciel en lui-même. |
| **La to-do list** | En s’inspirant des différentes phases d’un cycle de vie de projet. On recense toute les tâches et livrable à réaliser. | Permet de définir toutes les tâches dans le processus de création d’un projet. |

# Analyse des besoins et faisabilité

## CQQCOQP

* Quoi ?

Modelisator est un logiciel qui permet la conception d’objets et leurs propriétés physiques.

* Pourquoi ?

Modelisator répond au besoin de concepteurs et de modélisateurs recherchant un logiciel de modélisation d’objets selon un contexte, tels que des bonbonnes de gaz, ainsi que leurs propriétés physiques (pression, volume, etc.).

* Qui ?

Le logiciel est créé par des apprentis-ingénieurs en informatique. Il est destiné à des concepteurs/modélisateurs.

* Quand ?

Le projet compte cinq semaines de travail : il a commencé le 21 mai 2015 et a pour date de fin le 1er juillet 2015.

* Combien ?

L’équipe de travail est composée de quatre apprentis-ingénieurs informatiques ayant chacun leur ordinateur personnel.

* Où ?

Le déroulement du projet se fait dans les salles de classes disponibles de l’UTBM. Le déploiement de l’application sera fait sur un site en ligne de distribution gratuit.

* Comment ?

A l’aide d’une interface graphique intuitive enrichie par une banque de données mettant en relation l’ensemble des grandeurs physiques pour un objet dans un contexte. L’utilisateur peut alors explorer l’ensemble des propriétés physiques de l’objet.

## Liste des contraintes

Contraintes Projet :

* Temporelle :

Le livrable est attendu dans un délai assez court (6 semaines) vu l’importance du projet. Les ressources humaines ne sont pas disponibles à 100% durant ces périodes.

* Ressources humaines :

Le nombre de développeur est limité à 4. Le projet ne dispose pas de testeurs externes.

* Ressources matérielles :

Nous disposons juste de notre propres machines pour pouvoir développer. Le projet n’est pas financé.

* Légal :

Le logiciel Modelisator est sous licence MIT, toutes les technologies utilisées doivent être de licence équivalente ou moins restrictive.

Contraintes Métier :

* Qualité :

Le logiciel doit être performant et répondre exactement aux besoins. Il est soumis au respect des lois physiques universelles.

* Evolutif :

Modelisator permet de modéliser un contexte précis, mais on doit laisser la possibilité de pouvoir modéliser d’autres contextes.

## Liste des exigences

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |
|  | **Liste des Exigences 28/05/2015** | | | | |  |
|  | **Type** | **Nom** | **Résultat attendu** | **Ecart autorisé** | **Etat du test** |  |
|  | **Exigences fonctionnelles** | Pouvoir modéliser un objet physique en fonction de ses GP | Objet physique | Toutes les GP du contexte modélisables |  |  |
|  | Pouvoir renseigner une valeur de GP non calculée | Paramètre d'objet | Toutes les valeurs saisies sont modifiables |  |  |
|  | Visualiser toutes les GP liées directement à une GP | Graphe des GP | n/a |  |  |
|  | Calculer une GP à partir d'autres GP | Résultats de calculs mathématiques | Toutes les GP calculables depuis les GP saisies |  |  |
|  | Application compatible avec plusieurs systèmes | Le logiciel doit pouvoir être installé sur toutes plateformes Windows 7 ou antérieur | Minimum 2 OS |  |  |
|  | Application rapide d'installation | Le logiciel doit pouvoir s'installer en moins de cinq minutes | +/- 10% minutes |  |  |
|  | Performance de l'application |  |  |  |  |
|  | **Exigences non-fonctionnelles** | Ergonomie | Faire apparaître le logo dans chacune des fenêtres du logiciel | +/- 20% du nb de fenêtres |  |  |
|  | Facile à maintenir | Documentation suffisante pour un développeur logiciel | n/a |  |  |
|  |  |  |  |  |  |  |
|  | **GP** | Grandeur physique |  |  |  |  |
|  | **OS** | Système d'exploitation (Operating System) |  |  |  |  |

## Etat de l'art

|  |  |  |
| --- | --- | --- |
| **Logiciel existant** | **Similitude** | **Manquement** |
| **Calculs elec** | Contient une bibliothèque des formules électrique et les manipulent pour effectuer des calculs | Ne permet pas la modélisation et l’interprétation. |
| **ScieLab, Matlab** | Contient une bibliothèque des formules mathématiques et les manipulent pour effectuer des calculs.  Ils peuvent aussi réaliser des modèles mathématiques. | Il n’y a pas de liaison automatique entre les formules.  Pas d’aspects graphiques directs. |

## Analyse Faisabilité

|  |  |
| --- | --- |
|  | |
| **Technique (comparé à l’état de l’art)** | La faisabilité technique dépend du niveau de détail apporté à la modélisation qu’on choisira d’atteindre dans ce projet. En partant sur une modélisation basique (et non pas avancée telle qu’elle est dans Matlab par exemple), alors l’objectif est atteignable. |
| **Budgétaire** | Aucune dépense n’est prévue lors du projet. Nous disposons chacun d’une machine et nous possédons également une licence MSDN nous donnant accès à des outils de développement. |
| **Ressources humaines**  **(temps)** | Nous sommes 4 développeurs et nous disposons d’un temps de travail dans le cadre de l’UV SI73 et de l’UV LP74. Ce temps de travail représente environ une cinquantaine d’heure par personne, soit 28 jours/homme.  En dehors de ce cadre, les développeurs ne sont pas disponibles à 100%. |

## Justification et Méthode : Phase d’analyse des besoins et faisabilité

|  |  |  |
| --- | --- | --- |
| **Sujet** | **Comment** | **Pourquoi** |
| **Analyse poussée de l’objectif** | A l’aide des questions d’Aristote (QQOQCPC). | L’analyse poussée de l’objectif permet de faire le tour de toutes les questions qui doivent être posées avant le début du projet pour de futures analyses. |
| **Liste des contraintes** | En recensant l’ensemble des contraintes du projet et en les classant par typologie (métier, temporelle, légale, …) | Pour délimiter notre marge de manœuvre et d’orienter les différentes décisions, qu’elles soient techniques ou fonctionnelles. |
| **Liste des exigences** | En recensant l’ensemble des exigences du projet qu’elles soient fonctionnelles ou non. Chaque exigence est spécifiée par un résultat attendu et un écart autorisé. | Pour réaliser une application conforme aux exigences du client, il faut que cette liste soit créée avant le début du développement. Elle sert de base au cahier de recettes. |
| **Etat de l’art** | Après avoir identifié les différentes fonctionnalités attendues de Modelisator, nous avons cherché des logiciels similaires sur des plateformes de distribution logicielle (web). | Permet de se renseigner sur l’existant (concurrence éventuelle) ainsi que de profiter des retours d’expérience des précédents développements. On s’assure de ne pas réinventer la roue. |
| **Analyse de faisabilité** | Pour réaliser cette analyse, nous nous sommes basés sur trois composantes : le niveau de technicité apporté, le budget alloué, et les ressources déployées. | Etude menée afin de savoir si le projet est réalisable. Cela se fait en étudiant le délai et les ressources disponibles. |

# Conception générale :

## Analyse technique

* 1. **Choix technologique**

**Le langage de programmation C#**

Le langage de programmation choisi pour le développement de l’application était le C#. Ce choix était basé sur la simplicité de ce langage et la fonctionnalité qu’il peut offrir à notre application, par rapport aux autres langages de programmation. De plus, deux membres de l’équipe ont une expérience sur ce langage de programmation.

**Microsoft Visual Studio**

Microsoft Visual Studio est un ensemble d'[environnements de développement intégrés](https://fr.wikipedia.org/wiki/Environnement_de_d%C3%A9veloppement_int%C3%A9gr%C3%A9) développé par [Microsoft](https://fr.wikipedia.org/wiki/Microsoft). C’est l’outil de programmation le mieux adapté au C# vu les fonctionnalités qu’il contient. Il nous est accessible grâce aux licences UTBM.

**Windows Presentation Foundation (WPF)**

C’est un système de présentation nouvelle génération qui génère des applications clientes Windows. Cette technologie nous a particulièrement été utile dans le cadre de création et gestion de toutes les interfaces graphiques de l’application.

**Modèle-Vue-VueModèle (MVVM)**

C’est l’architecture et la méthode de conception utilisée car c’est la mieux adaptée pour les applications basées sur les technologies WPF, il permet de séparer entre les 3 couches : le modèle, la vue et la vue-Modèle.

* 1. **Description des traitements**

Comme nous l’avons défini précédemment, Modelisator permet de calculer des grandeurs physiques d’un objet ; ce calcul passe par plusieurs traitements.

Le premier traitement consiste à recenser la liste des objets ainsi que la liste des grandeurs physiques associées à chaque objet. Par la suite, il faut représenter sous forme de graphe l’objet relié à ses premières grandeurs physiques.

Ensuite, il faut représenter les relations entre les grandeurs physiques à partir de la liste des équations. Puis, il faut calculer automatiquement les grandeurs physiques vides en fonction des grandeurs physiques saisies.

Finalement, il faut afficher les valeurs calculées. Ainsi, la modification d’une grandeur physique implique de recalculer les grandeurs physiques qui sont en relation.

## Analyse fonctionnelle :

* 1. **Description des fonctions**

**Connexion :**

Il y a plusieurs types d’utilisateurs et chaque type d’utilisateur a un traitement spécifique à faire, mais nous avons décidé de ne pas implémenter ces différents modes d’utilisateurs par soucis de temps.

**Afficher les grandeurs physiques d’un objet :**

Après s’être connecté, nous pouvons apercevoir la liste des objets. Pour visualiser les grandeurs physiques d’un objet, il suffit de faire un glisser-déposer de ce dernier vers l’espace de travail. A ce stade-là, on n’affiche pas les relations entre les grandeurs physiques.

**Afficher les relations d’une grandeur physique :**

En cliquant sur une grandeur physique, on passe à un autre niveau d’affichage, qui permet d’afficher les grandeurs physiques en interdépendance avec celle-ci.

**Calcul des valeurs des grandeurs physiques :**

En saisissant la valeur d’une grandeur physique, l’application peut calculer la valeur des grandeurs physiques dont tous les paramètres de l’équation sont remplis. Les valeurs calculées ne sont pas modifiables, par contre on peut toujours modifier les grandeurs physiques saisies.

**Enregistrer l’historique :**

L’enregistrement de l’historique permettra d’implémenter les fonctions « précédent » et « suivant », qui consiste à mémoriser les étapes dès l’ouverture de l’application jusqu’à sa fermeture.

**Enregistrer la saisie :**

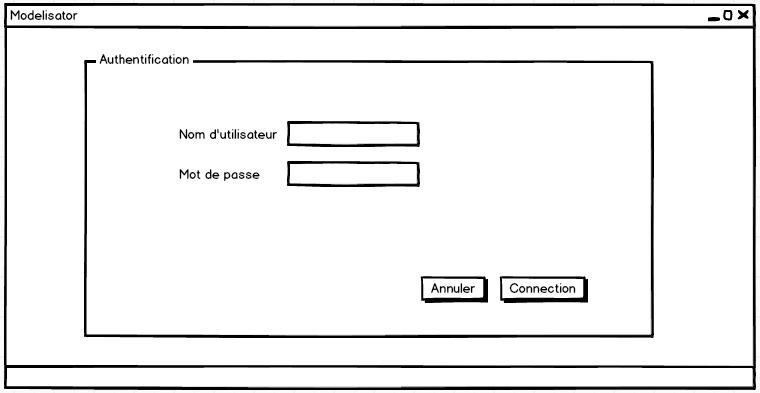
Cette fonctionnalité permet de garder la saisie des valeurs sous forme d’un projet, et on peut ouvrir le projet dans l’état que nous l’avions laissé.

* 1. **Maquettage**

Avant de passer aux étapes suivantes dans la conception du projet, il a été nécessaire de concevoir la première version des écrans sous forme de maquettes, dont il faut les respecter pendant la réalisation des écrans. Ces maquettes représentent la base et la source des IHM de l’application.

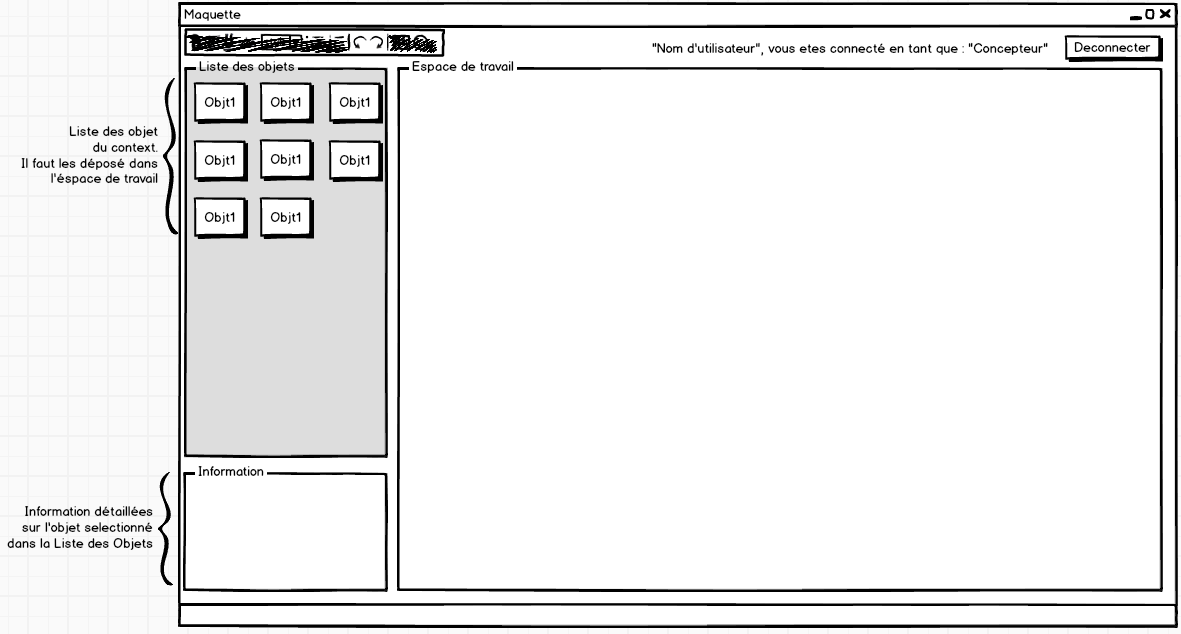
**Connexion**

C’est l’écran qui apparaît à l’ouverture de l’application.



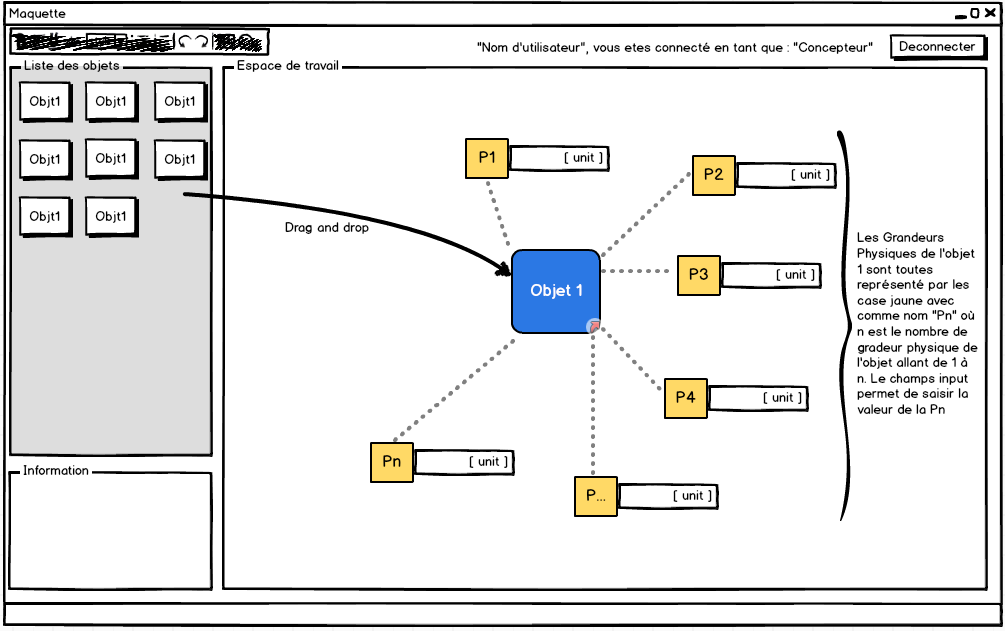
**Premier écran après connexion : affichage de la liste des objets**

Après s’être connecté, on affiche des informations sur l’utilisateur connecté, la liste des objets, l’espace de travail vide et une bulle d’information. L’affichage est voulu organisé, afin de ne pas se perdre dans des multitudes de fenêtres. Ainsi, la liste des objets est à gauche, l’espace de travail à droite, et en bas se trouvent les informations annexes.



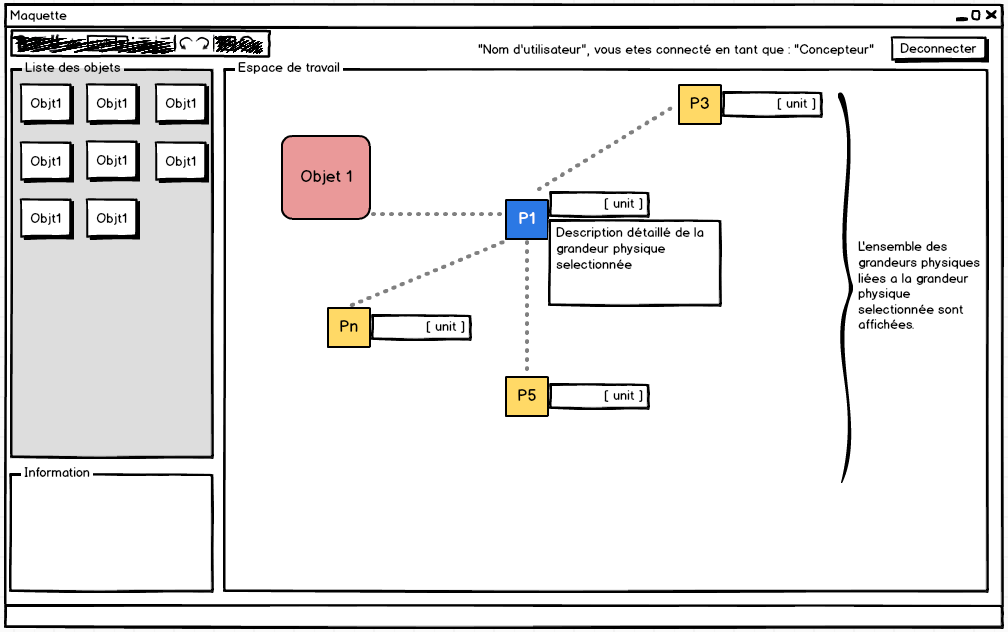
**Glisser-déposer un objet**

Les grandeurs physiques d’un objet sont affichées après un glisser-déposer de l’objet en question. Cela permet une préhension plus facile des différents composants affichés à l’écran et une maniabilité aisée.



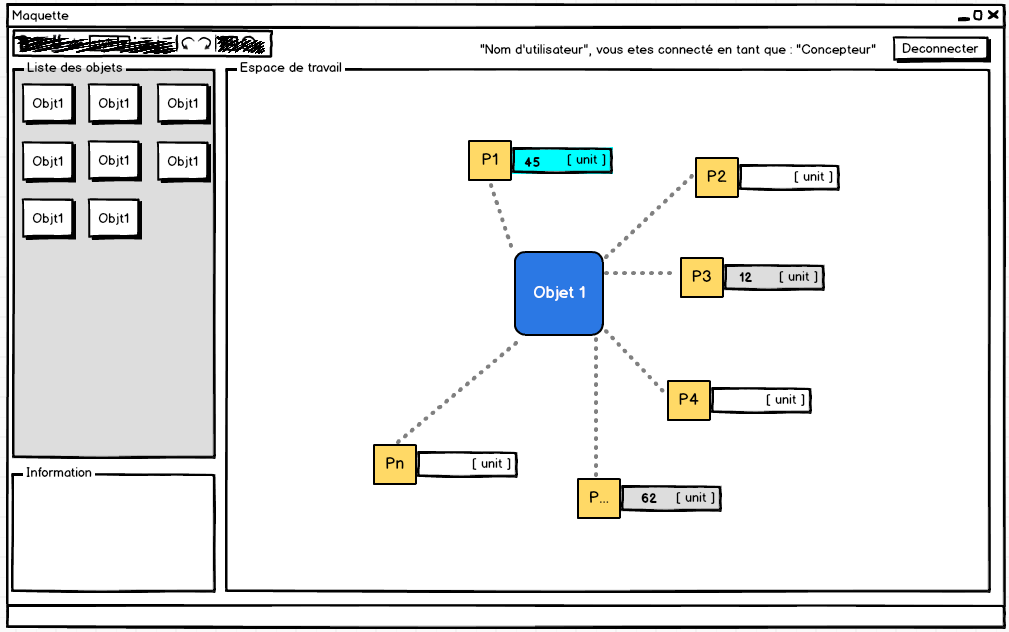
**Choix d’une grandeur physique**

Le clic sur une grandeur physique permet d’afficher toutes les autres grandeurs physiques qui sont en relation avec elle. La valeur et une description brève de la grandeur apparaissent.



**Saisie d’une grandeur physique**

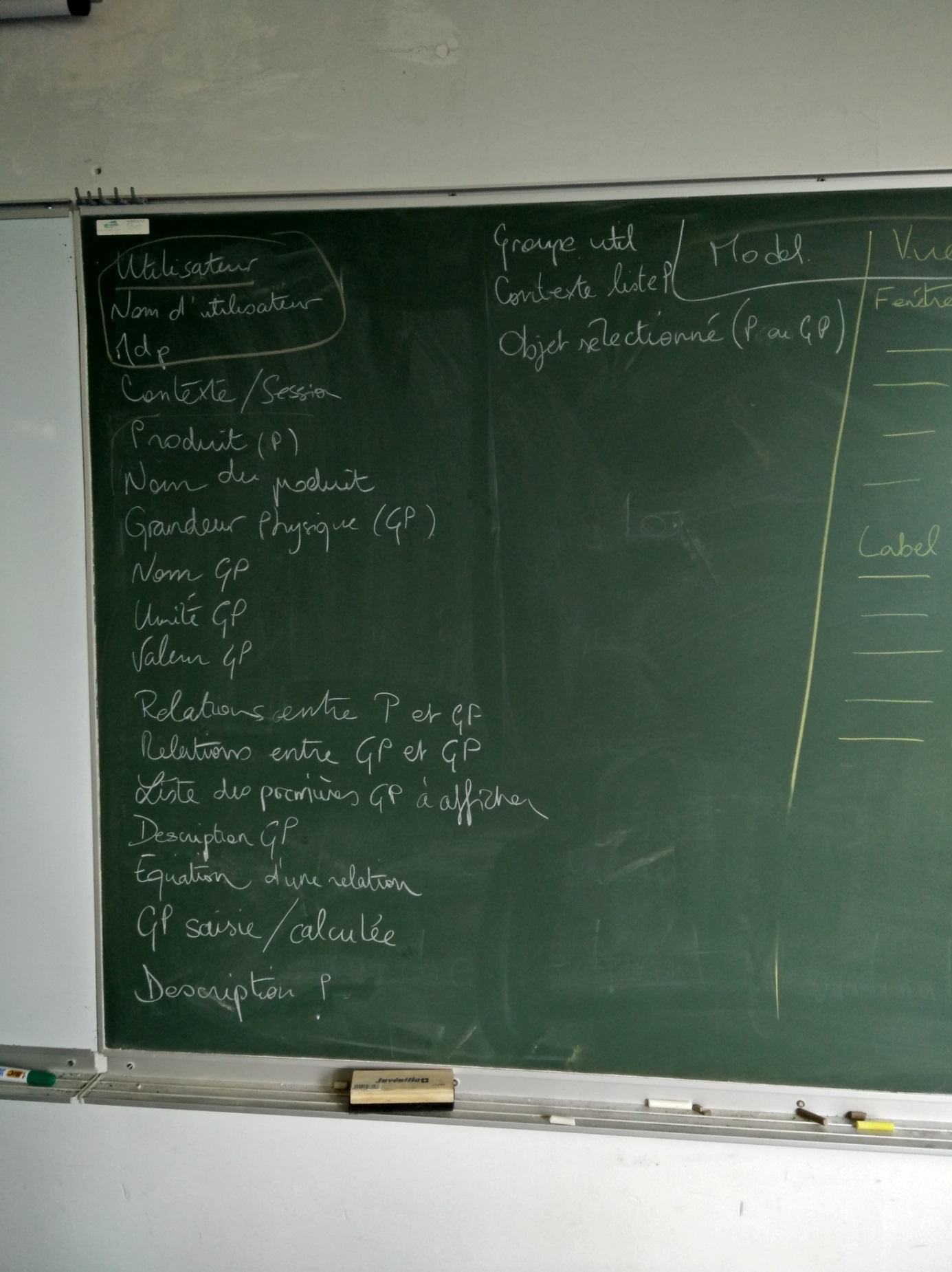
Les grandeurs physiques saisies sont modifiables, et les grandeurs physiques calculées sont grisées.



# Conception détaillée :

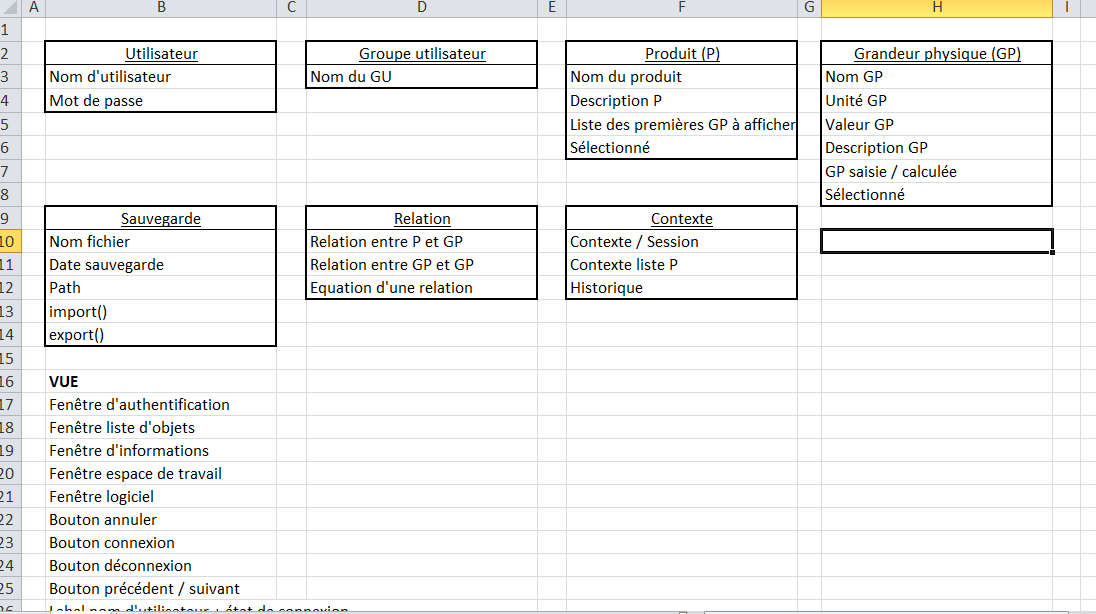
## Phase 1 : Dictionnaire des données

Dans cette phase, nous cherchons à définir le squelette de l’application, c’est pourquoi nous commençons par recenser l’ensemble de toutes les informations que contient l’application.



## Phase 2 : Données par groupe

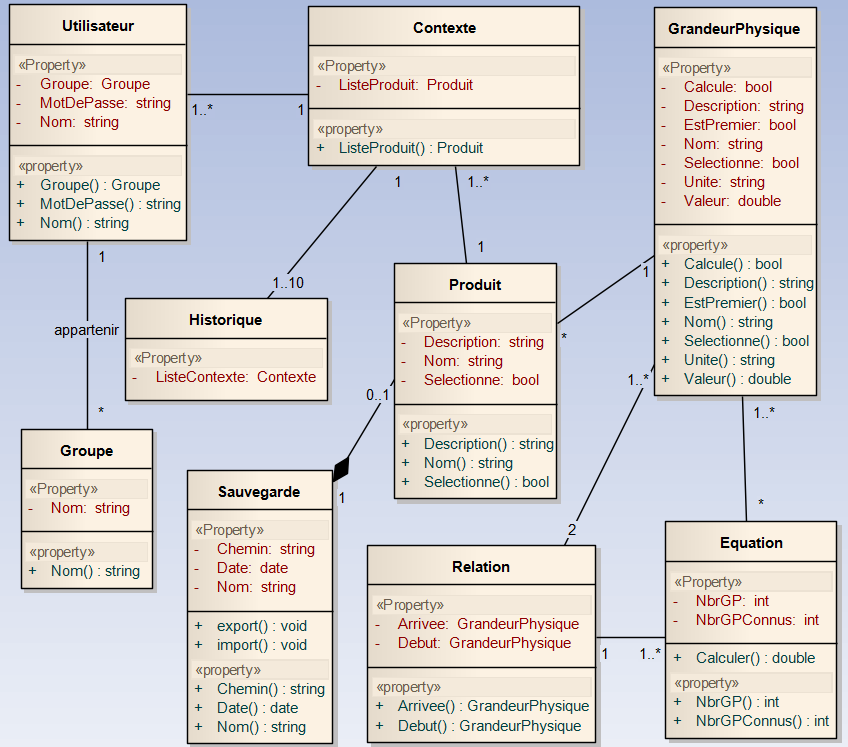
Ensuite, afin de pouvoir modéliser en diagrammes UML les données nous les avons au préalable triées par groupe.



Ce premier tri est déjà un niveau analytique du diagramme de classe UML.

## Phase final : Diagramme de Classe.

Voici donc le diagramme de classe qui conclut la conception. Réalisé depuis le logiciel Architect Entreprise, notre choix c’est pencher vers cet outil pour sa capacité à générer les classes modélisées dans le diagramme.



Ce modèle a servi de socle pour la construction de l’architecture, malgré les générations de code incomplète du logiciel AE, nous sommes restés fidèle au maximum aux conceptions précédentes.

# Développement

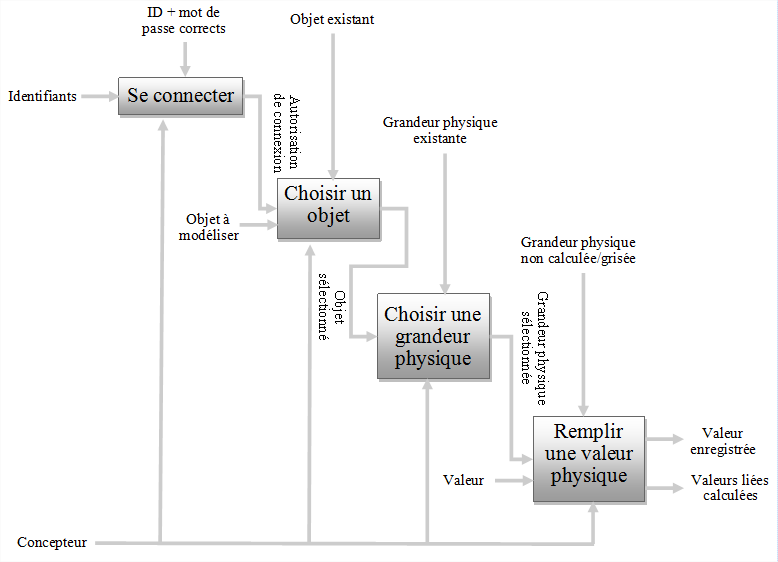
Avant de commencer dans le développement, on a découpé le projet en un ensemble de modules, pour pouvoir se répartir les tâches. Comme on a choisi d’utiliser la méthode MVVM, le projet se décompose sur 3 principaux modules : le modèle, la vue, et la vue-modèle. La vue concerne tout ce qui est IHM et interface graphique pour interagir avec l’utilisateur. Le modèle concerne tout ce qui est fonctionnalités de notre application. Et la vue-modèle permet de faire le lien entre l’IHM et les données du modèle.

Après avoir découpé le projet en module, on a réparti les tâches comme suit :

* Vue : Quentin et Omar
* Modèle : Taha et Cédric
* Vue-modèle : tous les membres de l’équipe.
* Architecture de l’application : Omar

## SADT

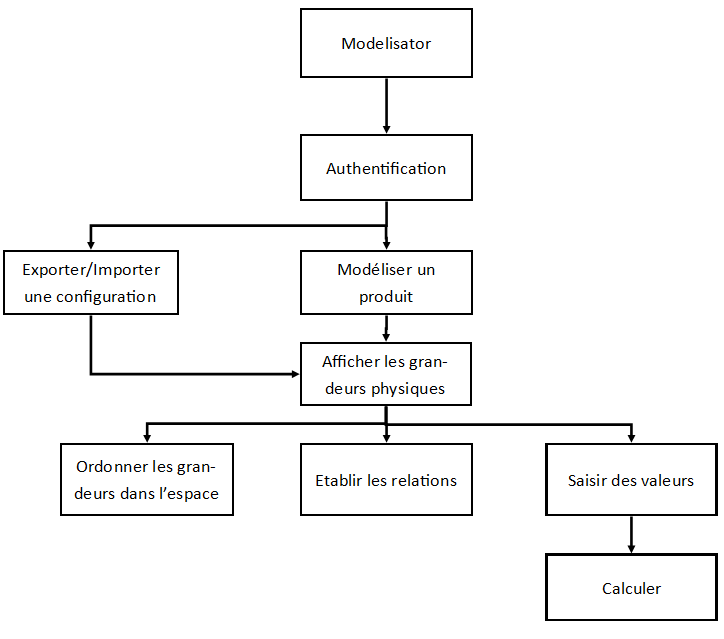
Afin de regrouper toutes nos idées en une seule, nous avons créé un diagramme SADT (*Structured Analysis and Design Technique*). Ce diagramme nous permet de garder les principales fonctions et le déroulement de l’application en tête. Nous avons fait ce diagramme surtout pour la modélisation de l’interface, car nous avons souhaité mettre en avant le côté « utilisation du programme ».



Le diagramme est volontairement simple, car il nous permettait de simplement mettre en place les bases de l’application, c’est-à-dire les principales utilisations de celle-ci. De plus, faire toutes les exécutions possibles sur un seul graphe aurait été illisible.

## Arbre des tâches

Dans la même optique que le diagramme SADT, nous avons utilisé un arbre des tâches afin de bien identifier les rôles, les fonctions et les différents liens entre tous les composants de l’application.



## Matrice R.A.C.I (Responsable Acteur Consulté Informé)

Pour donner plus de sens à notre ToDo List, nous avons utilisé la matrice R.A.C.I. Cette matrice reprend la ToDo List vue précédemment en y ajoutant les responsabilités de chacun des membres de l’équipe projet.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Taches** | | **Equipe et responsabilités** | | | | | |
| Cédric | Omar | | | Quentin | Taha |
| Définir l’objectif | | R | R | | | R | R |
| Analyse des besoins et faisabilité | | R | R | | | R | R |
| A | A | | | A | A |
| C | C | | | C | C |
| I | I | | | I | I |
| Conception générale | | I | A | | | R | C |
| Conception détaillée | | I | A | | | R | C |
| Développement | | R | R | | | R | R |
| C | C | | | C | C |
| Test | | R | R | | | R | R |
| C | C | | | C | C |
| Recette | | A | C | | | I | R |
| Documentation | | C | A | | | R | I |
| Accompagnement | | R | I | | | C | A |
| Mise en production | | A | R | | | I | C |
| Assistance et maintenance | | A | C | | | R | I |
|  | |  |  | | |  |  |
| **Légende** | | | | | | | | |
| **R** | Responsable de la réalisation | | | **C** | Doit être consulté | | | |
| **A** | Approbation finale pour l'autorité | | | **I** | Doit être informé | | | |

L’équipe projet étant composée de quatre développeurs, les responsabilités de chacun des membres de l’équipe sont communes sur certaines tâches.

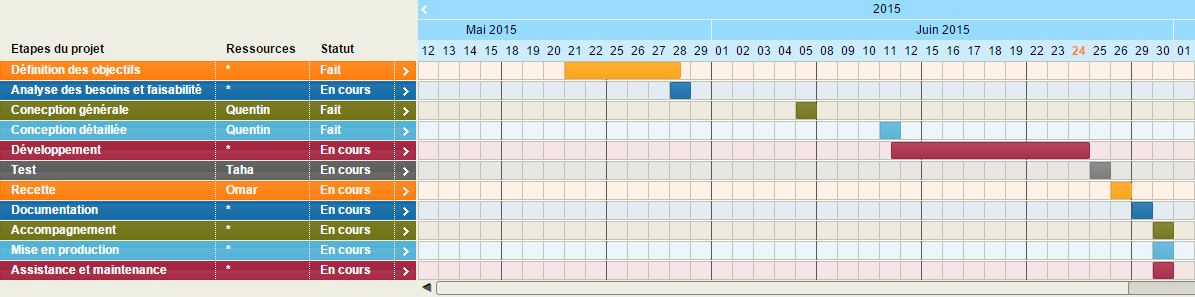
## L’analyse SWOT

L’analyse SWOT ci-dessous nous a permis de définir correctement les options stratégiques de notre projet. Nous avons ainsi cherché à savoir quelles étaient les points forts et les points faibles de notre projet, qu’ils soient internes au développement ou externes.

|  |  |
| --- | --- |
| **Forces** | **Faiblesses** |
| Maîtrise de C# par la plupart des personnes de l'équipe Ordinateurs personnels et équipements de l'UTBM à disposition | Non-maîtrise de WPF par les développeurs |
| **Opportunités** | **Menaces** |
| Pas de concurrence Aide du professeur lors de situations ou de décisions délicates | Se faire une place parmi des logiciels similaires plus connus (Matlab par ex.) Manque de temps cruel pour faire un projet totalement abouti |

## Planning / Diagramme de Gantt

Afin de visualiser dans le temps les grandes tâches du projet, le diagramme de Gantt est une méthode très efficace réalisant un planning. On peut éventuellement faire un diagramme de Gantt plus détaillé pour préciser chacune des tâches, suivant l’avancement dans le projet.



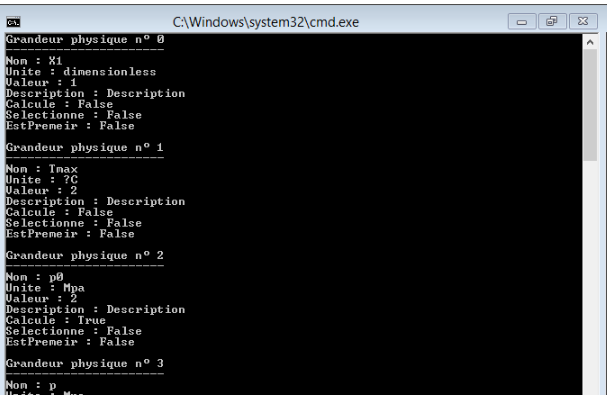
## Suivi et pilotage (Daily Scrum)

A chaque nouvelle réunion (dans notre cas il s’agissait de chaque nouvelle séance de cours de SI73 ou LP74), nous nous sommes réunis et avons pris à tour de rôle la parole afin de définir le travail qui a été fait, ce qui reste à faire et les difficultés éventuelles rencontrées depuis la dernière réunion. Un secrétaire était désigné afin de prendre note de ce qui était dit. Une fois que tout le monde avait parlé, on pouvait ensuite définir les nouvelles tâches à venir, ou les difficultés à résoudre.

# Les tests

## Tests unitaires

Un ensemble de tests a été réalisé sur l’ensemble des fonctionnalités de l’application dès leurs développement, à l’exemple de ce test unitaire sur la fonction d’import – export d’un espace de travail :



## Tests d’intégration

Les tests d’intégrations sont indispensables. Ils ont permis de vérifier le bon fonctionnement des différentes fonctionnalités entre elles.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Testes d'intégration** | | | | | |
| **Module** | **Fichier** | **Résultat attendu** | **Ecart autorisé** | **Etat du test** | **T/D** |
| Module authentification | Auth\_View\_UserControl.xaml | Fonctionnel | aucun | **OK** | Taha/ Quentin |
| Auth\_ViewModel.cs | Fiable | pas de base users | **OK** | Omar/ Omar |
| Auth\_Model.cs | Accessible | système de groupe de paille | **OK** | Omar/ Omar |
| Espace de travail | EspaceTravail\_View\_UserControl.xaml | fonctionnel | 1 seul objet | **OK** | Taha/ Cedric |
| EspaceTravail\_ViewModel.cs | Fiable | visualisation de GP partiel | **OK** | Taha/ Quentin |
| EspaceTravail\_Model.cs | Accessible | sauvegarde optionnelle | **OK** | Taha/ Cedric |
| Menu et info-utilisateur | Info\_View\_UserControl.xaml MenuTop\_View\_UserControl | fonctionnel | Visible | **OK** | Cedric / Quentin |
| Info\_ViewModel.cs MenuTop\_ViewModel.cs | Fiable | Import-export | **OK** | Cedric/ Quentin |
| Info\_Model.cs MenuTop\_View.cs | Accessible | - | **OK** | Cedric/ Quentin |
| Module structure | ModelisatorFram\_View\_UserControl.xaml | fonctionnel | fixe ou responsive | **Reserve** | Omar/ Taha |
| ModelisatorFrame\_ViewModel.cs | Fiable | centralise les contrôles | **OK** | Quentin/ Omar |
| ModelisatorFrame\_Model.cs | Accessible | - | **OK** | Taha/ Omar |
| Module démarrage | MainController.cs | Cerveau | - | **OK** | Cedric Omar |
|  |  |  |  |  |  |
| T/D: | Nom du testeur/Nom du développeur de la partie testé |  |  |  |  |

# Recettes ou tests de validation

Le cahier des recettes ou tests de validation s’effectuent avec M. Fougères pour valider ou non la conformité et le bon fonctionnement de l’application selon ses exigences au préalable recensés.

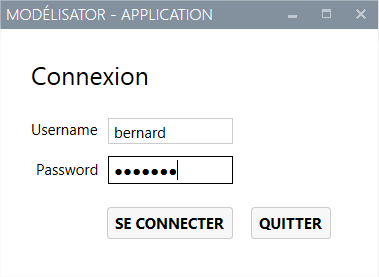
|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Type** | **Nom** | **Résultat attendu** | **Ecart autorisé** | **Etat du test** |  |
|  | **Exigences fonctionnelles** | Pouvoir modéliser un objet physique en fonction de ses GP | Objet physique | Toutes les GP du contexte modélisables | OK |  |
|  | Pouvoir renseigner une valeur de GP non calculée | Paramètre d'objet | Toutes les valeurs saisies sont modifiables | OK |  |
|  | Visualiser toutes les GP liées directement à une GP | Graphe des GP | n/a | OK |  |
|  | Calculer une GP à partir d'autres GP | Résultats de calculs mathématiques | Toutes les GP calculables depuis les GP saisies | OK |  |
|  | Application compatible avec plusieurs systèmes | Le logiciel doit pouvoir être installé sur toutes plateformes Windows 7 ou antérieur | Minimum 2 OS | OK |  |
|  | Application rapide d'installation | Le logiciel doit pouvoir s'installer en moins de cinq minutes | +/- 10% minutes | A tester |  |
|  | Performance de l'application |  |  |  |  |
|  | **Exigences non-fonctionnelles** | Ergonomie | Faire apparaître le logo dans chacune des fenêtres du logiciel | +/- 20% du nb de fenêtres | Sur écran de démarrage |  |
|  | Facile à maintenir | Documentation suffisante pour un développeur logiciel | n/a | OK |  |
|  |  |  |  |  |  |  |
|  | **GP** | Grandeur physique |  |  |  |  |
|  | **OS** | Système d'exploitation (Operating System) |  |  |  |  |

# Application finale

Pour présenter l’application, prenons l’exemple de Bernard, qui souhaite utiliser l’application.

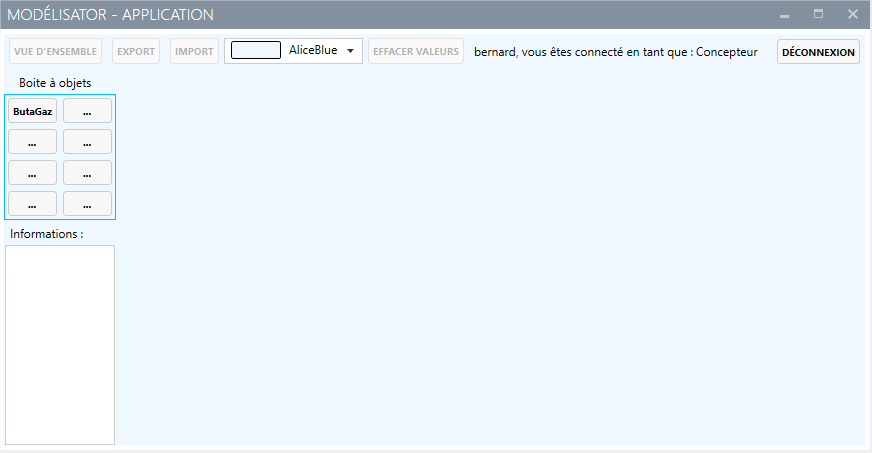
## 1) Ecran de connexion

Bernard possède les identifiants « bernard » et « bernard ». Il va donc les insérer sur l’écran de connexion ci-dessous :



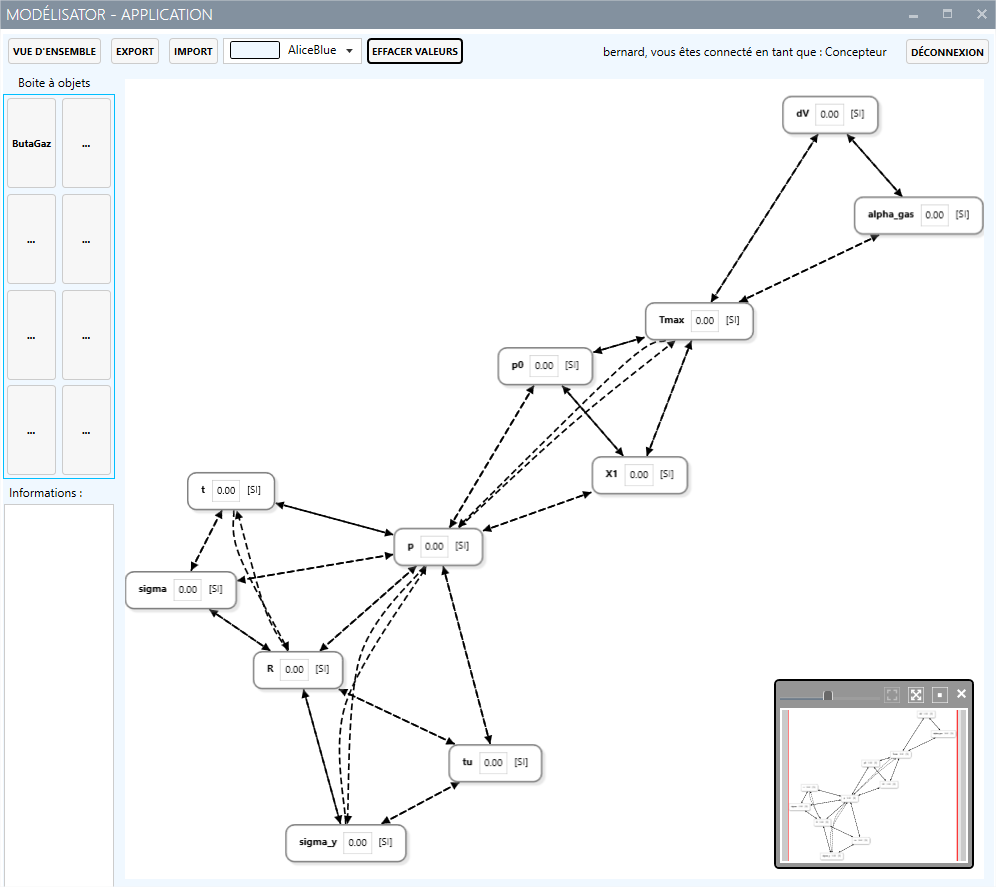
## 2) Ecran principal

Les identifiants de Bernard étant reconnus, on passe à l’écran principal de Modelisator. Cet écran regroupe les différentes fonctions disponibles, mais n’affiche aucun graphe. Ce qui est logique, puisque Bernard n’a pas encore choisi d’objet à étudier. Dans le même temps, quelques fonctions sont désactivées, car elles sont relatives au graphe. On observe également une ligne d’état de connexion, qui indique que Bernard est connecté en tant que Concepteur. Enfin, plutôt que de forcer une couleur d’affichage, nous proposons à Bernard de la changer par celle qu’il souhaite dans la liste déroulante.



## 3) Création du graphe

Bernard voudrait maintenant étudier l’objet « Butagaz ». Il clique donc sur cet objet dans la liste qui se trouve à sa disposition sur la gauche. C’est le déclencheur qui fait apparaître le graphe des grandeurs.

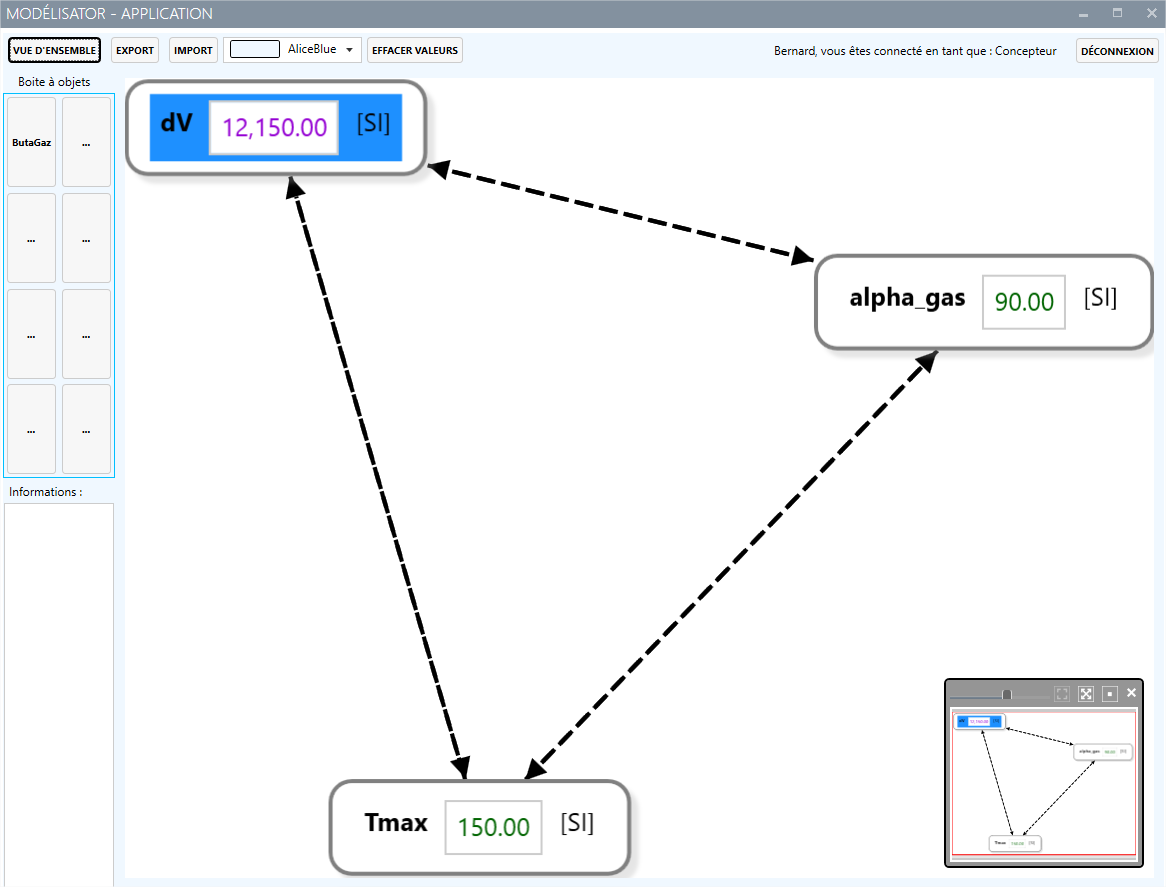


Toutes les grandeurs qui sont affichées à l’écran sont celles associées à cet objet. Elles sont liées par des flèches en pointillés (pleine lorsque la flèche va dans les deux sens), ce qui signifie qu’une équation mathématique relie les deux grandeurs.

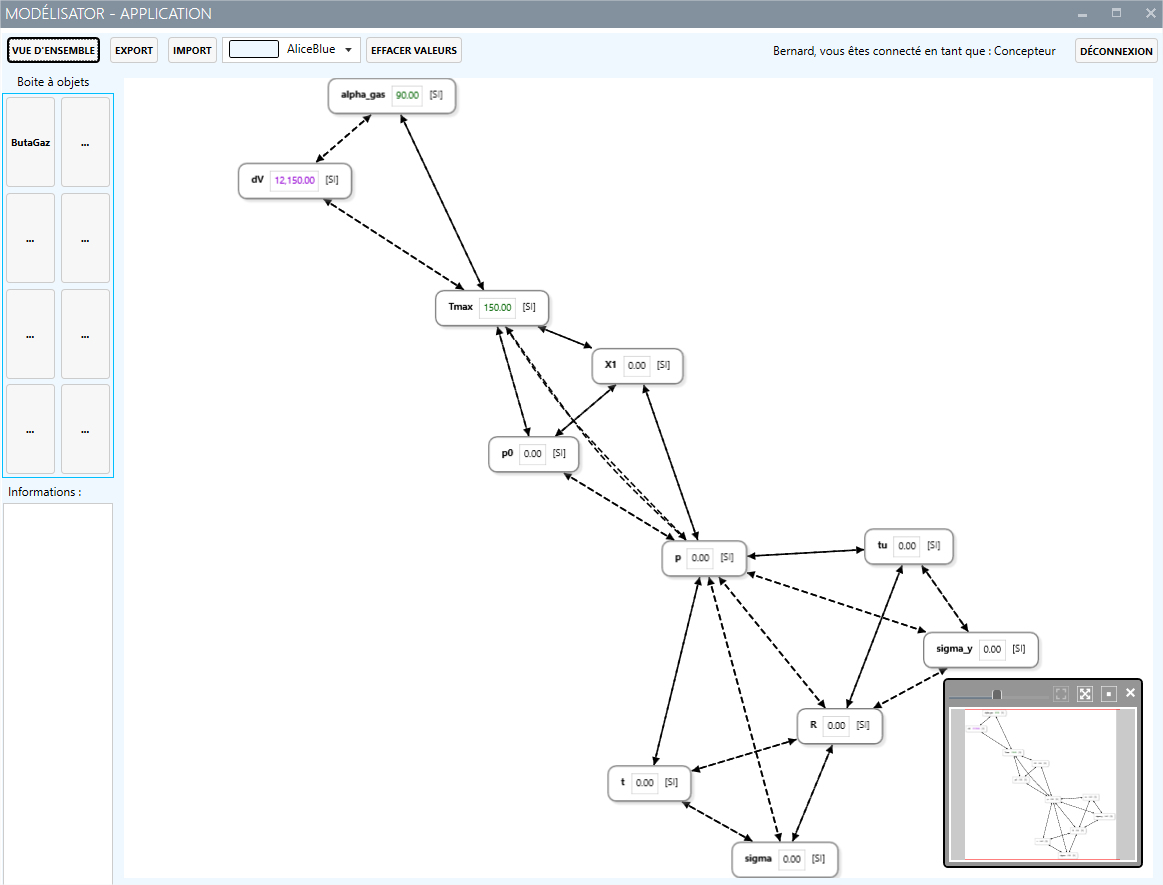
## 4) Le calcul de valeurs

Si Bernard avait déjà utilisé l’application dans une étude antérieure, il a peut-être exporté son étude via le bouton « Exporter » mis à disposition. Il peut donc également l’importer s’il veut continuer son étude.

Bernard souhaite maintenant calculer la valeur dV. Pour cela, il faut d’abord qu’il sache quelles autres grandeurs sont liés à celle-ci, afin d’y inscrire les valeurs qu’il possède. En cliquant sur la grandeur, il fait apparaître les liens avec les autres grandeurs physiques dont elle dépend. Elle est également surlignée en bleu afin d’identifier la grandeur sur laquelle on a cliqué.



On voit que dV est lié à Tmax et à alpha\_gas. Prenons le cas où le premier vaut 150 et le deuxième 90. Comme les valeurs sont inscrites à la main, elles s’affichent en vert. Tout de suite après, la valeur dV s’affiche automatiquement en violet, car elle est calculée.



En revenant au graphe complet (en cliquant sur « Vue d’ensemble »), on peut voir que les valeurs en question sont toujours présentes, avec les mêmes critères graphiques que précédemment.

# Bilan fin de projet

## L'objectif initial est-il respecté ?

L’objectif a été respecté, car tout ce qui devait être développé l’est.

## Quels sont les points positifs de ce projet ?

Ce projet nous a permis de réellement réaliser une application dans les règles de l’art et surtout de penser l’architecture, de se poser les questions indispensables au bon déroulement du projet. De plus, nous avons développé nos compétences de travail d’équipe et de mutualisation des efforts.

De plus, l’utilisation du paradigme MVVM avec toutes les règles de développement qu’il implique nous permet de passer de travaux approximatifs à des logiciels de qualité professionnelle.

Enfin, la modélisation d’objets physiques avec les dépendances de toutes ses grandeurs physiques est un sujet très intéressant qui pourraient séduire plusieurs industriels.

## Quels sont les points à améliorer ?

Nos ambitions ne sont pas à l’échelle du temps que l’on dispose. Une approche plus raisonnable en se concentrant sur la partie IHM au risque de ne pas implémenter les calculs des grandeurs physiques était une alternative possible.

## Respect du planning ?

La date initiale était fixée au 01/07/2015, et l’application était parfaitement fonctionnelle ce jour. Cependant, le planning n’a pas pu être respecté dans toutes ses lignes, car plusieurs retards sont apparus (formation à faire à une partie de l’équipe concernant le langage de programmation par exemple).

# Remerciements

Remerciements à l’ensemble de l’équipe pédagogique pour sa disponibilité, sa patience et surtout ses enseignements indispensables au succès de notre projet.

Spirit Master Team©