|  |
| --- |
| Rapport fin du projet |
|  |
|  |
| **Réalisé par :**  **Cédric Allemand**  **Quentin Bonichot**  **Omar Latreche**  **TahaMerrika** |

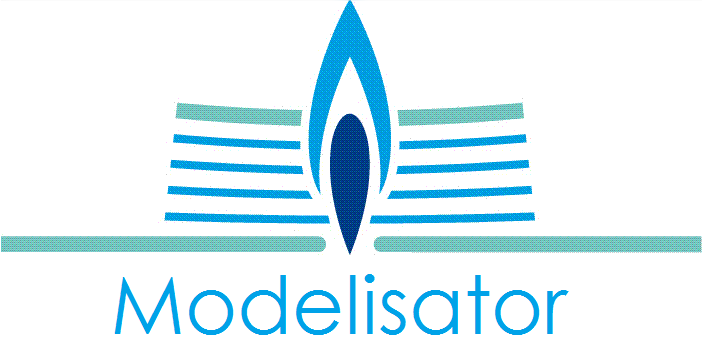
**Encadré par :**

**M. SaidBoujedli**

**M.Jean-Alain Fougère**

**Université de Technologie Belfort Montbéliard**

**Année universitaire 2014 /2015**



|  |
| --- |
| Rapport fin du projet |
|  |
|  |
| **Réalisé par :**  **Cédric Allemand**  **Quentin Bonichot**  **Omar Latreche**  **TahaMerrika** |



**Encadré par :**

**M. SaidBoujedli**

**M.Jean-Alain Fougère**

**Université de Technologie Belfort Montbéliard**

**Année universitaire 2014 /2015**

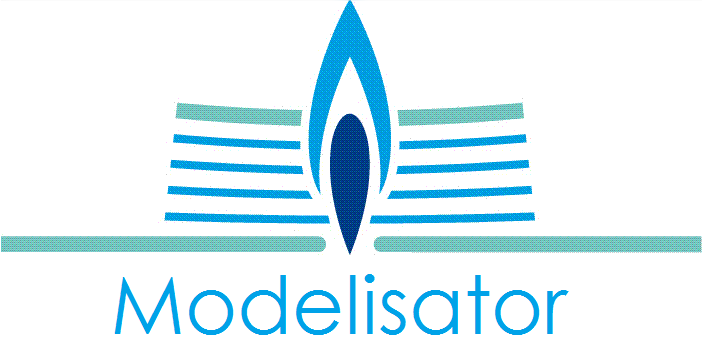


Table des matières

[I) Définition des objectifs 2](#_Toc423015654)

[1) Objectifs du projet 2](#_Toc423015655)

[2) Analyse « MALIN » de l’objectif 2](#_Toc423015656)

[3) Liste des taches à faire : 3](#_Toc423015657)

[4) Justification et Méthode : Phase Définition du projet 5](#_Toc423015658)

[II) Analyse des besoins et faisabilité 6](#_Toc423015659)

[1) CQQCOQP 6](#_Toc423015660)

[2) Liste des contraintes 7](#_Toc423015661)

[3) Liste des exigences 8](#_Toc423015662)

[4) Etat de l'art 9](#_Toc423015663)

[5) Analyse Faisabilité 9](#_Toc423015664)

[6) Justification et Méthode : Phase d’analyse des besoins et faisabilité 10](#_Toc423015665)

[III) Conception générale : 11](#_Toc423015666)

[1) Analyse technique 11](#_Toc423015667)

[2) Analyse fonctionnelle : 12](#_Toc423015668)

# 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** :  Grace 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 taches à faire :

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **TODO list on 28/05/2015** | | | | | | | |
| **Mission** | **But** | **Objectif** | **Livrable** | **Réfèrent** | **Date** | **Ressource** | **Etat** |
| **Définition des objectifs** | Objectifs |  | SMART/ MALIN | \* | 21/05/2015 | 2 | 3 |
| Choisir un nom |  | NOM | \* | 21/05/2015 | 3 |
| Choisir un Logo |  | Image du Logo | Q | 21/05/2015 | 0,5 | 3 |
| TO-DO list |  | ici | C + O | 21/05/2015 | 0,5 | 3 |
| Documentation | Justification et méthode | Justication phase d'analyse | \* | 28/05/2015 | 0,5 | 3 |
| **Analyse des besoins et faisabilité** | CQQCOQP | Analyse des besoins |  | Q + O | 28/05/2015 | 0,5 | 3 |
| Liste des contraintes |  | Analyse contraintes | T | 28/05/2015 | 0,5 | 3 |
| Liste des exigences |  | Analyse des requis | Q | 28/05/2015 | 0,5 | 3 |
| Etat de l'art | Analyse de l'existant |  | T | 28/05/2015 | 0,5 | 2 |
| Analyse Faisabilité | Faisabilité du projet | Analyse de faisabilité | C | 28/05/2015 | 0,5 | 2 |
| CDC | Reprend toutes les analyses | cdc | Q |  | 0,5 | 0 |
| Documentation | Justification et méthode | Justication phase de définitions | \* | 10/06/2015 | 0,5 | 3 |
| **Conception générale** | Analyse Technique | Le choix technologique, la description des traitements, le nouveau planning | Rapport technique + spécifications | \* |  | 1 | 0 |
| Analyse Fonctionnelle | la description des fonctions; L’enchainement des écrans (avec un maquettage) | Rapport fonctionnelle + maquettage écran | \* |  | 0,5 | 0 |
| **Conception détaillée** | Lot de travail | Découper le projet en un ensemble de module | PBS | \* |  | 0,5 | 0 |
| Répartition en taches | Répartir l'ensemble des fonctionnalités en taches | WBS | \* |  | 0,5 | 0 |

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Développement** |  |  |  | \* |  |  | 0 |
| Planning |  | PERT | \* |  | 0,5 | 0 |
|  |  | GANTT | \* |  | 1 | 0 |
| Responsabilités |  | RACI | \* |  | 0,5 | 0 |
| Gestion des risques |  | SWOT | \* |  | 1 | 0 |
| Modification CdC |  | révision 3 cdc | \* |  | 2 | 0 |
| Budget / Ressources | Analyse des ressource et de besoin pour réaliser le projet | tableur budget | \* |  | 0,5 | 0 |
| Suivi et pilotage | Ensemble de réunion, méthode AGILE | rapport réunion | \* |  | 1,5 | 0 |
| **Tests** | Prototype | Réaliser une version test du livrable souhaité | paquet | \* |  | 15 | 0 |
| **Recette** | Finalisation du livrable | Test d'intégration et test unitaire | cahier de test | \* |  | 4 | 0 |
| **Documentation** | Rapport + PPT |  |  | \* |  |  | 0 |
| Rapport fin de projet Globale | analyse, objectif, planning, objectif, Ressources |  | \* |  |  | 0 |
| Rapport fin projet Personnel | Analyse personnel sur le déroulement du projet |  | \* |  |  | 0 |
| Transfert du livrable | Remise du rapport de projet et prototype | récépissé de livraison | \* |  | 2 | 0 |
| **Accompagnement** | Formation MOA/Juge | Documentation pour la formation à l'utilisation du livrable | doc | \* |  | 3 | 0 |
| **Mise en production** | Réaffectation des ressources sur d'autre projet |  | retour en entreprise | \* |  | 0,5 | 0 |
| **Assistance et maintenance** |  |  |  | **Total Jour/Homme:** | | **41** | 1 |
|  |  | 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 isolé.
* 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 rôle, 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 taches et livrable a 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 24 juin 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 répond 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 de 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, 2 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) gratuits 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.

**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.

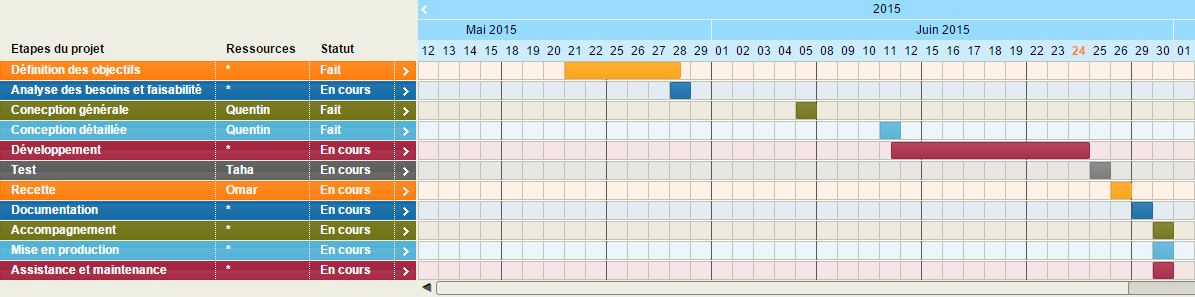
Le 1er traitement consiste à recenser la liste des objets ainsi que la liste des grandeurs physiques associées à chaque objet, par la suite il faut représenter sous forme de graphe, l’objet relié à ses premières grandeurs physiques.

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 relations.

* 1. **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.



## Analyse fonctionnelle :

* 1. **Description des fonctions**

**Connexion :**

Il y a plusieurs type 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 physique en interdépendances avec celle-ci.

**Calcul des valeurs des grandeurs physique :**

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 d’application jusqu’à sa fermeture.

**Enregistrer la saisie :**

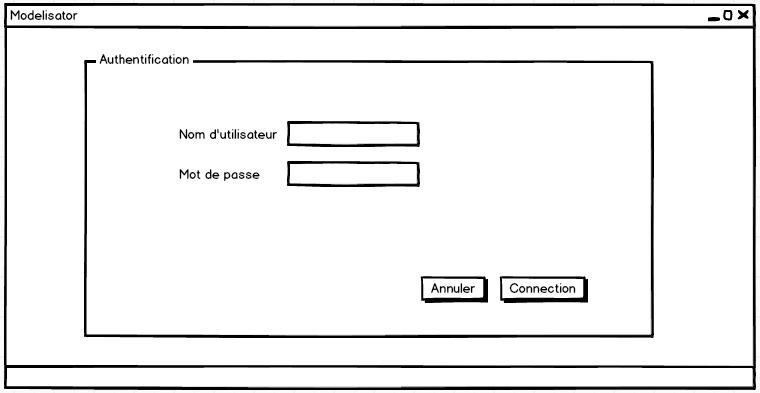
Cette fonctionnalité permet de garder la saisie des valeurs sous forme d’un projet, et on peut ouvrir le projet dans l’état que nous l’avions laissé.

* 1. **Maquettage**

Avant de passer aux étapes suivantes dans la conception du projet, il a était nécessaire de concevoir la 1ère 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 IHMs de l’application.

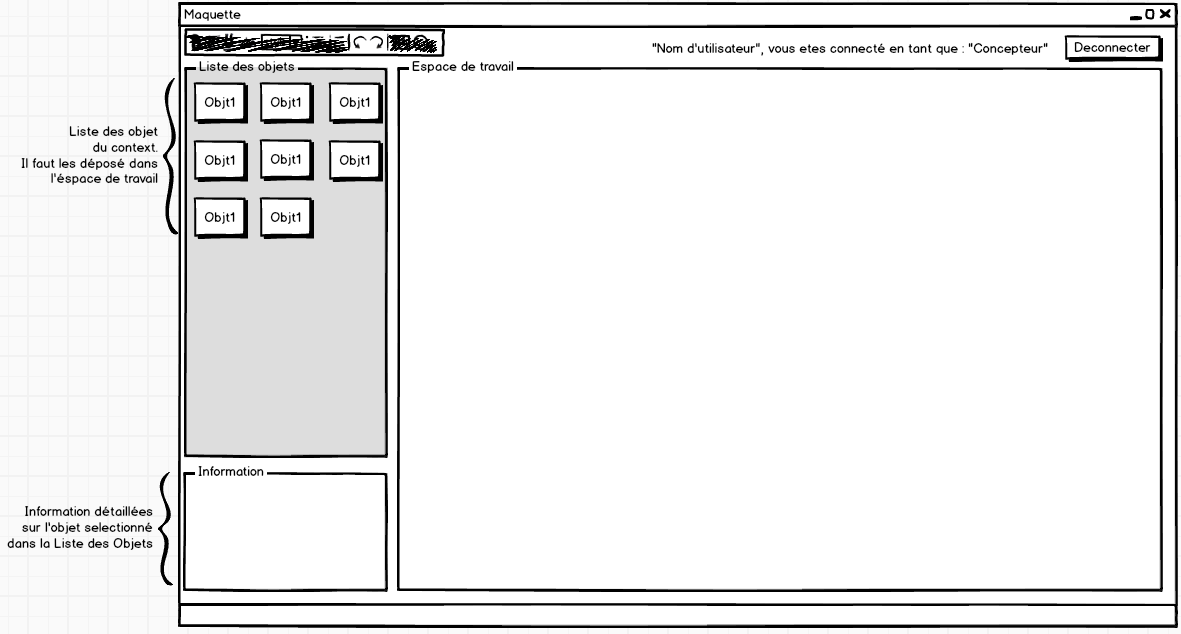
**Connexion**

C’est l’écran qui apparaisse après l’ouverture de l’application.



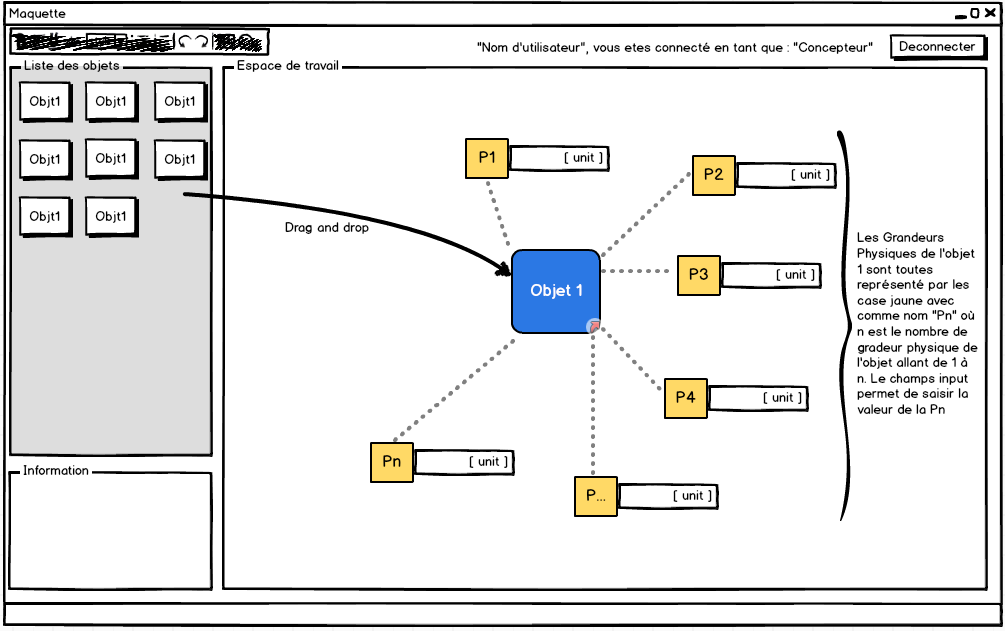
**1er écran après connexion : affichage de la liste des objets**

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.



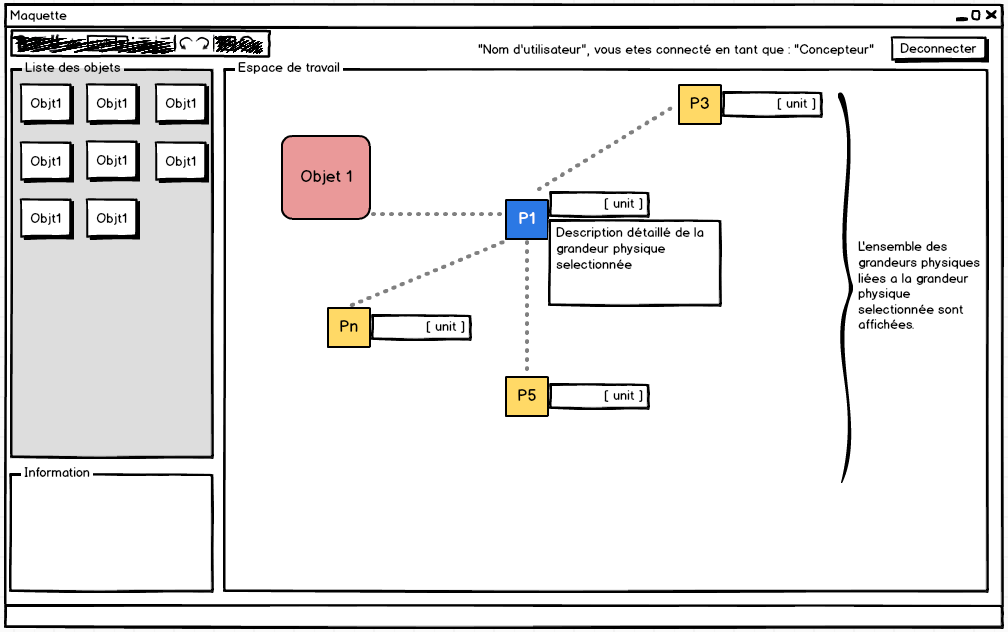
**Glisser déposer un objet**

Les grandeurs physiques d’un objet sont affichés après un glisser déposer de l’objet en question.



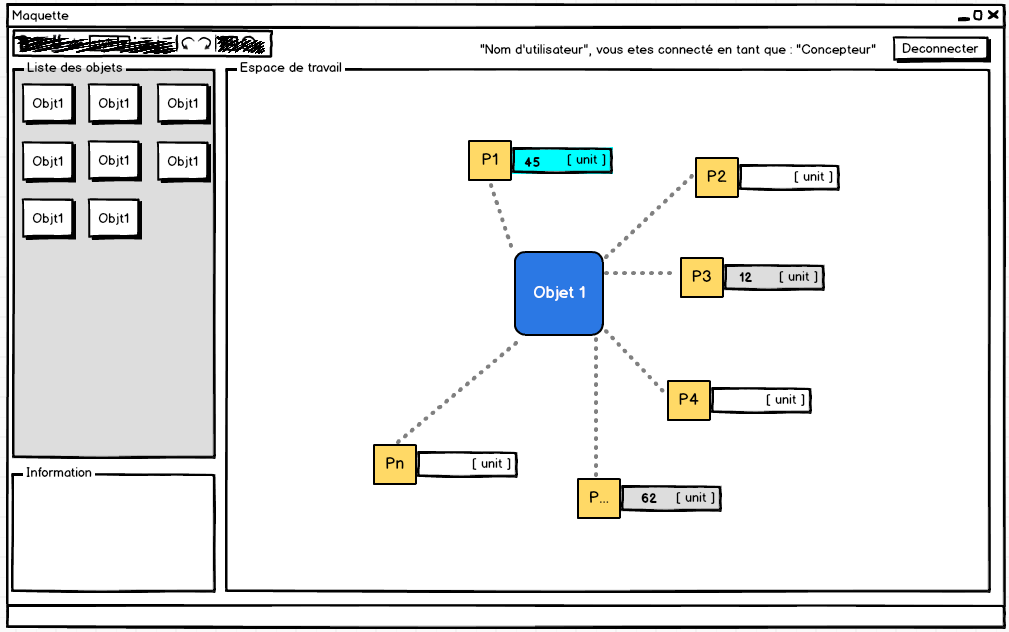
**Choix d’une grandeur physique**

Le clic sur une grandeur physique permet d’afficher toutes les autres grandeurs physiques qui sont en relation avec elle.



**Saisie d’une grandeur physique**

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



# Conception détaillée

Avant de commencer dans le développement, on a découpé le projet en un ensemble de module, pour pouvoir après répartir les tâches, comme on a choisi la méthode MVVM, alors 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 tous ce qui est fonctionnalités de notre application, et la vue-modèle concerne tous ce qui est lien entre la vue et le modèle, pour lier chaque écran à la fonctionnalité correspondante.

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.

# Développement :

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

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | **Equipe projet** | | | |
| **To Do List** | 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 |

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

|  |  |  |  |
| --- | --- | --- | --- |
| **Légende** | | | |
| **R** | Responsable de la réalisation | **C** | Doit être consulté |
| **A** | Approbation finale pour l'autorité | **I** | Doit être informé |