qqqqq

**R. CHRETIEN**

**A. BROCHARD**

**U. HASSED**

**V. HALLAY**

**Projet Madera**

**RIL 2018-2020**

**Spécifications techniques**

Table des matières

[Contrôle de saisie et données 3](#_Toc17901469)

[Description des modules et traitement de l’information associé 4](#_Toc17901470)

[a. Liste des modules primaires 4](#_Toc17901471)

[b. Description des modules primaires 4](#_Toc17901472)

[2. Choix technique de développement 5](#_Toc17901473)

[a. Type d’application 5](#_Toc17901474)

[Client léger : 5](#_Toc17901475)

[Client Lourd : 5](#_Toc17901476)

[Conclusion 6](#_Toc17901477)

[b. Langage de programmation 6](#_Toc17901478)

[c. Logiciel de développement 7](#_Toc17901479)

[d. Système de gestion de base de données relationnelle 7](#_Toc17901480)

[e. Méthode de développement 8](#_Toc17901481)

# Contrôle de saisie et données

Afin d’éviter les erreurs de traitement et erreur humaine, nous allons ajouter des contrôles sur les saisies et les données existante du projet MADERA et spécifier la possibilité ou non de certain traitement.

Si dessous, un découpage selon les différentes parties, acteurs du projet pour une meilleure visibilité.

**Plan :**

* Il ne sera pas possible d’ajouter de module en dehors des murs initiaux du plan.
* Il ne sera pas possible de supprimer les murs initiaux du plan.
* Toutes les configurations du plan (Coupe/Plancher/couverture) en dehors de la gamme seront obligatoire pour l’enregistrement de celui-ci.
* Un plan peut être édité par n’importe quel commercial.

**Devis :**

* Le prix d’un devis est fixe au moment de sa génération, il ne sera pas modifié dans le futur même si les prix des modules ont augmenté.
* Un devis ne peut revenir à un état précèdent.

Projet

* Un projet est mono client.
* Les commerciaux ont accès a tous les projets.

Commerciaux

* Les commerciaux devront avoir une majuscule et un chiffre dans leur mot de passe.

**Client**

* Le nom du client ne contiendra que des lettres.
* Un client peut avoir autant de projet qu’il souhaite. Et donc autant de plan qu’il veut.
* Deux clients avec des informations identiques ne doit pas être possible.

# Description des modules et traitement de l’information associé

## Liste des modules primaires

Voici la présentation des fonctionnalités principales qui seront présente dans l’application finale. Chaque parti sera détaillé dans la deuxième partie.

* Identification
* Gestion client
* Gestion projet
* Gestion de devis
* Modélisation

## Description des modules primaires

Ci-dessous on peut retrouver la liste complète des fonctionnalités répartie selon leur fonctionnalité principale à laquelle elles sont rattachées.

Pour l’identification on retrouvera le test de la connectivité, ensuite l’utilisateur se connectera, s’il y a des erreurs dans les champs elles devront être affichées. Il y aura aussi la possibilité de se déconnecter.

En ce qu’il s’agit de la gestion client on pourra retrouver la liste des clients ainsi que leur info pour chacun. La possibilité d’ajouter et d’éditer un client sera présente également et l’affichage des erreurs dans les champs des formulaires liés aux clients.

Dans la gestion de projets nous pourront retrouver la liste des projets ainsi que les infos de chacun. On pourra également créer et éditer un projet. Ces dernières fonctionnalités seront également présentes pour les plans d’un projet. On pourra récupérer la liste des clients, consulter le devis, copier le plan charger les données paramètres. Et enfin pour tous les champs présents dans les fonctionnalités précédentes on affichera les erreurs éventuelles.

Pour la fonctionnalité gestion de devis, on pourra charger les composants, afficher les infos du plan, charger les données client et les données commerciales, on vérifiera que les stocks sont suffisamment remplis pour le devis, on calculera les prix et la possibilité d’exporter le devis sera présente. On pourra également appliquer une remise, afficher un dossier technique ainsi que la modification de l’état d’un devis. Pour tous les champs présents dans les fonctionnalités précédentes on affichera les erreurs éventuelles.

En ce qui s’agit de la fonctionnalité de la modélisation on pourra éditer les paramètres modifiables, sauvegarder le plan et vérifier ses contraintes. On pourra également choisir ou sélectionner un plan. Enfin on pourra mettre à jour l’affichage et charger les modules compatibles et les composants d’affichages.

# Choix technique de développement

## Type d’application

Contrainte principale :

L’application pour l’entreprise MADERA doit être utilisable sur tablette et ordinateur ainsi que disponible hors connexion.

Nos possibilités sont donc :

## Client léger :

**Avantages :**

* Mise à jour centralisé du site. Le déploiement de MaJ se fait facilement et rapidement.
* Accessibilité.
* Peu sensible aux pannes matérielles.

**Inconvénient :**

* Langages client/serveur différents
* Partie cliente nécessitant plusieurs langages
* Evolutions des navigateurs
* Les performances de l’application peuvent baisser s’il a un trop grand nombres utilisateurs simultanés.
* **Ne permet pas le hors connexion.**

## Client Lourd :

**Avantages :**

* Un seul langage pour les parties client et serveur
* Interaction avec le serveur simplifié.
* **Hors connexion possible.**

**Inconvénients :**

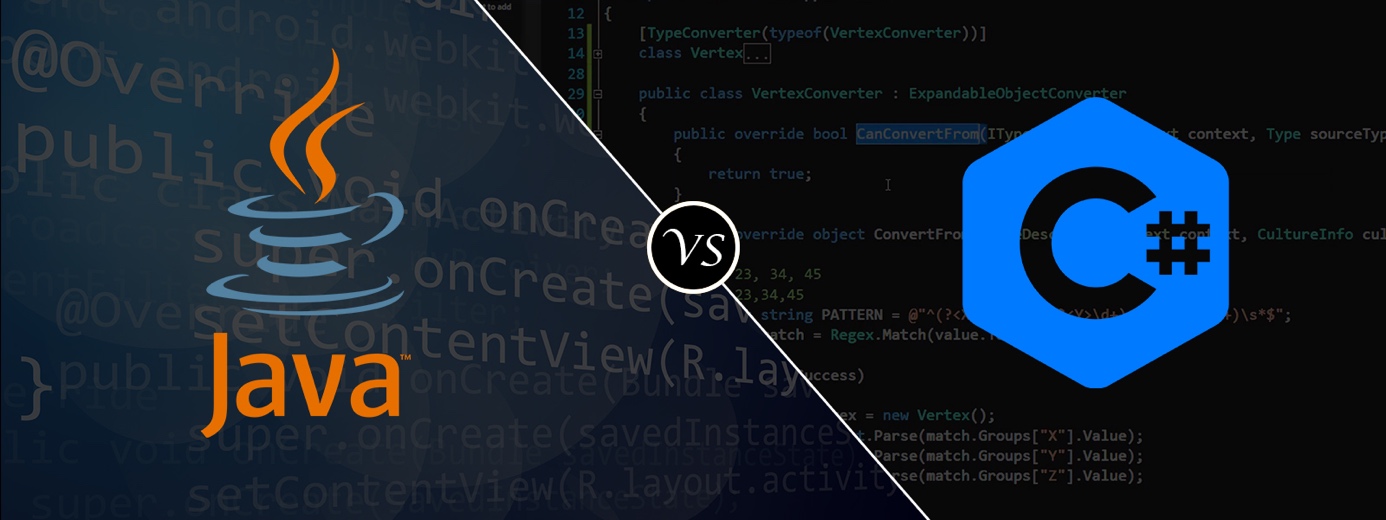
* Déploiement plus lourd
* Maintenance et Mises à jour plus lourdes
* Sécurité et sauvegardes dépendantes du poste client
* Performances en baisse si trop nombreux utilisateurs.
* Formation de l’équipe a la création de client lourd.

## Conclusion

L’obligation d’une utilisation hors connexion de l’application nous impose le choix d’un client lourd.

## Langage de programmation

Pour un client lourd tel que l’application que nous allons réaliser durant le projet MADERA, deux langages sortent du lot.



Le langage Java et le C# sont très clairement similaires et leurs performances globales pour la réalisation d’un projet tel que celui-ci se rejoignent.

Java et C# partagent un ensemble de fonctionnalités communes :

* Compilation dans un langage intermédiaire indépendant de la machine et exécution dans un environnement dédié (une machine virtuelle)
* Gestion automatique de la mémoire grâce à un ramasse-miettes
* Introspection pour manipuler dynamiquement les objets
* Toutes les classes héritent d'une même classe (Object) et sont allouées sur le tas
* Pas de support de l'héritage multiple mais utilisation d'interfaces
* Tout doit être encapsulé dans une classe : il n'existe pas de fonctions ou constantes globales
* Gestion des erreurs grâce aux exceptions

Notre principal facteur de décision pour nous est donc la connaissance du langage par les membres de l’équipe. Notre équipe ne possède pas de membre ayant réalisé de client lourd C# ou Java par le passé. Il faudra donc se former sur le langage pour réaliser ce projet.

|  |  |  |
| --- | --- | --- |
|  | Java | C# |
| Connaissance du langage | 1/10 | 4/10 |

**Nous avons donc fait le choix du C# pour la réalisation de l’application.**

Pour l’interface graphique de notre application, nous utiliserons XAML pour sa compatibilité avec le langage C#. Nous permettant de séparer les développements de l’équipe en deux parties.

L’une sur les traitements avec le serveur et la seconde sur la conception graphique de l’application.

## Logiciel de développement

Le logiciel de développement que nous avons choisis d’utiliser est Visual studio. Nous l’utiliserons car tous les projets de l’entreprises pour ce genre d’applicatif lourd est réalisé dessus. Nous avons donc les licences à disposition.

De plus il permet de générer des applications C#/XAML sans soucis.



## Système de gestion de base de données relationnelle

Dans le cadre du projet, il nous faut également faire le choix d’une base de données.

Nous allons tacher de mettre en place plusieurs bases de données, certains qu’on pourrait qualifier d’esclaves, elles seront sur les parties hors internet du projet. Une fois la connexion internet rétablies sur les machines possédants les bases locales, elles se synchroniseront avec la base dites Maitre qui elle se trouvera sur le réseau.

Notre choix de SGBD c’est porté sur MySQL car :

#### Rapide

Le serveur MySQL est très rapide. Des tests de performances sont disponibles sur le site de MySQL

#### Facile à utiliser

MySQL est beaucoup plus simple à utiliser que la plupart des serveurs de bases de données commerciaux.

#### API diverses

On peut effectuer diverses opérations sur une base MySQL en utilisant des interfaces écrites en C, Perl, C++, Java, Python, PHP.

#### Connexion et Sécurité

MySQL dispose d'un système de sécurité permettant de gérer les personnes et les machines pouvant accéder aux différentes bases.

#### Portabilité

MySQL tourne sur divers systèmes tels que Unix, Windows, Linux ou OS/2.



## Méthode de développement

Dans le cadre du projet, la méthode de développement choisie par l’équipe sera la méthode AGIL.

Nous allons procéder à des pokers pour définir les priorités des modules du projet, ainsi que des périodes de rush pour le développement. Des réunions régulières sur l’avancement du projet seront également mises en place pour avoir une bonne cohésion et lisibilité entre les membres l’équipe et leur travail associé, cela permettra aussi de prévenir d'éventuels blocages ou manques de connaissances.

La méthode AGIL nous permettra d’avancer le plus rapidement possible dans le projet tout en ayant une vision sur le travail de chacun et une connaissance de l’avancée dans chaque module du projet développé au même moment.

