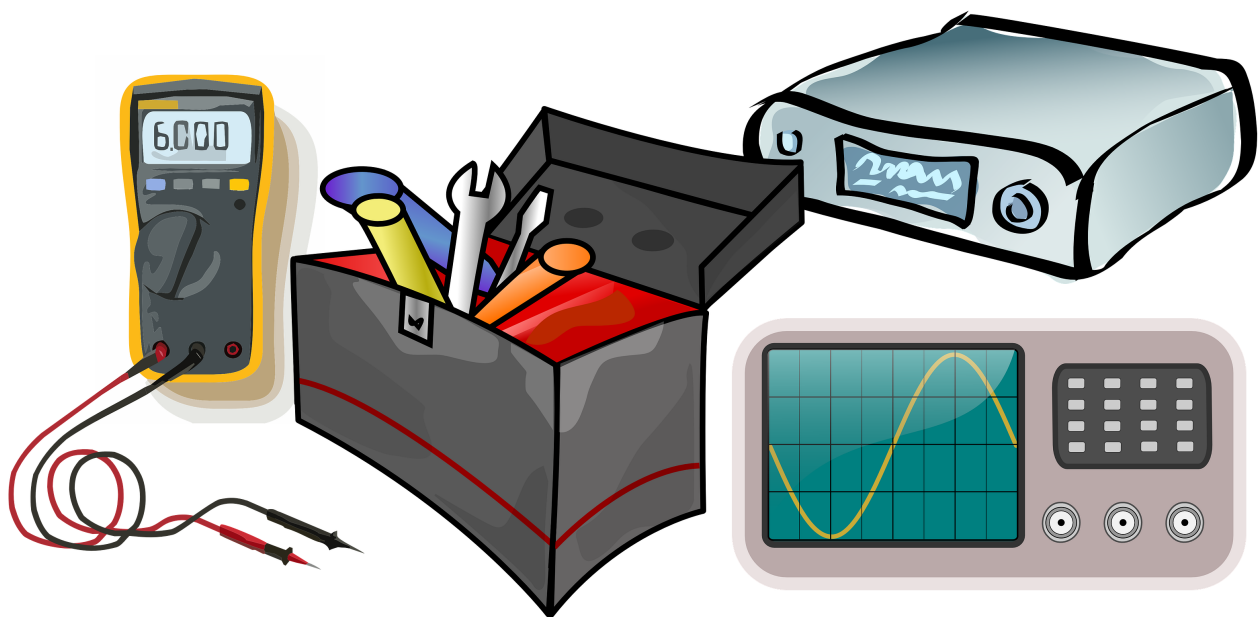


RAPPORT PERSONNEL

Projet: Localisation de matériel



BTS IRIS
Lycée Modeste Leroy

Evreux

Table des matières

1 Présentation du projet.....	3
1.1. Rappel de la problématique.....	3
1.2 Zoom sur la conception préliminaire.....	4
1.2.1 Diagramme des cas d'utilisation.....	4
1.2.1.1 Cas d'utilisation Localiser.....	5
1.2.1.2 Cas d'utilisation Gérer (Matériels).....	5
1.2.1.3 Cas d'utilisation Authentifier.....	6
1.2.2 Diagramme de déploiement.....	7
1.3 Mise en œuvre personnelle.....	8
1.4 Conception détaillée.....	9
1.4.1 Diagramme de la couche métier.....	9
1.4.2 Diagramme de la couche physique.....	10
1.4.3 Diagramme de la base de donnée.....	11
2 Réalisation.....	13
2.1 Classes entités et interfaces.....	13
2.2 Réalisation sous Qt.....	17
2.2.1 Test de la couche physique.....	17
2.2.2 Projet de gestion de matériel.....	22
2.3 Réalisation sous Netbeans.....	26
2.3.1 Design pattern MVC.....	26
2.3.2 Différentes balises JSTL.....	28
2.3.3 Template.....	29
2.3.4 Site web de localisation de matériel.....	30
3 Conclusion.....	34

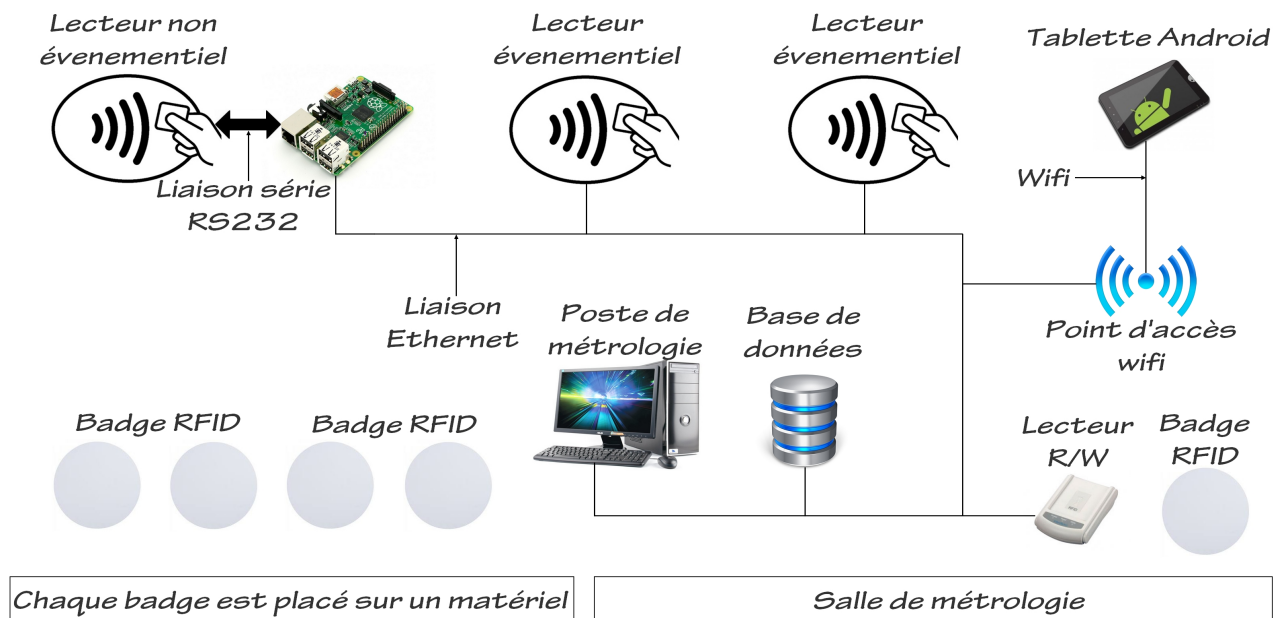
1) Présentation du projet

1.1) Rappel de la problématique

Comment localiser un ou des matériel(s) dans un parc de plusieurs dizaines d'hectares?

Une cellule de métrologie peut comprendre un parc de plusieurs dizaines d'hectares. Retrouver un appareil s'avère alors fastidieux pour un métrologue ou même pour un technicien. Pour résoudre cela, le projet consiste à pouvoir retrouver un appareil ou un groupe d'appareils par une simple recherche sur une tablette ou sur un ordinateur.

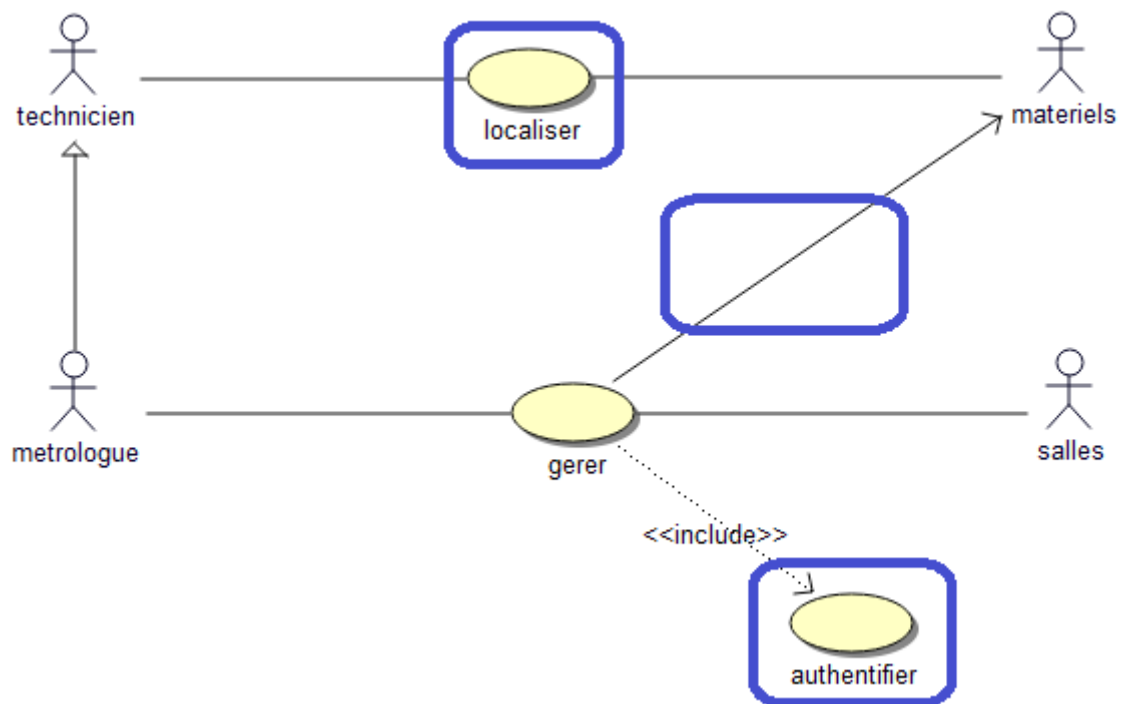
Chaque salle devra avoir à ses entrées un portique (carte Raspberry pi et un lecteur RFID). Sur les appareils de métrologie, il devra y avoir un tag. Le tag sera lu par le portique dès qu'il passe devant. Le métrologue pourra contrôler tout ses matériels et consulter où ils sont via le plan mis à jour dès qu'un appareil est détecté par un portique.



Schémas synoptique

1.2) Zoom sur la conception préliminaire

1.2.1) Diagramme des cas d'utilisation



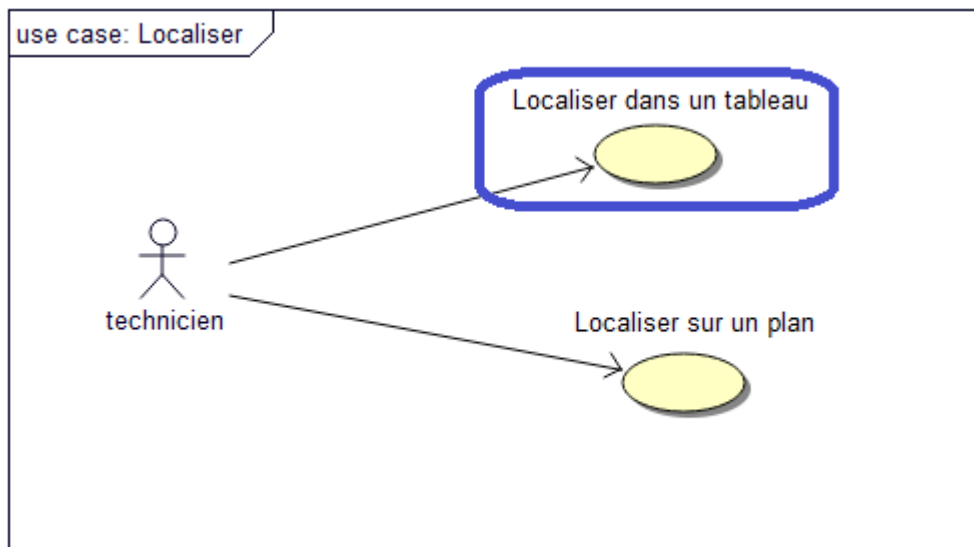
J'étais en charge de plusieurs cas d'utilisations selon sur quel projet je travaillais.

Pour le client Qt, je devais m'occuper du cas d'utilisation "authentifier" afin qu'une personne non désirée ne puisse pas changer la base de donnée ou écrire sur les badges grâce aux applications.

Mon client Qt devais permettre au métrologue de pouvoir gérer le matériel. Cette même application écrit sur les badges et modifie la base de données.

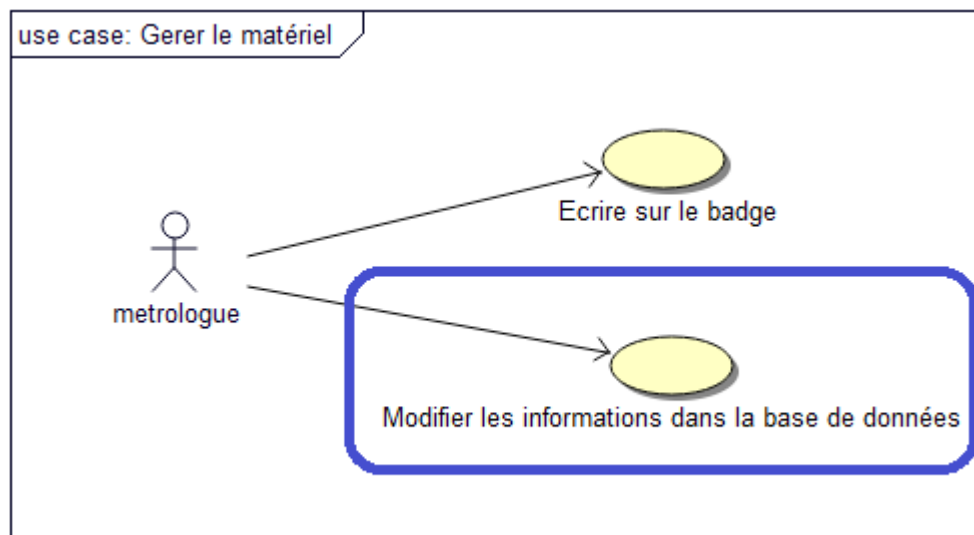
Pour ce qui est du site web, il devais permettre d'afficher le matériel avec sa localisation dans un tableau.

1.2.1.1) Cas d'utilisation Localiser



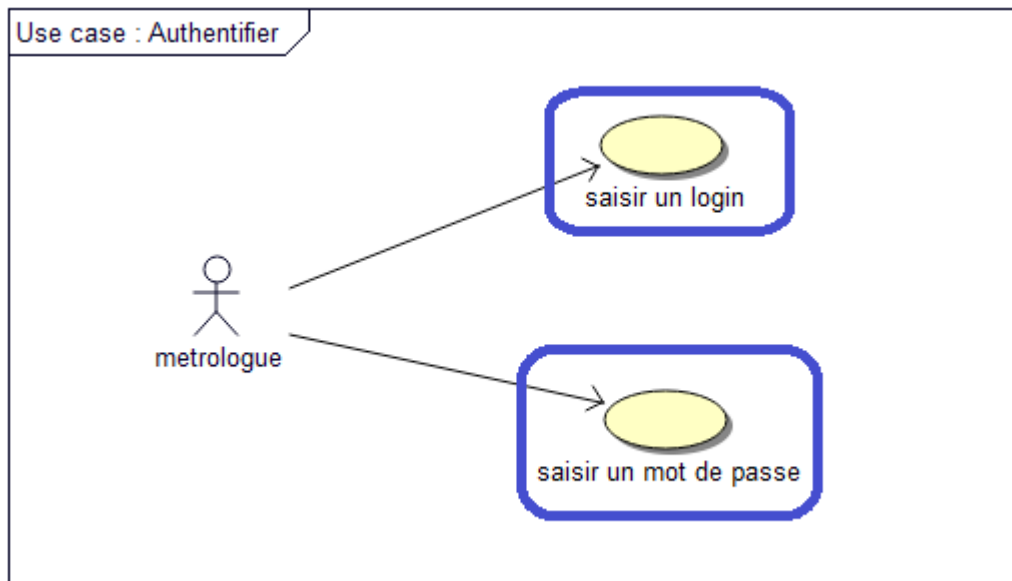
Un technicien peut à tout moment localiser un ou des matériel(s) selon des critères spécifiques, soit directement depuis le site web sous la forme d'un tableau, soit depuis l'application Qt qui se trouve dans la salle de métrologie. Cette dernière permet de localiser le matériel depuis un plan.

1.2.1.2) Cas d'utilisation Gérer (Matériel)



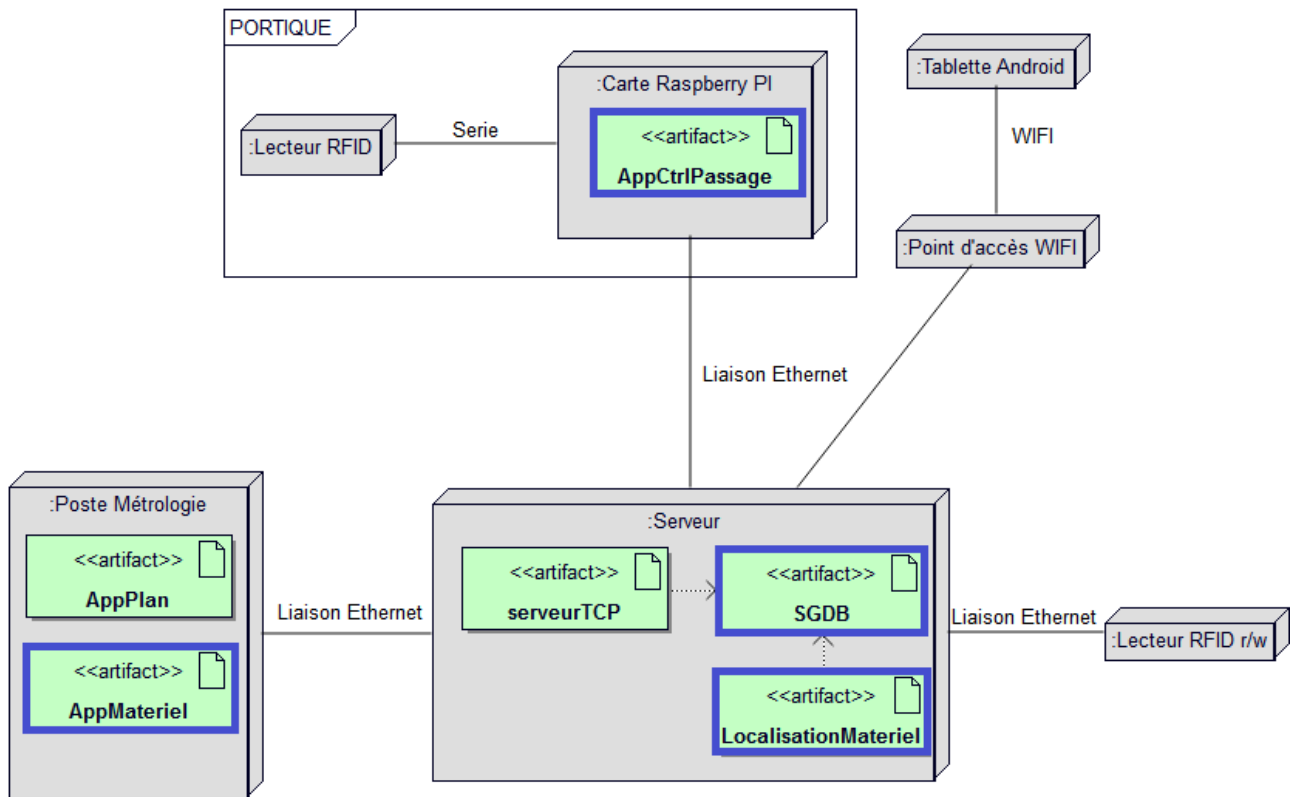
Un métrologue peut à tout moment ajouter, supprimer ou modifier un matériel. Pour cela, il passe par l'application située dans le poste de métrologie. Cette même application permet de modifier la base de données mais fait aussi appel à l'une des parties dont je n'avais pas la charge pour pouvoir écrire sur un badge depuis un lecteur r/w.

1.2.1.3) Cas d'utilisation Authentifier



Pour accéder à la partie gestion du système, une personne doit préalablement s'authentifier. Cette authentification demande à saisir un login correct (qui existe dans la base de données) et le mot de passe qui correspond à ce login.

1.2.2) Diagramme de déploiement



Le diagramme de déploiement me permet de vous présenter l'infrastructure globale du projet et les relations entre les différents composants afin que le projet final soit fonctionnel. Il me permet aussi de vous présenter les différentes tâches qui m'ont été confiées.

La première tâche qui m'a été confiée est la gestion de la base de données. Elle contient toutes les informations nécessaires au fonctionnement du projet et se trouve au niveau du serveur pour que toutes les applications puissent communiquer avec elle si besoin.

La deuxième tâche était la réalisation d'un client lourd (IHM) pour ajouter, modifier ou supprimer un matériel de la base de données ainsi que modifier le contenu du badge associé au matériel. Cette application se trouve dans le poste de métrologie, au même endroit où se trouve le lecteur r/w permettant de modifier le contenu d'un badge.

La dernière tâche était de réaliser un client riche (site web) afin de permettre à n'importe quelle personne de connaître l'emplacement d'un appareil grâce à une tablette Android ou autres matériels connectés au réseau de la cellule de métrologie.

1.3) Mise en œuvre personnelle



Le système d'exploitation utilisé lors du développement des applications est windows, mais la cellule de métrologie utilise linux. Les projets étant portable, l'utilisation de windows ne pose aucun problème et respecte les contraintes de l'environnement.



Contrainte de l'environnement: Le développement des logiciels devait être réalisé avec Qcreator/Qt-designer.



Les applications sous Qt sont en C++.
Le C++ est un langage de programmation compilé.



Je devais préparer la machine pour l'exploitation d'un serveur MySQL avec phpmyadmin. Elle fait partie des contraintes de la partie "Gestion Métrologie et Donnée".



Langage utilisé par la base de données Mysql.



Contrainte de l'environnement: Le développement de l'application web devait être réalisé avec Netbeans.



Java est un langage de programmation orienté objet.
Les classes sur Netbeans sont codés en Java.



J2EE est une plate-forme fortement orientée serveur pour le développement et l'exécution d'applications distribuées. Elle permet la création d'un site web grâce à des composant comme les servlets ou encore à ces bibliothèques de balise comme JSTL et en facilite le découpage de l'application (MVC).



Apache Tomcat est un conteneur web libre de servlets et JSP.
L'utilisation de ce serveur faisait partie des contraintes du cahier des charges.



L'HTML est un langage balise conçu pour créer et structurer des pages web.
Le Javascript permet d'exécuter des commandes du côté client.
Le CSS permet de faciliter la mise en page et la présentation d'un site web.

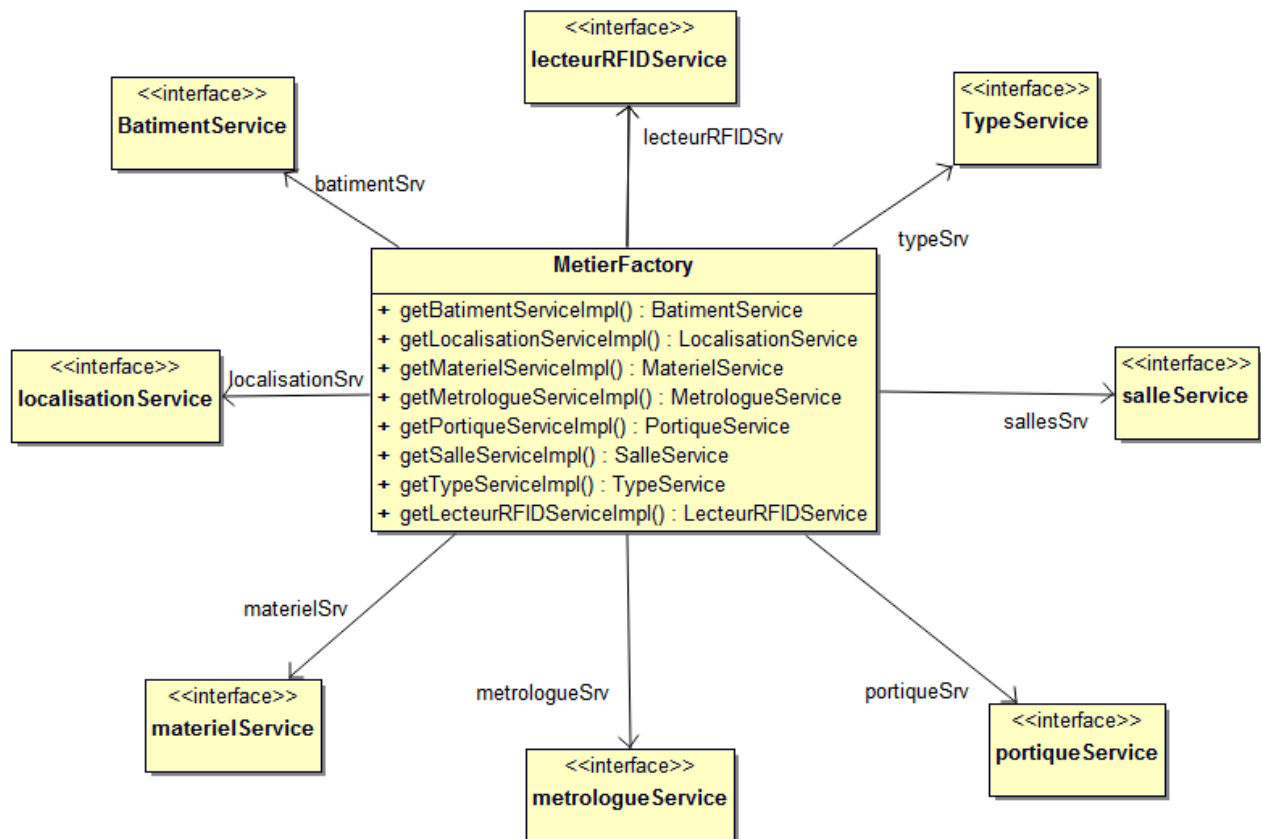


L'Ajx permet de construire des sites web dynamiques interactifs sur le poste client. C'est à dire qu'elles ne nécessitent le moindre rechargement visible par l'utilisateur de la page Web.

1.4) Conception détaillée

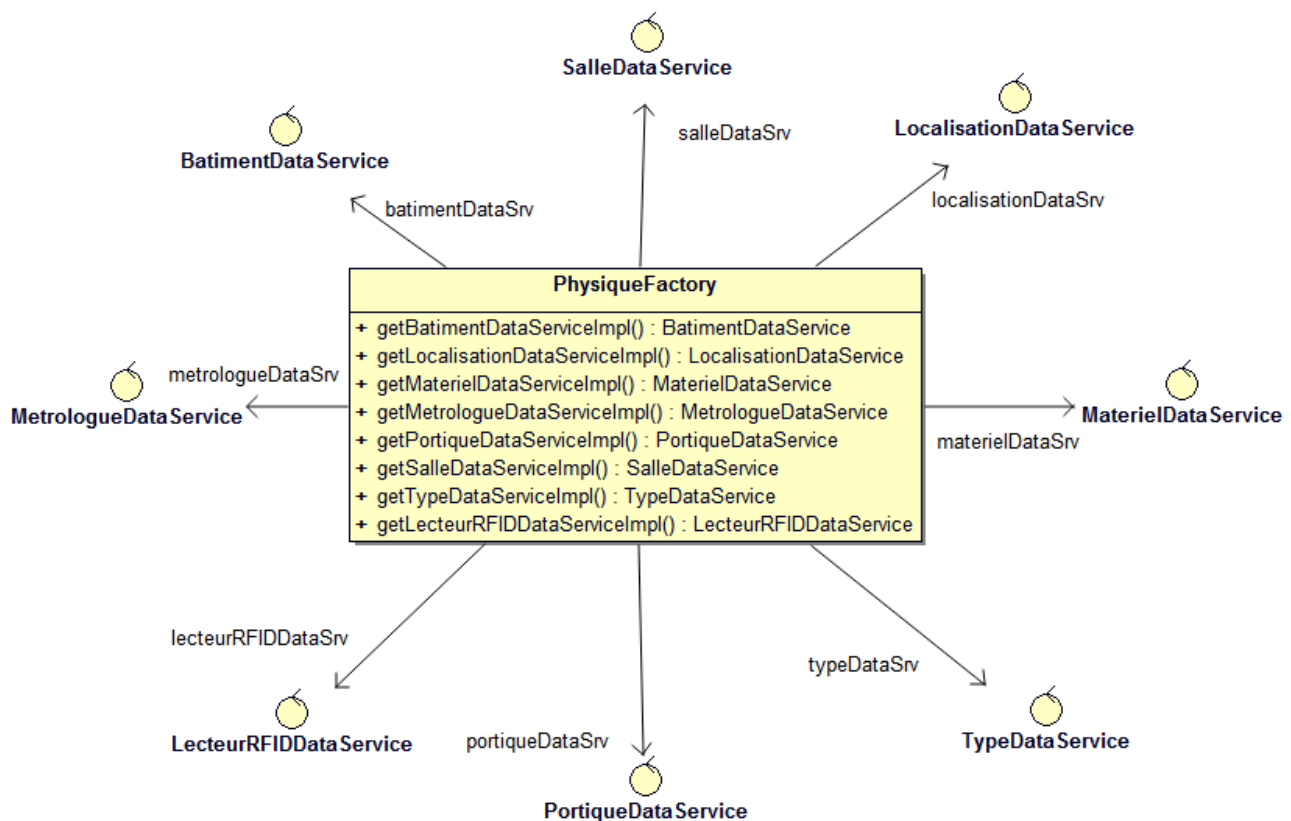
1.4.1) Diagramme de la couche métier

Ce paquetage contient les entités et leurs classes de service associées. Chaque classe de service est instanciée par la MetierFactory grâce à une méthode propre de cette factory. Les classes d'implémentation définissent les règles métiers, c'est à dire de contrôler les informations qui y circulent.

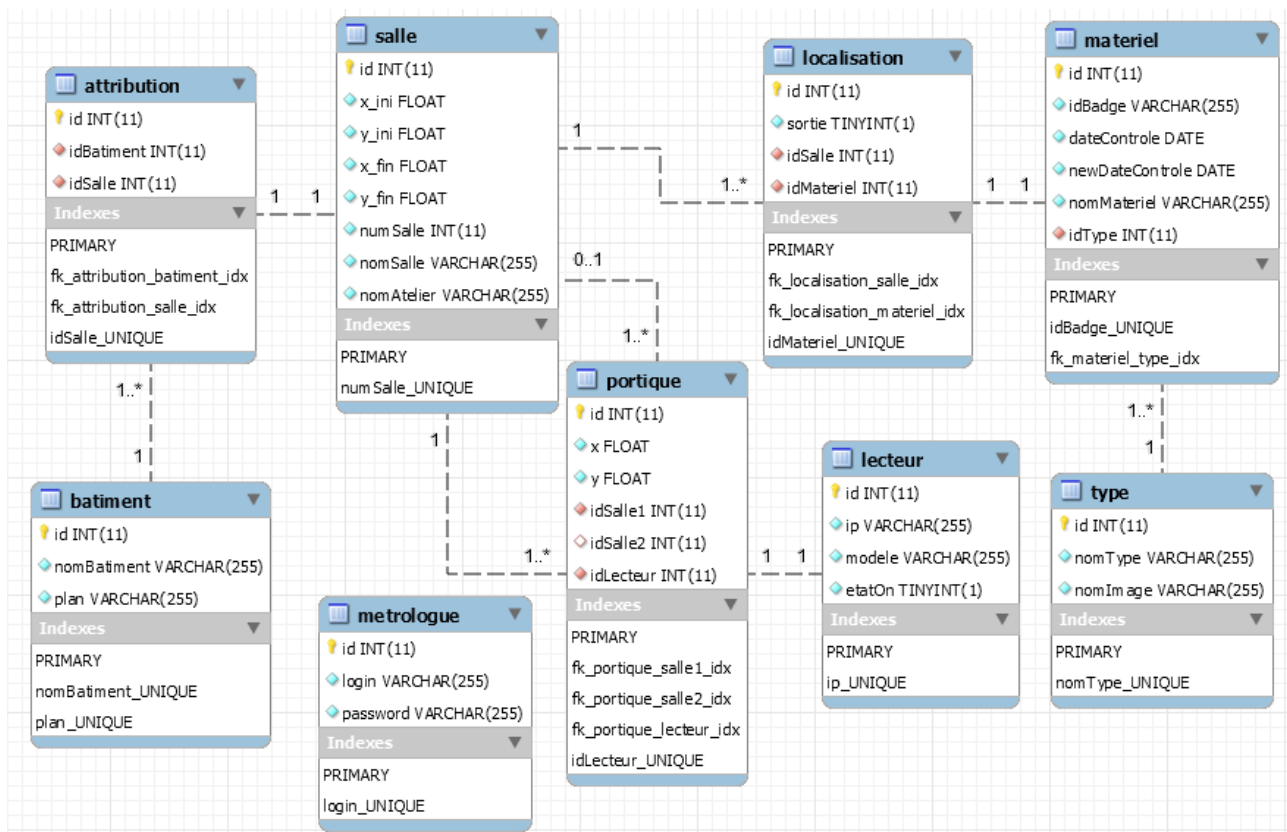


1.4.2) Diagramme de la couche physique

Le paquetage physique contient les classes de services des données associée aux entités de la couche métier. Chacune de ces classes est instancié par la PhysiqueFactory grâce à une méthode propre de cette factory. Les classes d'implémentation permettent d'exécuter des requêtes sur la base de données.



1.4.3) Diagramme de la base de données



Le diagramme de la base de données permet de représenter la base de données. On peut y retrouver toutes les classes présentes sur le diagramme entités avec les mêmes attributs à l'exception d'un cas. Une table attribution s'est rajoutée, elle est nécessaire puisque un bâtiment contient une liste de salles. En effet, dans une base de données il est impossible de passer directement une liste dans un champ, pour contourner cela, on utilise une table d'association.

Les clés primaires (clés jaunes) permettent d'identifier de manière unique chaque ligne de la table. Il ne peut y avoir qu'une clé primaire par table. Elles permettent à un champ d'être unique, indexé et sa valeur ne peut pas être nulle. Ici chaque clé primaire s'auto-incrémente lors d'une nouvelle ligne.

Les clés étrangères (losange rouge) permettent ainsi d'établir des liens entre plusieurs tables et garantissent que les valeurs de chaque ligne de la table référençant existent dans la table référencée. Les colonnes de la table référencée doivent faire partie d'une contrainte de clé primaire ou d'une contrainte d'unicité.

La clé étrangère de la salle 2 d'un portique peut être nulle puisqu'un portique n'est pas forcément associé à 2 salles mais peut l'être à une salle et un couloir. Le couloir ne faisant pas partie des salles.

Un champ unique ne peut pas avoir deux fois la même valeur et il est également indexé.

Un champ indexé permet d'organiser ses valeurs lorsqu'ils sont stockées dans la base de données (donc plus gourmand en écriture). En contre-partie, cela permet à la base de données de retrouver plus rapidement une information lors de la lecture. Malgré que sur le diagramme apparaissent seulement les champs indexés ayant une clé étrangère, tous les champs de la base de données sont indexés (à part ceux ayant une clé primaire ou ceux qui sont unique vu qu'ils le sont déjà).

Les tables sont reliées par une relation non identifiée, c'est à dire que l'id faisant référence à une autre table est une clé étrangère. Alors que pour une relation identifiée, le champ qui fait référence à une autre table est la clé primaire de la table référençant.

Les relations 1-1 signifie qu'un enregistrement de la table A se rapporte seulement à un seul enregistrement de la table B, il est donc unique.

Les relations 1-N signifie qu'un enregistrement de la table A se rapporte à un ou plusieurs enregistrement(s) de la table B.

Le moteur de stockage utilisé par MySQL pour ce projet est InnoDB. Malgré que ce moteur de stockage est plus lourd et plus lent que MyISAM, il permet de gérer les transactions (un ensemble de requêtes est soit exécuté intégralement soit pas du tout, mais jamais partiellement), et gère les clés étrangères.

L'interclassement est un ensemble de règles permettant la comparaison de caractères dans un jeu grâce à un numéro de code qui est associé à un caractère. Ce paramètre influe donc sur le résultat des tris. Il faut tout d'abord savoir que le tri est influencé par 3 arguments:

- La sensibilité à la casse: les majuscules et minuscules doivent-elles être traitées de la même façon?
- La sensibilité aux accents: Une lettre avec un accent doit-il être traitée de la même façon qu'une lettre sans accent?
- L'expansion: Est-ce qu'une ligature, par exemple 'œ', doit être traitée de la même façon que les caractères 'oe'?

L'interclassement utilisé pour ce projet est `utf8_unicode_ci`. Il est un peu plus lent par rapport à d'autres interclassement mais il permet de ne pas être sensible à la casse (pas de différence entre minuscule et majuscule) et aux accents (pas de différence entre accent et non accent) mais permet de gérer l'expansion (les ligatures sont traitées de la même façon que les deux caractères qui ont été fusionnés).

2) Réalisation

2.1) Classes entités et interfaces

La première chose à faire lors de l'étape de codage est de coder les classes entités. Chaque classe entité possède des attributs conformement à ceux du diagramme entités, les méthodes d'accès à celle ci (getter et setter) ainsi qu'un destructeur et un ou plusieurs constructeurs.

```
class Materiel
{
    private:
        long id;
        QString idBadge;
        QDate dateControle;
        QDate newDateControle;
        QString nomMateriel;
        Type type;
    public:
        Materiel();
        Materiel(QString idBadge, QDate dateControle, QDate newDateControle, QString nomMateriel, Type type);
        ~Materiel();

        long getId(void);
        QString getIdBadge(void);
        QDate getDateControle(void);
        QDate getNewDateControle(void);
        QString getNomMateriel(void);
        Type getType(void);

        void setId(long id);
        void setDateControle(QDate dateControle);
        void setNewDateControle(QDate newDateControle);
        void setNomMateriel(QString nomMateriel);
        void setIdBadge(QString idBadge);
        void setType(Type type);
};
```

Chacune de ces méthodes présentes dans le fichier d'en-tête possédant l'extension .h sont implémenter dans un fichiers à part portant l'extension .cpp.

```
Materiel::Materiel()
{
    this->id = 0;
    this->idBadge = QString();
    this->dateControle = QDate();
    this->newDateControle = QDate();
    this->nomMateriel = QString();
    this->type = Type();
}

Materiel::Materiel(QString idBadge, QDate dateControle, QDate newDateControle, QString nomMateriel, Type type)
{
    this->id = 0;
    this->idBadge = idBadge;
    this->dateControle = dateControle;
    this->newDateControle = newDateControle;
    this->nomMateriel = nomMateriel;
    this->type = type;
}

Materiel::~Materiel()
{
}

long Materiel::getId()
{
    return this->id;
}

void Materiel::setId(long id)
{
    this->id = id;
}
```

Toutes nos classes entités possèdent 2 constructeurs, un pour changer la valeur des attributs et un constructeur par défaut qui ne prend aucun paramètre et initialise les valeurs par défaut. Le constructeur par défaut est automatiquement appelé lors de la création d'un objet, par exemple lors de la création d'un objet Materiel, et nous permet d'initialiser les valeurs à 0 ou nulle pour éviter, par exemple, de se retrouver avec un nombre aléatoire généré automatiquement pour l'id. Le deuxième constructeur fonctionne comme des setters pour tous les paramètres passés. 'this' permet de faire référence à l'attribut de la classe et non le paramètre.

Lorsque l'objet sera détruit, le destructeur sera appelé. Le destructeur ne prend aucun argument et comme le constructeur, il ne renvoie pas de valeur de retour, même pas void.

On y retrouve aussi les getters qui permettent de récupérer la valeur d'un attribut privé depuis l'extérieur de la classe.

Et pour finir on y retrouve les setters qui permettent de changer la valeur d'un attribut depuis l'extérieur, tout en pouvant dans le cas échéant de faire des vérifications.

Maintenant que les classes entités sont terminées, il a fallu créer les différentes interfaces présentent sur le diagramme d'interface.

Les interface de service des classe entités offre ce que l'on appelle les service CRUD (Create, Read, Update, Delete). La méthode Add permet d'ajouter un Matériel, Remove d'en supprimer un et Update de le modifier. La méthode getAll permet de récupérer un nombre défini d'élément à partir d'un certain index alors que la méthode getCount permet de récupérer le nombre total d'élément. Les méthode getBy permet de récupérer un ou une liste d'élément selon la valeur du paramètres et getCount permet d'en récupérer le nombre. Les méthodes getList permettent de récupérer toutes les valeurs insérés pour cet attribut en évitant les redondances et getCountList permet de connaître le nombre de valeurs inséré à cet attribut sans les redondances.

```
namespace Metier {
class MaterielService
{
public:
    virtual Materiel add(Materiel materiel) = 0;
    virtual bool remove(Materiel materiel) = 0;
    virtual bool update(Materiel materiel) = 0;
    virtual QList<Materiel> getAll(int debut, int parPage) = 0;
    virtual int getCount() = 0;
    virtual Materiel getById(long id) = 0;
    virtual Materiel getByIdBadge(QString idBadge) = 0;
    virtual QList<QString> getListIdBadge(int debut, int parPage) = 0;
    virtual int getCountListIdBadge() = 0;
    virtual QList<Materiel> getByDateControle(QDate dateControle, int debut, int parPage) = 0;
    virtual int getCountByDateControle(QDate dateControle) = 0;
    virtual QList<QDate> getListDateControle(int debut, int parPage) = 0;
    virtual int getCountListDateControle() = 0;
    virtual QList<Materiel> getByNewDateControle(QDate newDateControle, int debut, int parPage) = 0;
    virtual int getCountByNewDateControle(QDate newDateControle) = 0;
    virtual QList<QDate> getListNewDateControle(int debut, int parPage) = 0;
    virtual int getCountListNewDateControle() = 0;
    virtual QList<Materiel> getByPeriode(QDate date, bool avant, int debut, int parPage) = 0;
    virtual int getCountByPeriode(QDate date, bool avant) = 0;
    virtual QList<Materiel> getByType(Type type, int debut, int parPage) = 0;
    virtual int getCountByType(Type type) = 0;
    virtual QList<Type>getListType(int debut, int parPage) = 0;
    virtual int getCountListType() = 0;
    virtual QList<Materiel>getByNomMateriel(QString nomMateriel, int debut, int parPage) = 0;
    virtual int getCountByNomMateriel(QString nomMateriel) = 0;
    virtual QList<QString>getListNomMateriel(int debut, int parPage) = 0;
    virtual int getCountListNomMateriel() = 0;
};
}
```

Une interface ne possède aucun attribut mais que des fonctions virtuelles pures. Les « = 0 » à la fin de chaque fonction permet de préciser que des fonctions ne peuvent pas être implémentées.

Une interface ne peut pas être instanciée, c'est à dire que l'on ne peut pas créer d'objet issus de cette classe, mais on doit faire une classe concrète qui implémente le concept qu'on instanciera grâce à une Factory.

La Factory est un patron de conception utilisé en programmation orientée objet. Elle permet d'instancier des objets dont le type est dérivé d'un type abstrait (classe abstraite, interface). La classe exacte de l'objet n'est donc pas connue par l'appelant. De plus la Factory génère seulement des singleton, dont le but est de restreindre l'instanciation d'une classe à un seul objet et ainsi éviter les erreurs.

```
Physique::BatimentDataService* PhysiqueFactory::batimentDataSrv = 0;
Physique::LocalisationDataService* PhysiqueFactory::localisationDataSrv = 0;
Physique::MaterielDataService* PhysiqueFactory::materielDataSrv = 0;
Physique::MetrologueDataService* PhysiqueFactory::metrologueDataSrv = 0;
Physique::PortiqueDataService* PhysiqueFactory::portiqueDataSrv = 0;
Physique::SalleDataService* PhysiqueFactory::salleDataSrv = 0;
Physique::TypeDataService* PhysiqueFactory::typeDataSrv = 0;
Physique::LecteurRFIDDataService* PhysiqueFactory::lecteurRFIDDataSrv = 0;

Physique::MaterielDataService* PhysiqueFactory::getMaterielDataServiceImpl() {
    if(materielDataSrv == 0) {
        materielDataSrv = new Physique::MaterielDataServiceImpl();
    }
    return materielDataSrv;
}
```


2.2) Réalisation sous Qt

2.2.1) Test de la couche physique

La réalisation de la couche physique représente la partie la plus importante de toutes mes parties. J'ai d'ailleurs réalisé un projet à part qui me permettait de tester toutes les méthodes de mes classes physiques. Avant de voir l'IHM de test je vais vous présenter la classe me permettant la connexion à la base de données, les 5 principales méthodes de `MaterielDataServiceImpl` et différentes requêtes (toujours les classes de Qt, la syntaxe pour le code java diffère un peu mais l'idée reste la même).

Connexion à la base de données:

```
ServiceSQL::ServiceSQL()
{
    db = QSqlDatabase::addDatabase("QMYSQL"); // Type de base de donnée
    db.setDatabaseName("localisationmateriel"); // Nom de la base de donnée
    db.setUserName( "root" ); // Nom d'utilisateur
    db.setPassword("cobra"); // Mot de passe
    db.setHostName("localhost"); // Nom du domaine
    db.setPort(3306); // Port (par défaut: 3306)
    if (db.open()) {
        qWarning("Connexion Database OK");
    }else{
        QMessageBox::critical(0, QObject::tr("Database Error"), db.lastError().text());
    }
}

ServiceSQL::~ServiceSQL() {
    if(db.isOpen()){
        db.commit();
        db.close();
    }
    qWarning("Fermeture et destruction du service SQL\n");
}

bool ServiceSQL::databaseIsOpen()
{
    return db.isOpen();
}
```

Le constructeur permet de remplir les informations nécessaires à la connexion de la base de données. Si la connexion a échoué, l'utilisateur sera prévenu par une boîte de dialogue. Le destructeur permet de fermer cette connexion si elle est ouverte. Avant de la fermer il s'assure que toutes les modifications sur la base de données seront prises en compte avant la fermeture de la connexion en validant la transition. Cette classe possède une méthode publique qui permet de savoir si la base de données est ouverte.

Méthode Add :

```
Materiel Physique::MaterielDataServiceImpl::add(Materiel materiel)
{
    Materiel newMateriel;
    query.prepare("INSERT INTO materiel (idBadge, dateControle, newDateControle, nomMateriel, idType) "
        "VALUES (:idBadge, :dateControle, :newDateControle, :nomMateriel, :idType)");
    query.bindValue(":idBadge", materiel.getIdBadge());
    query.bindValue(":dateControle", materiel.getDateControle());
    query.bindValue(":newDateControle", materiel.getNewDateControle());
    query.bindValue(":nomMateriel", materiel.getNomMateriel());
    query.bindValue(":idType", (qlonglong)materiel.getType().getId());
    if(query.exec()) {
        newMateriel = getById(query.lastInsertId().toLongLong());
    }else{
        qWarning("Error: %s\n", query.lastError().text().toLatin1().data());
    }
    return newMateriel;
}
```

Les requêtes utilisées sont des requêtes préparées. Leurs avantages est d'éviter les injections SQL puisqu'elles effectuent des vérifications portant sur les variables transmises avant d'exécuter la requête. Le deuxième avantage qui n'est pas présent sur cette classe mais qui est présent sur d'autre, c'est qu'elles sont plus rapide pour les requête exécutées plusieurs fois dans le script.

Les paramètres dans la requête sont représentés comme ceci :param. La requête permet d'insérer dans la table matériel une valeur dans le champ idBadge, dateControle, nomMateriel et idType. On exécute la requête et on récupère l'id créé par la base de données (auto-incrémente). On récupère ensuite le matériel ajouté dans la base de données grâce a cet id. Les résultats retournés sont des QVariant, type très puissant permettant de stocker n'importe quel type de variable.

Méthode Remove :

```
bool Physique::MaterielDataServiceImpl::remove(Materiel materiel)
{
    bool execute;
    query.prepare("DELETE FROM materiel WHERE id = :id");
    query.bindValue(":id", (qlonglong)materiel.getId());
    if(!(execute = query.exec())) {
        qWarning("Error: %s\n", query.lastError().text().toLatin1().data());
    }
    return execute;
}
```

Cette requête permet de supprimer un matériel selon son id, qui est la clé primaire de la table, donc le champ qui permet d'identifier les lignes de la table.

Méthode Update :

```

bool Physique::MaterielDataServiceImpl::update(Materiel materiel)
{
    bool execute;
    query.prepare("UPDATE materiel SET idBadge = :idBadge, dateControle = :dateControle, newDateControle = "
        ":newDateControle, nomMateriel = :nomMateriel, idType = :idType WHERE id = :id");
    query.bindValue(":id", (qulonglong)materiel.getId());
    query.bindValue(":idBadge", materiel.getIdBadge());
    query.bindValue(":dateControle", materiel.getDateControle());
    query.bindValue(":newDateControle", materiel.getNewDateControle());
    query.bindValue(":nomMateriel", materiel.getNomMateriel());
    query.bindValue(":idType", (qulonglong)materiel.getType().getId());
    if(!(execute = query.exec())) {
        qWarning("Error: %s\n", query.lastError().text().toLatin1().data());
    }
    return execute;
}

```

La requête permet de modifier les champs concernés par le set d'un matériel identifié selon son id.

Méthode getAll :

```

QList<Materiel> Physique::MaterielDataServiceImpl::getAll(int debut, int parPage)
{
    QList<Materiel> materiels;
    Materiel materiel;
    query.prepare("SELECT * FROM materiel ORDER BY nomMateriel ASC LIMIT :debut,:parPage");
    query.bindValue(":debut", debut);
    query.bindValue(":parPage", parPage);
    if(query.exec()) {
        while(query.next())
        {
            materiel = Materiel(query.value(1).toString(), query.value(2).toDate(), query.value(3).toDate(),
                query.value(4).toString(), typeDataSrv->getById(query.value(5).toLongLong()));
            materiel.setId(query.value(0).toLongLong());
            materiels.append(materiel);
        }
    }else{
        qWarning("Error: %s\n", query.lastError().text().toLatin1().data());
    }
    return materiels;
}

```

La requêtes permet de récupérer toutes les informations de n matériel(s) (définie par parPage) à partir de l'élément voulu (le premier élément est à l'index 0). Tant qu'il y a des résultats (un résultat est composé de tous les éléments d'un matériel) on extrait les informations correspondantes (le premier élément commence lui aussi à l'index 0).

Méthode getCount :

```

int Physique::MaterielDataServiceImpl::getCount()
{
    int count = 0;
    query.prepare("SELECT COUNT(*) FROM materiel");
    if(query.exec()) {
        if(query.next()){
            count = query.value(0).toInt();
        }
    }else{
        qWarning("Error: %s\n", query.lastError().text().toLatin1().data());
    }
    return count;
}

```

La requête permet de récupérer le nombre de matériel correspondant au nombre de lignes présentent dans la base de données. Préciser une étoile dans le count permet d'être sûr de prendre en compte toutes les ligne de la table. Si on avait passé un champ de la table au lieu de l'étoile, seul les champs non nuls aurait été pris en compte.

GetBySalle :

```
"SELECT * FROM portique WHERE idSalle1 = :idSalle1 OR idSalle2 = :idSalle2 "+  
"ORDER BY id ASC LIMIT :debut,:parPage");
```

Cette requête permet de récupérer tous les portiques qui possède l'id de la salle dans un des deux champs.

GetListSalle :

```
"SELECT * FROM salle WHERE id IN (SELECT DISTINCT idSalle1 FROM portique UNION "  
+"SELECT idSalle2 FROM portique) ORDER BY numSalle ASC LIMIT :debut,:parPage");
```

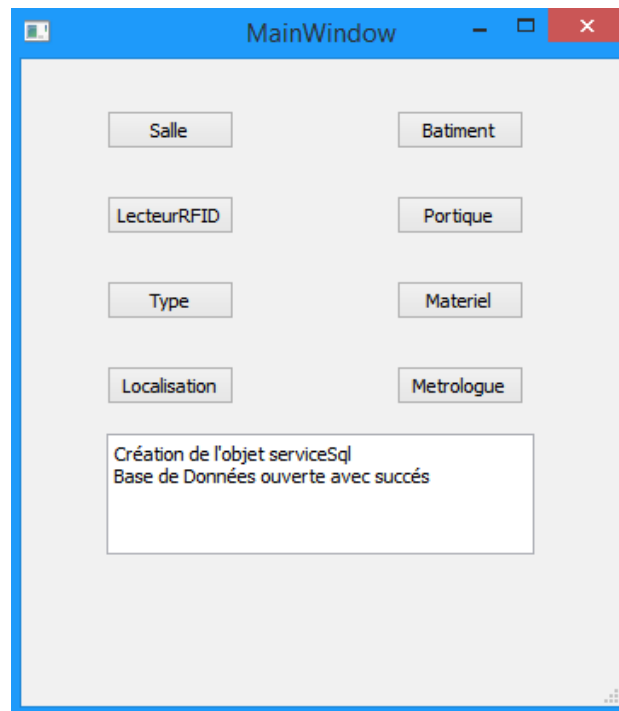
Cette requête permet de récupérer toutes les salles dont l'id correspond à un des id présent dans les 2 champs de portique tout en évitant les redondances.

La classe d'implémentation de la classe de service bâtiment de la couche physique était différente des autres à gérer. En effet, un bâtiment contient une liste de salles et ceux ci est impossible a représenter sans table d'association. Cette classe d'association doit être invisible pour les autres parties du projet. A chaque fois que quelqu'un voulais récupérer un ou une liste de bâtiment, il fallait que je récupère séparément la liste de salle de chaque bâtiment grâce à la table attribution. En cas d'ajout mais surtout de modifications, il faut savoir quel opération effectuée sur cette table attribution :

- Ajout/Modification : Les salles ne sont pas déjà attribuées, on peut donc les rajouter.
- Modification : Les salles sont déjà attribuées, il faut vérifier si ces salles sont attribuées à notre bâtiment, si oui on supprime ces salles de la liste des salles déjà attribuées à notre bâtiment. Les éléments restant de la liste des salles déjà attribuées sont donc des salles qui ne sont plus attribuées à notre bâtiment, on supprime donc ces attributions.

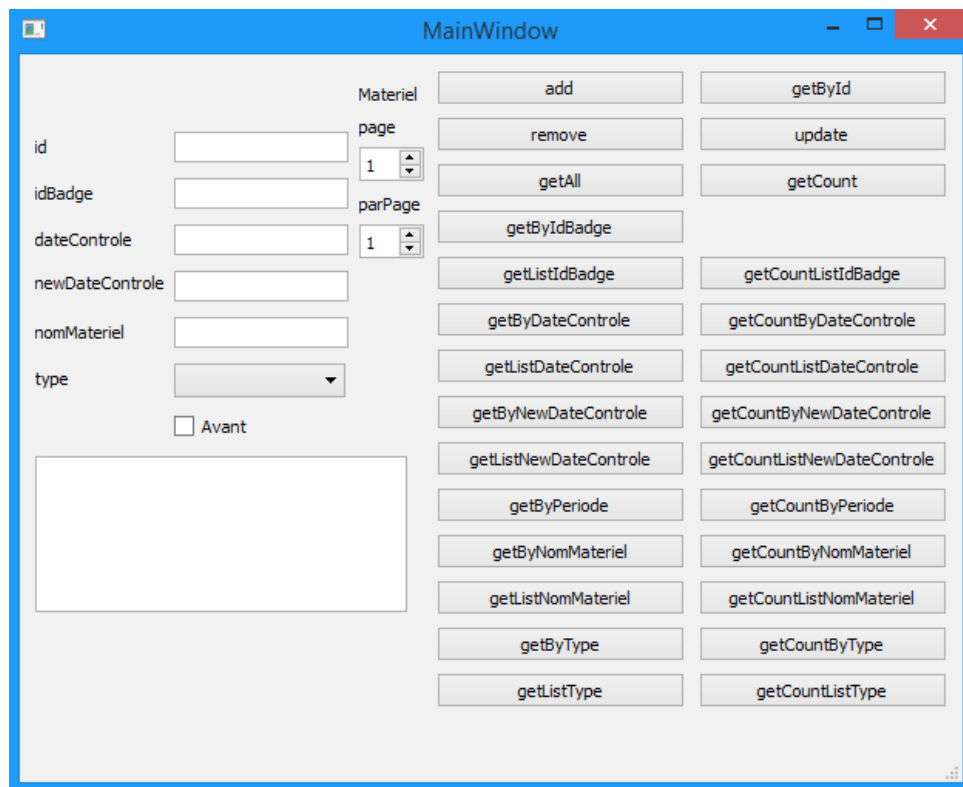
Lorsqu'un un bâtiment ou une salle est détruite, on supprime la ou les attributions liées.

Voici la première partie de l'IHM me permettant de choisir quelle classe je veux tester

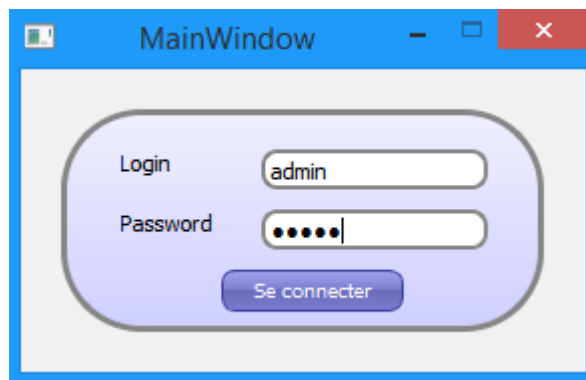


Et celle qui me permet de tester les différentes méthodes de cette classe. Elle sont légèrement différentes selon les classes.

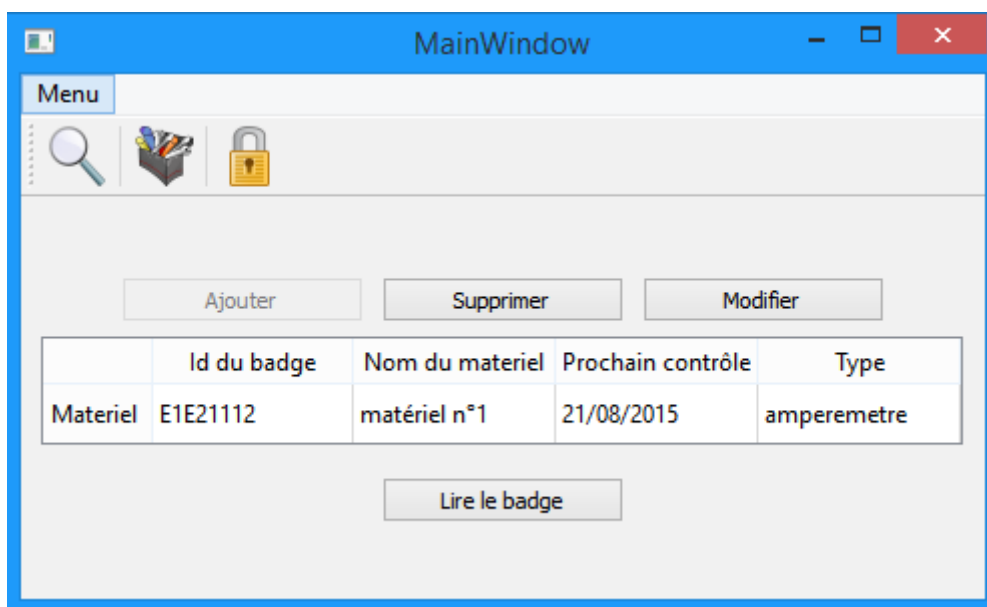
On y retrouve tous les attributs de la classes à gauche, la pagination au milieu, les méthodes à droite et en bas à gauche une zone de texte pour afficher les différents résultats.



2.2.2) Projet de gestion de matériel



La première IHM sur laquelle on arrive est celle de connexion afin d'éviter que n'importe qui modifie la base de données ou écrit sur les badges grâce aux différentes applications situées dans le poste de métrologie.



Après connexion, l'utilisateur arrive sur cette IHM et il peut naviguer entre les différentes pages de celle-ci grâce à la barre d'outil.

La première page, celle où l'on arrive après la connexion, permet de voir les informations d'un matériel selon l'id du badge que l'on lit grâce à un lecteur RFID. Selon si le matériel existe ou non, le bouton Ajouter ou les boutons Supprimer et Modifier seront disponibles. Si le badge n'a pas pu être lu, aucun de ces boutons ne seront disponibles.

Si l'id du badge que l'on récupère lors de la lecture du badge existe dans la base de données, on remplit les informations supplémentaires correspondant au matériel en plus de celle-ci. De plus si sa date de contrôle est dépassée, elle sera affichée en rouge.

```
// Recherche le materiel si l'id du badge a été lu et active les boutons voulus selon si la materiel existe ou non
void MenuWindow::validerMenu()
{
    QString idBadge = ui->tableWidget->item(0,0)->text();
    ui->tableWidget->clearContents(); // Supprime le contenu de la table excepté les en-tête
    Materiel materiel = materielSrv->getByIdBadge(idBadge);
    long id = materiel.getId(); // On récupère l'id dans une variable qui permettra d'identifier le matériel
    ui->tableWidget->setItem(0, 0, new QTableWidgetItem(idBadge));
    if(id > 0){ // Si le matériel existe, on ajoute les infos récupérées de la bdd dans la table
        // et on permet à l'utilisateur de supprimer ou modifier ce matériel
        ui->tableWidget->setItem(0, 1, new QTableWidgetItem(materiel.getNomMateriel()));
        ui->tableWidget->setItem(0, 2, new QTableWidgetItem(materiel.getNewDateControle().toString("dd/MM/yyyy")));
        ui->tableWidget->setItem(0, 3, new QTableWidgetItem(materiel.getType().getNomType()));
        if(QDate::currentDate() >= materiel.getNewDateControle()){
            ui->tableWidget->item(0, 2)->setTextColor(Qt::red);
        }
        ui->pushButtonAjouter_Menu->setEnabled(false);
        ui->pushButtonModifier_Menu->setEnabled(true);
        ui->pushButtonSupprimer_Menu->setEnabled(true);
    }else{ // Sinon on l'autorise à l'ajouter si l'id du badge a pu être lu.
        if(idBadge.isEmpty()){
            ui->pushButtonAjouter_Menu->setEnabled(false);
        }else{
            ui->pushButtonAjouter_Menu->setEnabled(true);
        }
        ui->pushButtonModifier_Menu->setEnabled(false);
        ui->pushButtonSupprimer_Menu->setEnabled(false);
    }
}
```

Dans notre cas, lors de la lecture du badge, le matériel existait déjà dans la base de données et l'on propose donc à l'utilisateur de modifier ou supprimer ce matériel.

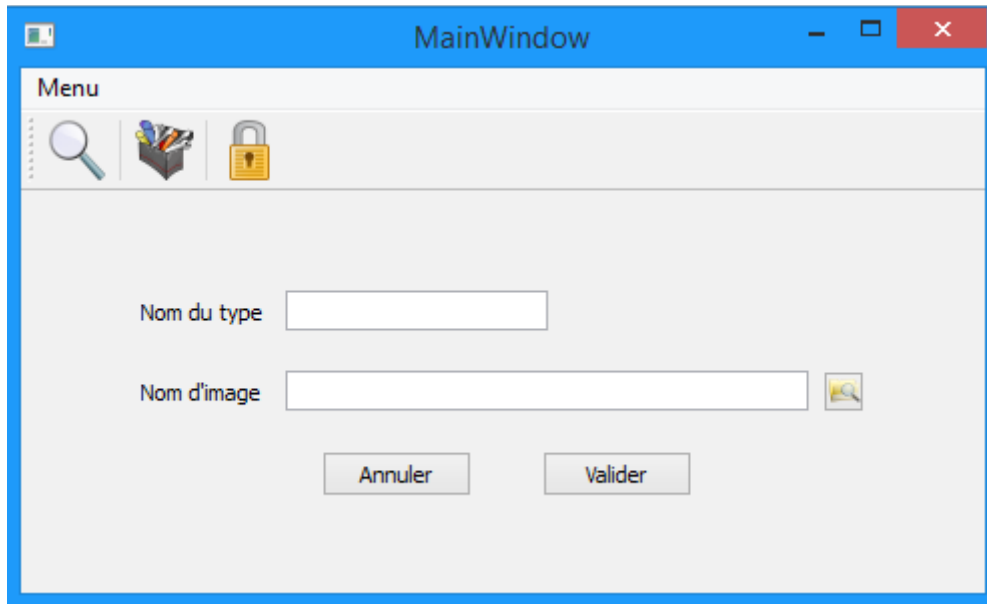
The screenshot shows a Qt-based application window titled "MainWindow". Inside, there's a sub-window titled "Menu". At the top of the "Menu" window is a toolbar with three icons: a magnifying glass, a toolbox, and a padlock. Below the toolbar are several input fields and controls:

- "Id du badge:" followed by a text box containing "E1E21112".
- "Date du dernier contrôle:" followed by a date picker showing "07/05/2015".
- "Nombre de jour entre 2 contrôles:" followed by a spin box set to "120".
- "Nom du materiel:" followed by a text box containing "matériel n°1".
- "Type:" followed by a dropdown menu showing "amperemetre". To the right of the dropdown is a blue hyperlink that says "Besoin de rajouter un type?".

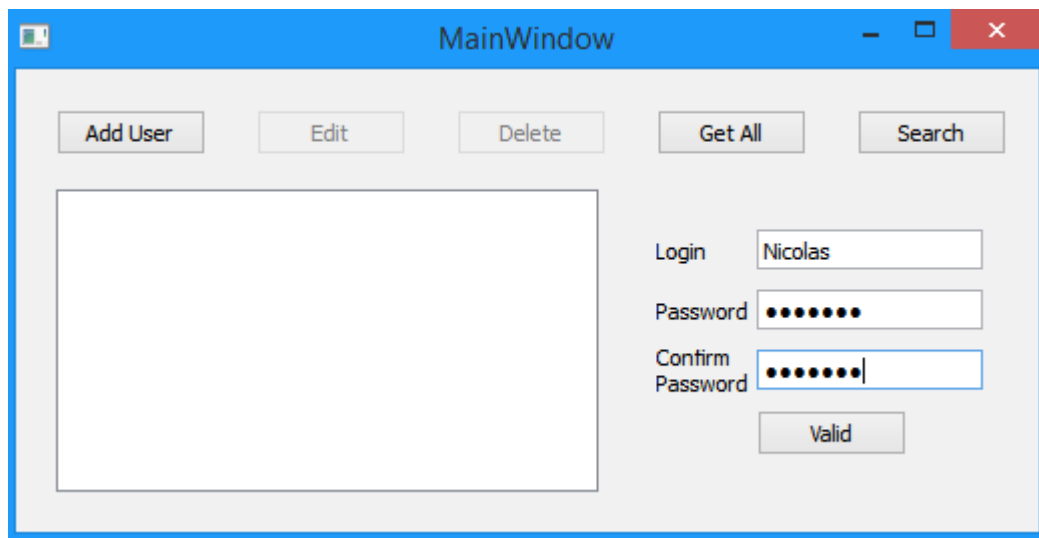
At the bottom of the "Menu" window, there are two buttons: "Reset" and "Modifier".

Cette page est celle affichée lorsque l'on veut modifier le matériel recherché auparavant. Les champs sont pré-remplis, il suffit juste de modifier ceux qui nous intéressent. Les informations seront changées dans la base de données et seront écrites sur le badge. Si l'écriture de badge échoue, les informations du matériels seront remises à leurs valeurs d'apparavant dans la base de données. Pour l'ajout, la page est la même sauf que le bouton Modifier sera remplacer par le bouton Ajouter. Si un matériel a été recherché sans succès, donc à ajouter, le champs pour l'id du badge sera comme même pré-remplis afin d' éviter à l'utilisateur de l'écrire.

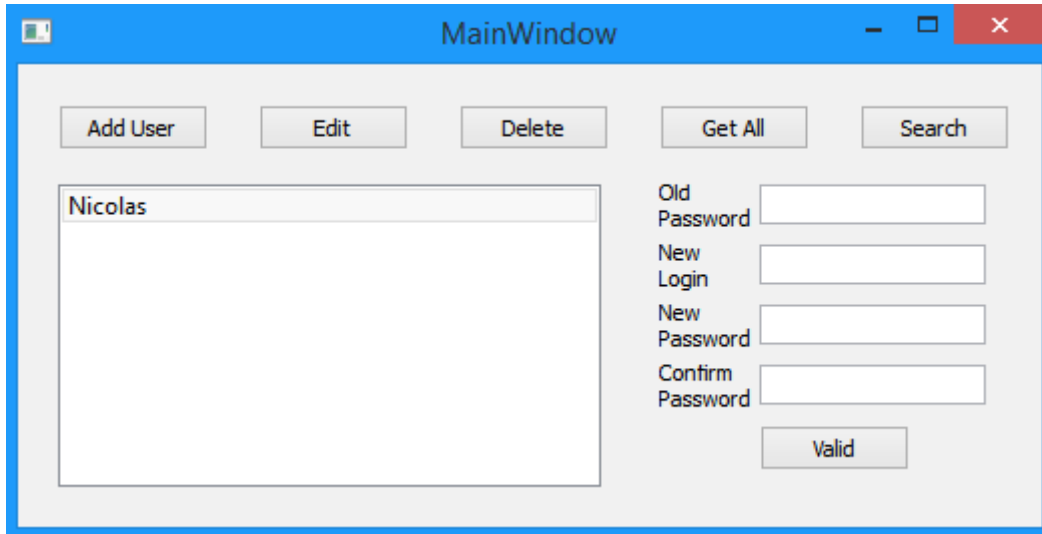
Un lien est présent en plus de la barre d'outil pour se rendre à la page permettant d'ajouter un type.



Cette page permet de rajouter un type. Le champ du nom de l'image ne peut pas être édité directement, la petite icône à côté permet de rechercher un fichier image valide. Lors de la validation deux cas peuvent se produire, soit le nom de type n'existe pas et on ajoute le type soit il existe et l'on demande à l'utilisateur s'il veut annuler l'opération, modifier ou supprimer le type.



Depuis la barre d'outil, on peut accéder à la gestion d'administrateur. Cette IHM permet d'ajouter, modifier, supprimer, voir les différents utilisateurs ou rechercher un utilisateur précis. On se trouve actuellement sur la page d'ajout d'utilisateur. Le compte est ajouté seulement si le champ password et confirm password sont identiques. Pourquoi ajouter un utilisateur alors qu'un seul compte aurait suffi ? Imaginez qu'il y est plusieurs métrologues, chacun aurait pu avoir le même compte mais si jamais un de ces métrologues quitte son poste, pour des raisons de sécurité il aurait fallu changer le mot de passe et tous les autres métrologues auraient du apprendre le nouveau mot de passe. Alors que dans notre cas chaque métrologue a son propre compte et s'il quitte son poste, il suffira de supprimer son compte et cela n'aura pas d'incidence sur les autres.



The screenshot shows a Windows application window titled "MainWindow". At the top, there are five buttons: "Add User", "Edit", "Delete", "Get All", and "Search". Below these buttons is a list box containing the name "Nicolas". To the right of the list box, there are four text input fields labeled "Old Password", "New Login", "New Password", and "Confirm Password". A "Valid" button is located below the "Confirm Password" field.

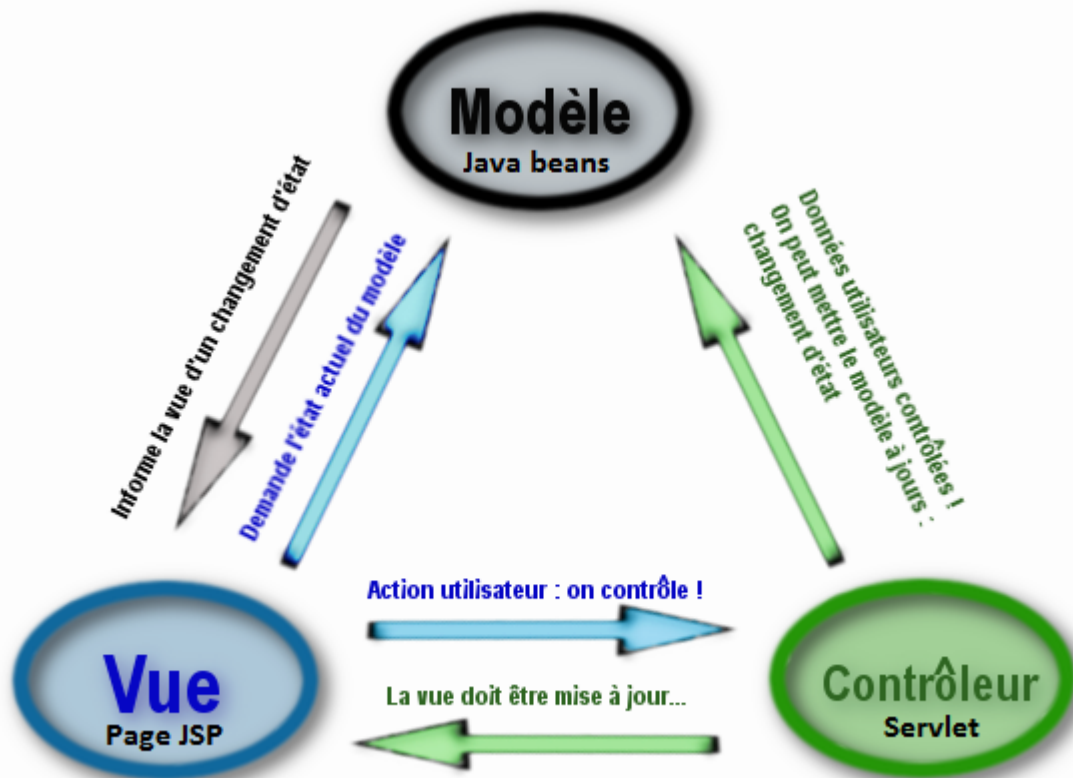
Nous voici maintenant dans la page de modification d'utilisateur. Il faut savoir que cette page et celle de suppression d'utilisateur ne sont accessibles que si un utilisateur à été sélectionné dans la liste. La page de modification demande l'ancien mot de passe pour que seulement la personne qui est propriétaire de ce compte puisse le modifier, on y retrouve ensuite les même champs que ceux de l'ajout.

2.3) Réalisation sous Netbeans

2.3.1) Patron de conception MVC

Les patrons de conceptions décrivent une structure à respecter pour un certain nombre de problèmes récurrents en programmation orientée objet.

Afin de structurer au mieux mon site web, j'ai respecté le design pattern MVC.



Modèle: cette partie gère les données du site. Son rôle est d'aller récupérer les informations «brutes» dans la base de données, de les organiser et de les assembler pour qu'elles puissent ensuite être traitées par le contrôleur.

Le modèle correspond aux java beans. Chaque java bean représente une classe du modèle. Cette classe est caractérisée par des attributs (propriétés et attributs d'associations avec les autres classes) qui sont souvent privés, les accesseurs (get..), mutateurs (set..) et des méthodes métiers qui assurent les différents traitements du métier et la persistance de ces objets (souvent dans une base de données relationnelle).

Vue : cette partie se concentre sur l'affichage. Elle ne fait presque aucun calcul et se contente de récupérer des variables pour savoir ce qu'elle doit afficher. On y trouve essentiellement du code HTML mais aussi quelques boucles et conditions JSTL très simples. Sa seconde tâche est de recevoir toute action de l'utilisateur (hover, clic de souris, sélection d'un bouton radio, cochage d'une case, entrée de texte, etc.) et de renseigner le contrôleur.

Les JavaServer Pages ou JSP jouent le rôle de la vue. Le JavaServer Pages ou JSP est une technique basée sur Java qui permet aux développeurs de créer dynamiquement du code HTML, XML ou tout autre type de page web. Il permet de déclarer l'utilisation d'une bibliothèque de balise comme JSTL.

Contrôleur : Le contrôleur prend en charge la gestion des événements de synchronisation pour mettre à jour la vue ou le modèle et les synchroniser. C'est en quelque sorte l'intermédiaire entre le modèle et la vue. Le contrôleur va demander au modèle les données, les analyser, prendre des décisions et renvoyer le texte à afficher à la vue.

Les servlets effectuent le rôle de contrôleurs. Une servlet est une classe Java qui permet de créer dynamiquement des données au sein d'un serveur HTTP. Elle est développée dans un contexte client-serveur, et vient se greffer sur des applications existantes afin de les étendre ou de les implémenter différemment. Le but d'une servlet est de pouvoir envoyer et recevoir une requête HTTP. Pour exécuter les servlets, on doit utiliser un serveur appelé conteneur de servlets. Pour ce projet, j'ai utilisé Tomcat qui est le plus connu.

2.3.2) Différentes balises JSTL

La librairie JSTL 1.2.2 de java ajoute différentes bibliothèques de balise.

Bibliothèque	URI	Prefixe	Fonction
core	http://java.sun.com/jsp/jstl/core	c	Contient les principales balises JSTL.
Format	http://java.sun.com/jsp/jstl/fmt	fmt	Permet de formater des dates, temps et nombres.
XML	http://java.sun.com/jsp/jstl/xml	x	Permet de créer et manipuler des documents XML.
SQL	http://java.sun.com/jsp/jstl/sql	sql	Permet d'interagir avec la base de données.
Function	http://java.sun.com/jsp/jstl/functions	fn	Contient des fonctions standards comme des manipulations de chaîné.

Malgré que j'ai utilisé la bibliothèque Format pour les dates, la bibliothèque que j'ai le plus utilisé est la core, je vais donc présenter les différentes balises dont elle est constituée.

Balise	Description
<c:out >	Permet de faire afficher.
<c:set >	Permet d'inclure une valeur dans une variable.
<c:remove >	Supprimer une variable.
<c:catch>	Intercepte toutes les erreurs produites.
<c:if>	Vérifie si la condition est respectée.
<c:choose>	Permet de traiter différent cas.
<c:when>	Sous tag de <c:choose>. Vérifie si la condition est respectée.
<c:otherwise >	Sous tag de <c:choose>. Définit un cas si aucune condition n'a été respectée.
<c:import>	Affiche le contenu d'une ressource passé grâce à son URL dans la page actuelle. Contrairement à <jsp:include>, la ressource peut être hébergée sur un autre serveur.
<c:forEach >	Parcours chaque élément de la collection en exécutant le code à chaque passage.
<c:forTokens>	Permet de découper des chaînes de caractères selon un ou plusieurs délimiteurs. Chaque marqueur ainsi obtenu sera traité dans une boucle d'itération.
<c:param>	Ajoute un paramètre à l'URL.
<c:redirect >	Redirige vers une nouvelle URL.
<c:url>	Crée une URL.

2.3.3) Template

Un template est un patron de mise en page où l'on place images et textes. Il est souvent utilisé de manière répétitive pour créer des documents présentant une même structure. Il permet donc d'éviter de recopier le code pour chaque page voulu.

```
<%@ page pageEncoding="UTF-8" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html>
<html>
  <head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <link rel="stylesheet" href="Template/Default/default.css" type="text/css">
    <link rel="icon" type="image/png" href="Image/icon.png">
    <title>Localisation de matériel</title>
  </head>
  <body>
    <div id="en_tete">
      <jsp:include page="/Template/Default/menu.jsp" flush="false"/>
    </div>
    <div id="corps">
      <c:choose>
        <c:when test="${view != null}">
          <jsp:include page="${view}" flush="false"/>
        </c:when>
        <c:otherwise>
          <jsp:include page="/accueil.jsp" flush="false"/>
        </c:otherwise>
      </c:choose>
    </div>
    <div id="pied de page">
      <...34 lines />
    </div>
  </body>
</html>
```

Le template est définie par 3 grandes parties représentées par des div. La première partie concerne l'en-tête. Elle va contenir le menu du site que j'inclue grâce à la balise jsp. Le paramètre flush peut prendre 2 valeur, si true alors la réponse générée par la page appelante est effacée, sinon la réponse de la ressource cible est fusionnée à la réponse de la page appelante. La deuxième partie est celle qui va afficher le contenu de la page demandé grâce à son URI. Si aucune page n'est demandée, on affiche par défaut la page d'accueil. Pour finir la dernière partie permet d'afficher la pagination si besoin.

2.3.4) Site web de localisation de matériel

Comme pour le projet Qt, le projet Java comporte toutes les classes entités, interfaces, implémentations et les Factory présentent dans les packages métier et physique qu'il a fallu recoder « à la sauce Java ».

Le site web est composé d'un menu permettant de faire les différentes recherches et d'un lien pour retourner vers l'accueil. Lorsque un utilisateur effectue une recherche, la localisation des matériels demandée s'affiche dans un tableau. Le tableau peut contenir 10 matériels au maximum, si le nombre de matériels recherché excède cette valeur, une pagination s'affiche en dessous pour pouvoir consulter tous les autres matériels. Les matériels qui ont besoin d'un nouveau contrôle voient leurs dates affichées en rouge.

Accueil Voir tous le matériel Matériel à contrôler Rechercher par numéro Rechercher par date Rechercher par nom Rechercher par type							
Information sur l'emplacement				Information sur le matériel			
Nom du bâtiment	Numéro de salle	Nom de la salle	Nom de l'atelier	N° du matériel	Période de validation	Nom du matériel	Type de matériel
bâtiment 1	101	Salle 101	Français	EE23EG23D	du 07/04/2015 au 07/04/2016	matériel 1	oscilloscope
bâtiment 4	433	Cours info	Informatique	K2F0LLOK009POP	du 03/07/2012 au 03/07/2013	matériel 10	voltmètre
bâtiment 4	435	Tp info n°1	Informatique	F23F89UFGFG4F	du 03/07/2011 au 03/07/2012	matériel 11	ampèremètre
bâtiment 4	436	Tp info n°2	Informatique	VVJR23BEBVN	du 21/03/2015 au 21/03/2016	matériel 12	ampèremètre
bâtiment 4	436	Tp info n°2	Informatique	EJEFH243B34F	du 28/04/2015 au 28/05/2015	matériel 13	oscilloscope
bâtiment 4	436	Tp info n°2	Informatique	VNURHV34VNE	du 14/03/2015 au 14/03/2016	matériel 14	voltmètre
bâtiment 4	455	Cours physique	Physique	F23J82HFU23FH8	du 12/12/2007 au 12/12/2008	matériel 15	oscilloscope
bâtiment 4	453	Tp physique	Physique	DK32JFFHU3H28	du 05/06/2000 au 05/06/2001	matériel 16	oscilloscope
bâtiment 4	453	Tp physique	Physique	2JCHC72HF82F	du 03/01/2015 au 03/01/2016	matériel 17	ampèremètre
bâtiment 4	453	Tp physique	Physique	F23FJ98H7F2U3FI	du 26/01/2015 au 26/02/2015	matériel 18	voltmètre

<< < 1 2 3 > >>

Page : 1 / 3

On peut choisir la page voulue depuis la barre de pagination ou depuis le champ de saisie dédié.

La barre de pagination affichera seulement 10 pages au maximum. Si il y a plus de 10 pages, selon où se trouve l'utilisateur les pages seront affichées différemment. Dans la mesure du possible il affichera les 4 pages d'avant, la page actuelle et les 5 d'après. S'il ne peut pas le faire un calcul sera réalisé pour afficher plus de page à droite ou à gauche.

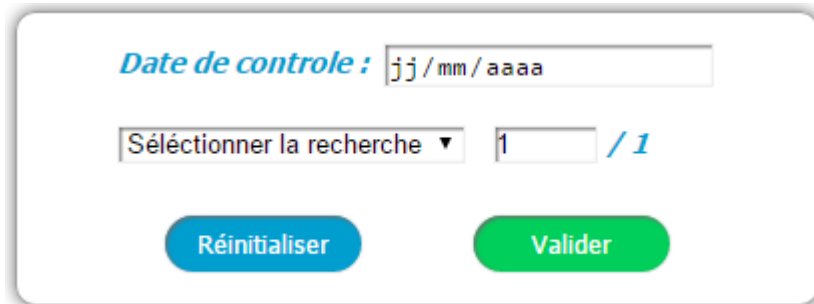
<< < 1 2 3 4 5 6 7 8 9 10 > >>

<< < 7 8 9 10 11 12 13 14 15 16 > >>

<< < 13 14 15 16 17 18 19 20 21 22 > >>

Le signe supérieur et inférieur permet de se déplacer de page en page. Le double supérieur ou le double inférieur permet de se rendre à la première ou dernière page.

Le but du site web est de pouvoir localiser un ou des matériel(s) selon le numéro, la date de contrôle, le nom ou le type.



Le formulaire de recherche comporte 3 champs. Le premier est la zone de recherche que l'on peut remplir manuellement. Le deuxième champ liste tous les éléments existants dans la base de données. Sélectionner un de ces éléments permet de remplir automatiquement le premier champ et ainsi éviter de le taper. Le troisième champs permet de paginer cette liste. En effet une cellule de métrologies peut contenir plusieurs milliers d'appareils, les lister en une seule fois peut ralentir le site et n'est pas forcément très pratique pour l'utilisateur. Le remplissage de la liste déroulante se fait en Ajax depuis des informations envoyées en JSON par une Servlet. La librairie GSON trouvable sur [internet](#) permet de convertir des objets java en JSON.

```
json = new Gson().toJson(listElement);  
response.setContentType("application/json");  
response.setCharacterEncoding("UTF-8");  
response.getWriter().write(json);
```

Après validation du formulaire les données seront traitées par une servlet. Cette servlet va gérer le fonctionnement de la pagination mais surtout gérer les matériels à afficher. Selon le type de recherche que l'utilisateur a effectué, la servlet va demander au modèle la liste des matériels qui corresponde à la demande de l'utilisateur tout en effectuant des tests auparavant comme pour l'existence de la date. En effet, par défaut lorsqu'une chaîne de caractère va être convertie en date, Java va essayer de l'interpréter même si cette date n'est pas valide. La servlet fait donc appelle à une autre classe qui va convertir la chaîne de caractère en date tout en forçant une erreur si cette date n'est pas valide. Par exemple le dernier jour du mois de janvier est le 30, le 31 janvier ne sera pas considéré comme le 1 février mais comme une erreur. La servlet retournera ensuite la liste des matériels qui devra être affichée par vue.

```
if(attribut.equals("all")){
    // addAll permet d'ajouter tous les éléments d'une collection à notre liste actuelle.
    materiels.addAll(materielSrv.getAll(debut, parPage));
} else if (attribut.equals("controleAFaire")) {
    materiels.addAll(materielSrv.getByPeriode(date, true, debut, parPage));
} else if (attribut.equals("idBadge")) {
    Materiel materiel = materielSrv.getIdBadge(recherche);
    if (materiel != null) {
        // add permet d'ajouter un élément non null à notre liste actuelle
        materiels.add(materiel);
    }
} else if (attribut.equals("dateControle")) {
    Date dateControle;
    dateControle = DateValidator.validParse(recherche, dateFormatEn);
    if (dateControle != null) {
        materiels.addAll(materielSrv.getByDateControle(dateControle, debut, parPage));
    } else {
        dateControle = DateValidator.validParse(recherche, dateFormatFr);
        if (dateControle != null) {
            materiels.addAll(materielSrv.getByDateControle(dateControle, debut, parPage));
        }
    }
} else if (attribut.equals("nomMateriel")) {
    materiels.addAll(materielSrv.getByNomMateriel(recherche, debut, parPage));
} else if (attribut.equals("nomType")) {
    Type type = typeService.getByNomType(recherche);
    materiels.addAll(materielSrv.getByType(type, debut, parPage));
}
```

Afin que le site web soit compatible avec tous les navigateurs, il était nécessaire de gérer les différents cas lorsqu'un moteur de recherche ne supporte pas un élément. Le champ de recherche pour la date de contrôle est un input date, ce type d'input n'est pas supporté par tous les navigateurs et sera transformé en input text par ce là. La fonction Javascript pour transférer le texte sélectionné dans la liste déroulante au champ de recherche diffère donc.


```
function TransferTextToText() {  
    var selectText = document.getElementById('selectElement').value;  
    document.getElementById('recherche').value = selectText;  
}  
  
function TransferTextToFrDateFormat(separateur) {  
    var type = document.getElementById('recherche').type;  
    if (type === "date") { // Si c'est un input date  
        var selectText = document.getElementById('selectElement').value;  
        // A chaque fois que le caractère rencontré correspond au séparateur  
        // le mot est éclaté et donne donc un tableau de sous-mot  
        var parts = selectText.split(separateur);  
        // Il est impossible d'insérer une date dans un input date  
        // si celui ne correspond pas au format yyyy-MM-dd  
        var date = parts[2] + "-" + parts[1] + "-" + parts[0];  
        document.getElementById('recherche').value = date;  
    } else { // Sinon c'est un input text  
        TransferTextToText();  
    }  
}
```

Lorsque le champ est une simple zone de texte, il suffit de récupérer le texte contenu dans la liste déroulante et de l'insérer dans l'autre champ. Mais lorsque le champ correspond à un champ date, on vérifie qu'il est supporté par le navigateur (donc qu'il est pas devenu un input text), on décompose le texte récupéré puis on le rassemble afin de respecter le format de la date à inclure dans l'input date.

Il est de même pour le fichier css, certaines propriétés css ne sont pas supporté de la même façon par les navigateurs, il faut pour cela rajouté un préfixe selon le navigateur.

- -o- pour Opera
- -moz- pour Mozilla
- -webkit- pour Webkit (Chrome, Safari, Android...)
- -ms- pour Microsoft (Internet Explorer)

3) Conclusion

Ce projet en groupe m'a permis dans un premier temps de toucher à différents langages et à apprendre à gérer le temps passé sur chacun de ces langages afin respecter le planning. Le projet fût aussi vraiment intéressant car il m'a permis de travailler en groupe.

Certaines améliorations ont été apportées au projet comme la sécurisation à l'aide d'un mot de passe crypté en MD5 des projets permettant la modification de la base de données ou de l'écriture des badges. Autre amélioration était de permettre au métrologue ou aux techniciens de voir quels matériels ont besoin d'être contrôlés.

D'autres améliorations sont encore possibles comme afficher le matériel dans un plan pour le site web.

Je compte poursuivre mes études en licence professionnelle dans le web et ce projet m'a beaucoup apporté que ce soit la partie web, base de données ou Qt.