BERNIER Benoît
BLANDEL Alexandre
DE CEITA Alex
STEPHANT Thomas

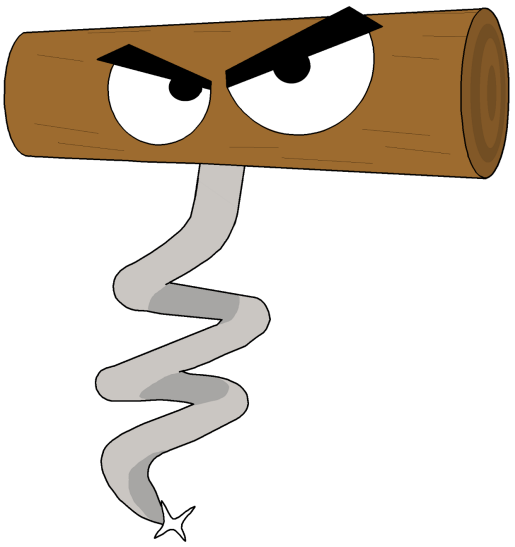
ECOLE SUPERIEURE
D'INGENIEURS DE RENNES

Délivrable workpackage 4:

Traitement de l'information

13 mars 2018

Groupe : IoTireBouchon



WorkPackage 4 : Traitement de l'information

Ce workpackage a pour but la réalisation de l'application qui va permettre de mettre en relations les informations captées par le drone. L'application sera développée en Java avec la bibliothèque graphique JavaFx, les données de vols seront écrites dans une base de données MySQL ou PostgreSQL.

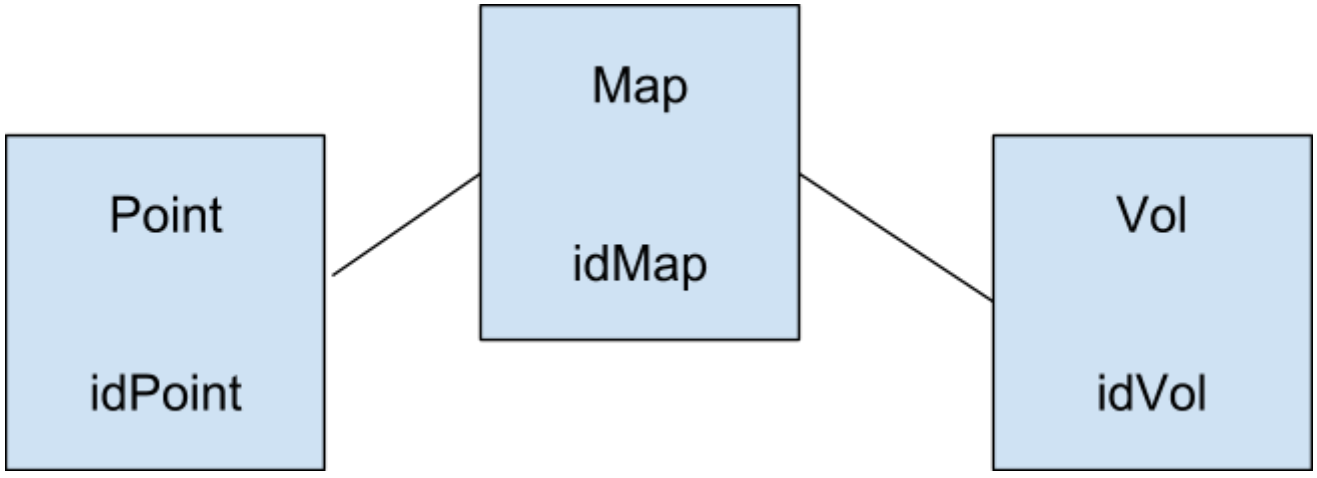
WorkPackage 4.1 : Stockage de l'information

Nous voulons sauvegarder les données captées par le drone et les envoyer au logiciel dans une base de données externe, nous permettant de traiter et récupérer les informations sur n'importe quel device du moment que celui-ci soit connecté à internet (l'application sera développée en java donc multiplateforme).

WorkPackage 4.1.1 : Conception du schéma de la base de données

Objectif : Le but de l'ingénieur est de réfléchir à la meilleure solution pour résoudre un problème, nous avons à coeur de réaliser une application la plus complète possible avec le temps imparti du projet. Nous voulons réaliser une base de données qui correspond à notre projet et qui sera ajustable en fonction des topographies réalisées. C'est pour ceci que la réflexion sur ce package est important pour optimiser le temps de développement du software.

Réalisation :



Une map à un vol
Une map à un ou plusieurs points
Un vol à une map
Un point à une map

WorkPackage 4.1.2 : Création de la base de données

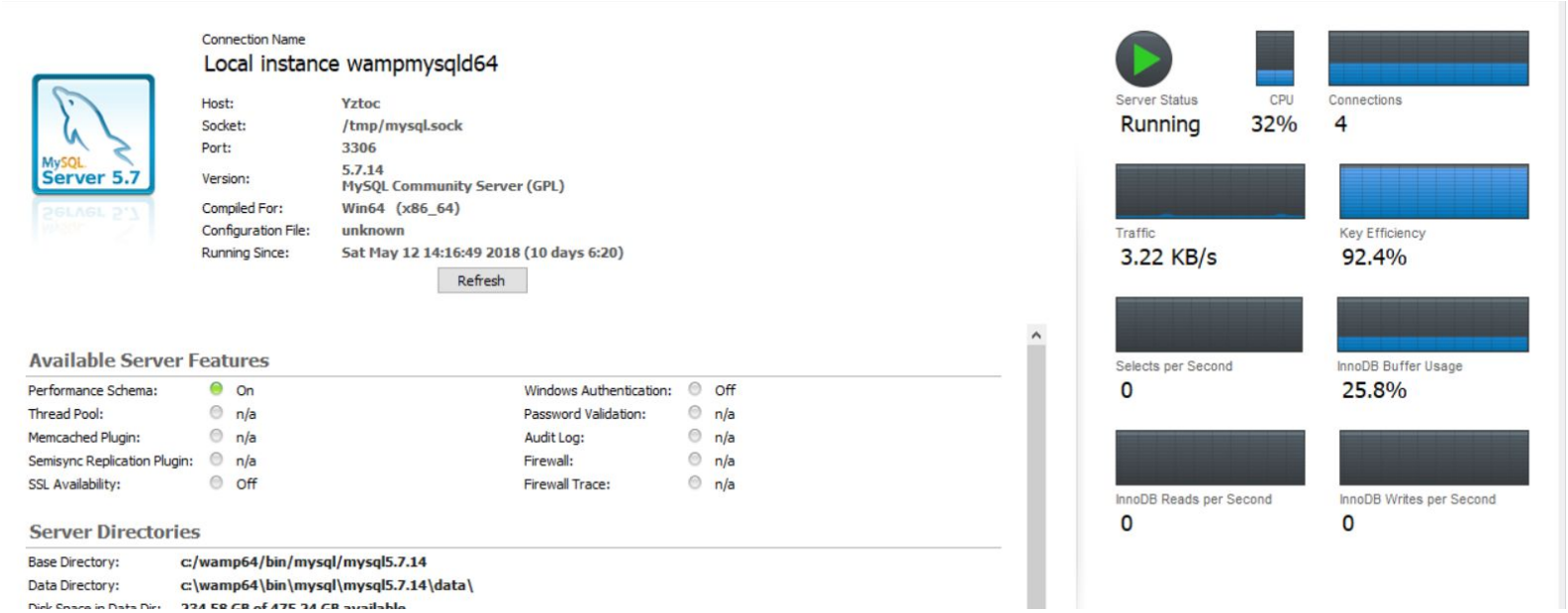
Objectif : Une fois le workpackage 4.1.1 réalisé et validé la réalisation de la base de données ne prendra que peu de temps sur le temps complet de développement de l'application.

Réalisation : Nous avons utilisé pour ce faire le SGBD MySQL avec l'interface graphique MySQL Workbench pour nous permettre de gagner du temps en développement.

WorkPackage 4.1.3 : Relier la base de données au logiciel

Objectif : Ce WorkPackage est relié avec le 4.2.1, une fois la structure de la base de données développée, nous hébergerons celle-ci sur un serveur connecté à internet ce qui permettra de récupérer les données de n'importe quel endroit sans que le drone soit avec nous par exemple.

Réalisation :Pour le moment nous développons sur un serveur local (Wamp) car le logiciel est en cours de développement et il se pourrait que le schéma puisse encore évoluer.



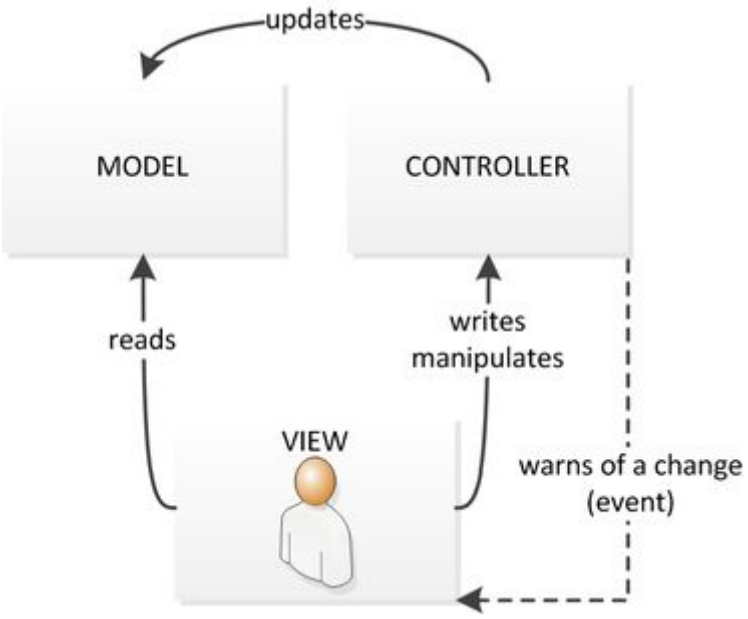
Quelques triggers déclenchés lors de la suppression d'éléments pour éviter de violer les clefs étrangères :

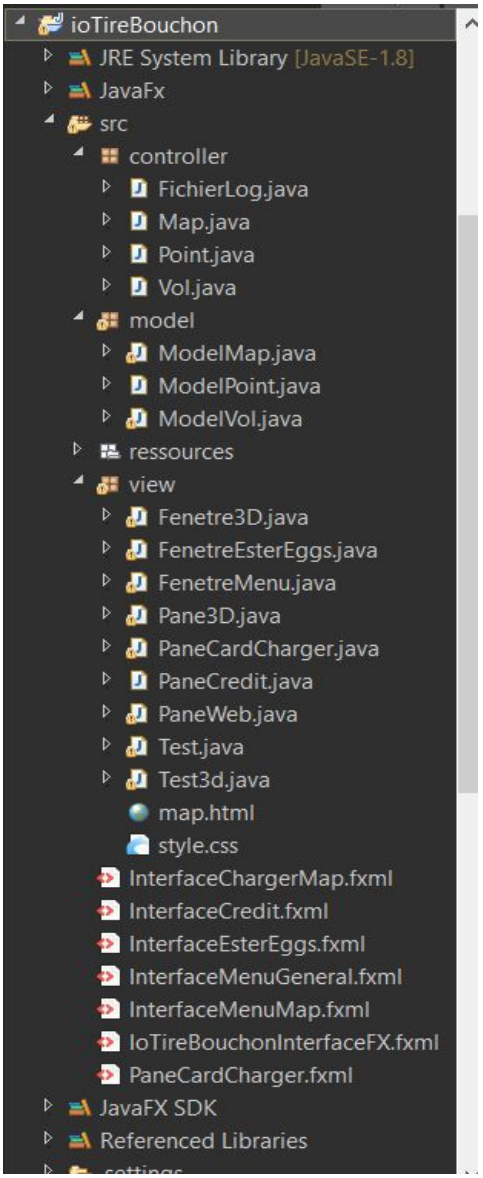
```
DELIMITER //
CREATE TRIGGER suppression_map BEFORE DELETE on map FOR EACH ROW
BEGIN
    DELETE FROM point WHERE map_idMap=Old.idMap;
END; //
DELIMITER //
CREATE TRIGGER suppression_vol BEFORE DELETE on map FOR EACH ROW
BEGIN
       DELETE FROM vol WHERE idVol=Old.vol_idVol;
END; //
```

WorkPackage 4.2 : Développement logiciel

Une grosse partie du projet est la retranscription visuelle des données captées par le drone. Pour ce faire nous allons développer une application dans ce WorkPackage en java. Celle-ci aura pour but de dessiner des points dans un espace 3D propulsé par la bibliothèque JavaFX (un peu dans la façon de la fonction plot dans matlab mais ici en 3D).

Pour structurer notre code nous avons utilisé un design pattern **MVC** un peu modifié spécialement pour notre projet.





Nous avons constitué plusieurs packages contenant le principale model view controller cependant nous utilisons un packages ressources qui contient toutes nos photos, son, fichiers externes.

Pour chaque table de la base de données nous avons créer des objets qui sont constitués de tous les n-uplets de chaques tables. Nous avons intégré des getter and setters pour permettre la récupération d'informations.

WorkPackage 4.2.1 : Récupération des données de la base de données

Objectif *:* Dans ce WorkSpace nous allons nous connecter de l'application vers la base de données via JDBC (Java DataBase Connectivity), ce qui va nous permettre de faire des requêtes sur la base de données hébergé sur le serveur depuis notre code java.

Réalisation : Nous nous sommes connecter à notre base via JDBC, nous avons un fonction d'initialisation qui nous permet de nous connecter à la base

```
public static void init() throws ClassNotFoundException, SQLException
	{
		Class.forName("com.mysql.jdbc.Driver");
		conn = (Connection) DriverManager.getConnection(DB_URL,USER,PASS);
		stmt = (Statement) conn.createStatement();
	}

	public static ArrayList<Map> getAllMap() throws SQLException
	{
		try {
			init();
		} catch (ClassNotFoundException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
		ArrayList<Map> tabMapRes = new ArrayList<Map>();
		String sql;
		sql = "SELECT * FROM Map";
		ResultSet rs = (ResultSet) stmt.executeQuery(sql);
		while(rs.next())
		{
			int id  = rs.getInt("idMap");
			String nomMap = rs.getString("nomMap");
			Date dateGenere = rs.getDate("dateMapGenere");
			int vol_idVol = rs.getInt("vol_idVol");
			double neLat = rs.getDouble("noLat");
			double neLng = rs.getDouble("noLng");
			double ouLat = rs.getDouble("seLat");
			double ouLng = rs.getDouble("seLng");

	tabMapRes.add(new Map(id,nomMap,dateGenere,vol_idVol,neLat,neLng,ouLat,ouLng));

		}

		return tabMapRes;}
```

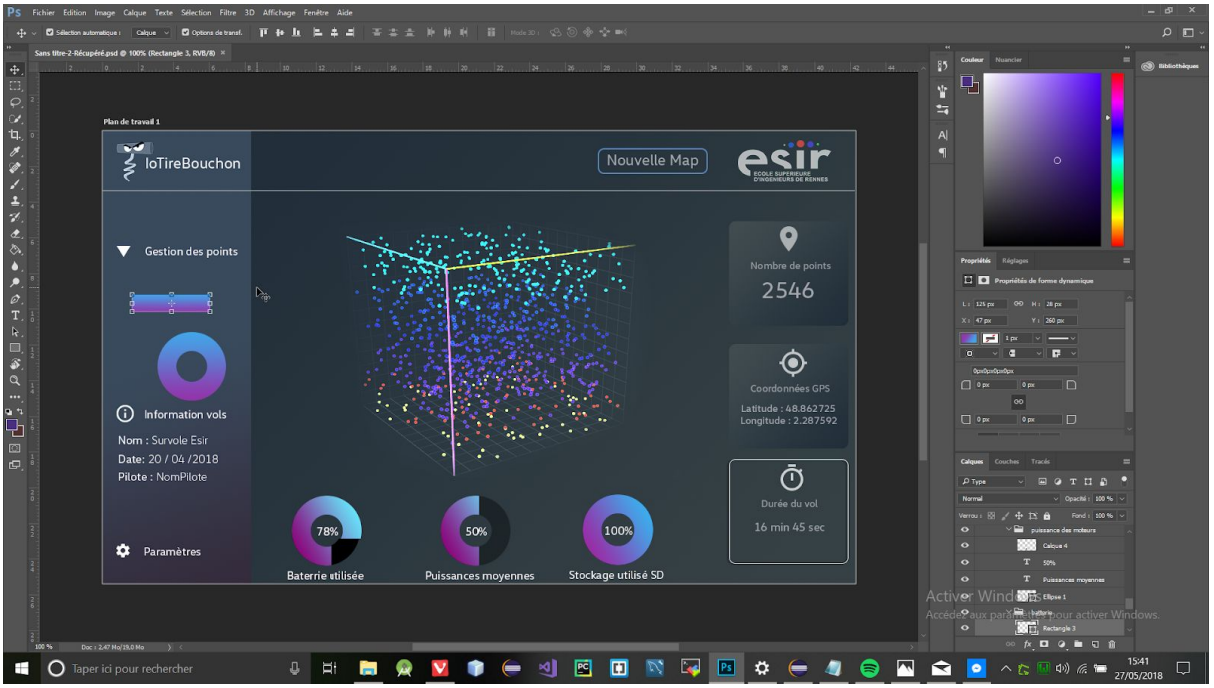
Ici la fonction permettant de récupérer un tableau dynamique d'objets Map. Cette méthode appartient à la classe ModelMap qui contient d'autres fonction comme la suppression, l'ajout et la réception.

WorkPackage 4.2.2 : Traitement de l'information via Java

Ce WorkPackage résume les fonctionnalités de base de notre application. Celles-ci vont peut être évoluer d'ici le lancement du développement ce logiciel.

WorkPackage 4.2.2.1 : Interface graphique en 3D (IHM)

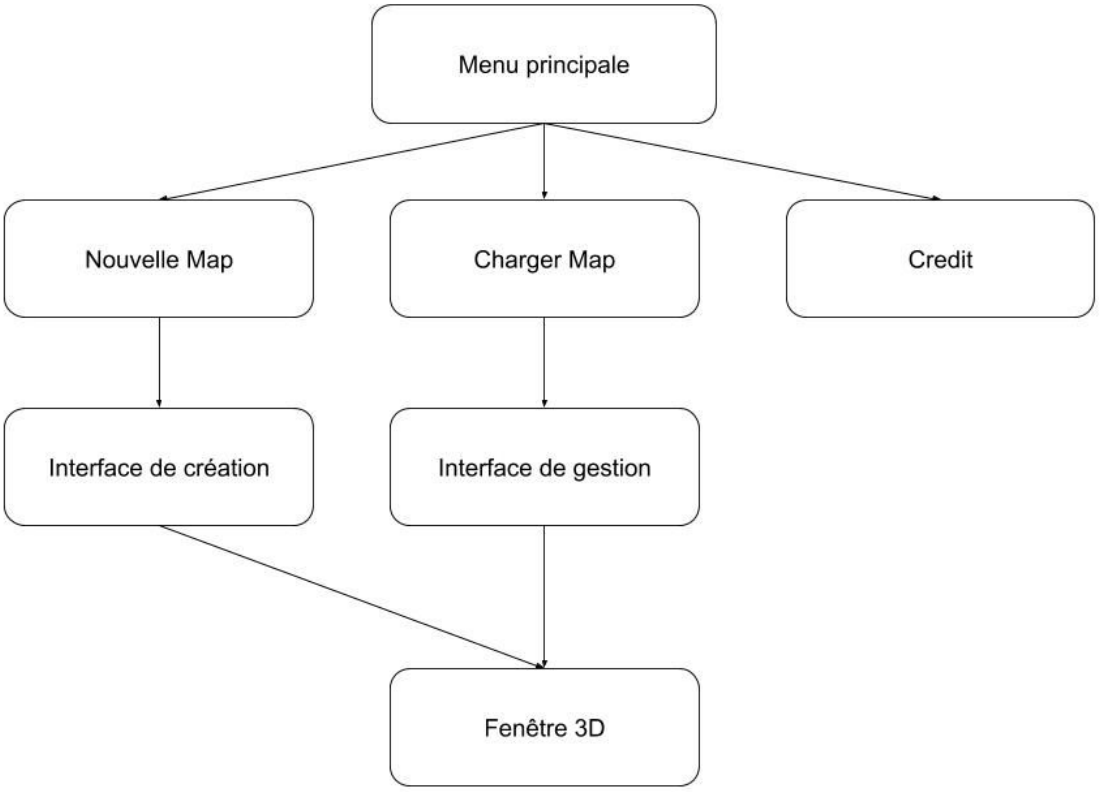
Pour ce faire nous avons créé des maquettes via photoshop pour nous visualiser l'interface graphique et la charte graphique du logiciel :



Objectif : L'interface graphique va permettre dans un premier temps à l'utilisateur de choisir de créer une nouvelle carte pour un nouveau terrain à cartographier. Ou bien de visualiser une carte déjà existante:

- Premier cas :
  - Création d'une map
  - Upload des données dans cette map (Workpackage 4.2.2.4)
  - Avertissement à l'utilisateur que sa carte a été créé.
  - Renvoi au menu principal
- Deuxième cas :
  - Visualisation d'une carte existante
  - Affichage des cartes existantes
  - Choix d'une carte à visualiser
  - Affichage de la carte en 3D avec les points captés dans la base de données.

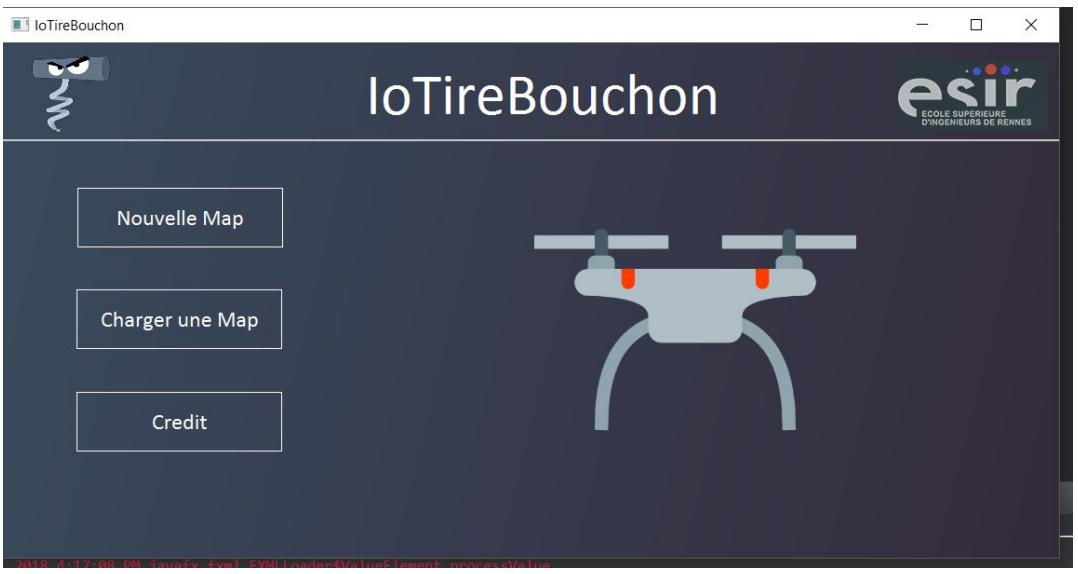
Résultat : Pour réaliser ce projet nous avons découpé l'IHM en plusieurs panels et fenêtres pour offrir une expérience utilisateur qui pète sa mère.



Il y a donc la fenêtre principale qui contient 3 panels, dans le premier panel "nouvelle map" qui permet de créer une map en chargeant le fichier de log récupéré dans le pixhawks. Le panel charger map contient toutes les maps déjà créés. Le panel crédit contient les informations des élèves ingénieurs.

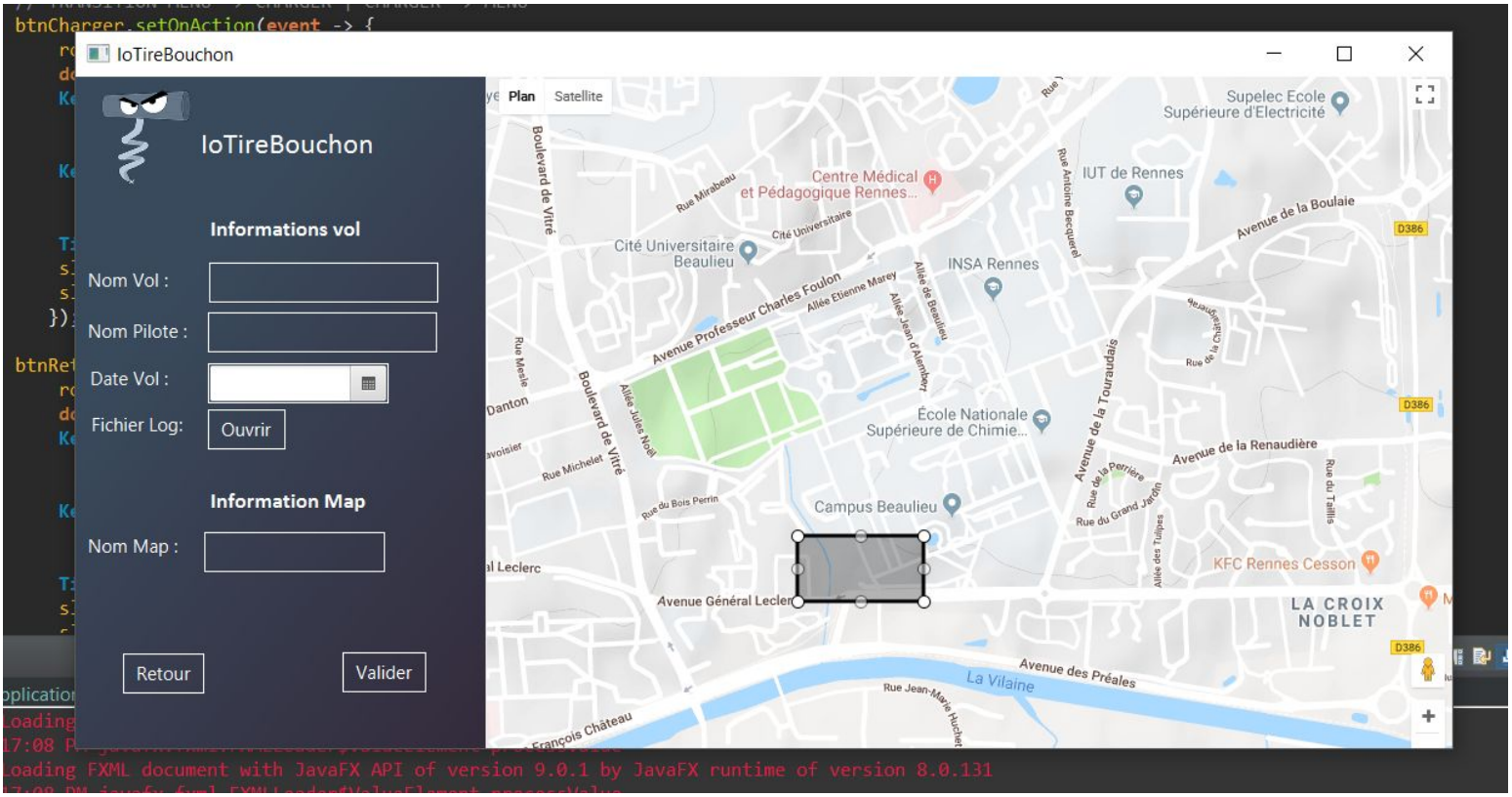
La deuxième fenêtre contient le panel qui place les points en 3D.

Menu Principale:



Ici la fenêtre est composée de 3 boutons qui permette de créer une animation de slide pour afficher les panneaux correspondant aux boutons. On peut y retrouver le logo de l'ESIR, d'IOTireBouchon et le nom de notre groupe avec une petite image d'un drone.

Le panel contenant l'IHM "nouvelle map" est composé de plusieurs inputs text pour que l'utilisateur rentre les informations concernant le vol, ainsi qu'un bouton "ouvrir log" qui permet d'ouvrir un fenêtre permettant d'aller chercher le path du fichier à traiter.



Ce panel a été assez compliqué à réaliser il utilise différentes technologies :

- JAVA pour le back (les inputs)
- FXML pour l'interface graphique
- WebView pour afficher un fichier html
- Javascript pour interagir avec l'API Google Maps

Nous avons donc utilisé l'API google pour permettre de récupérer les coordonnées sphériques de la zone survolé par le drone

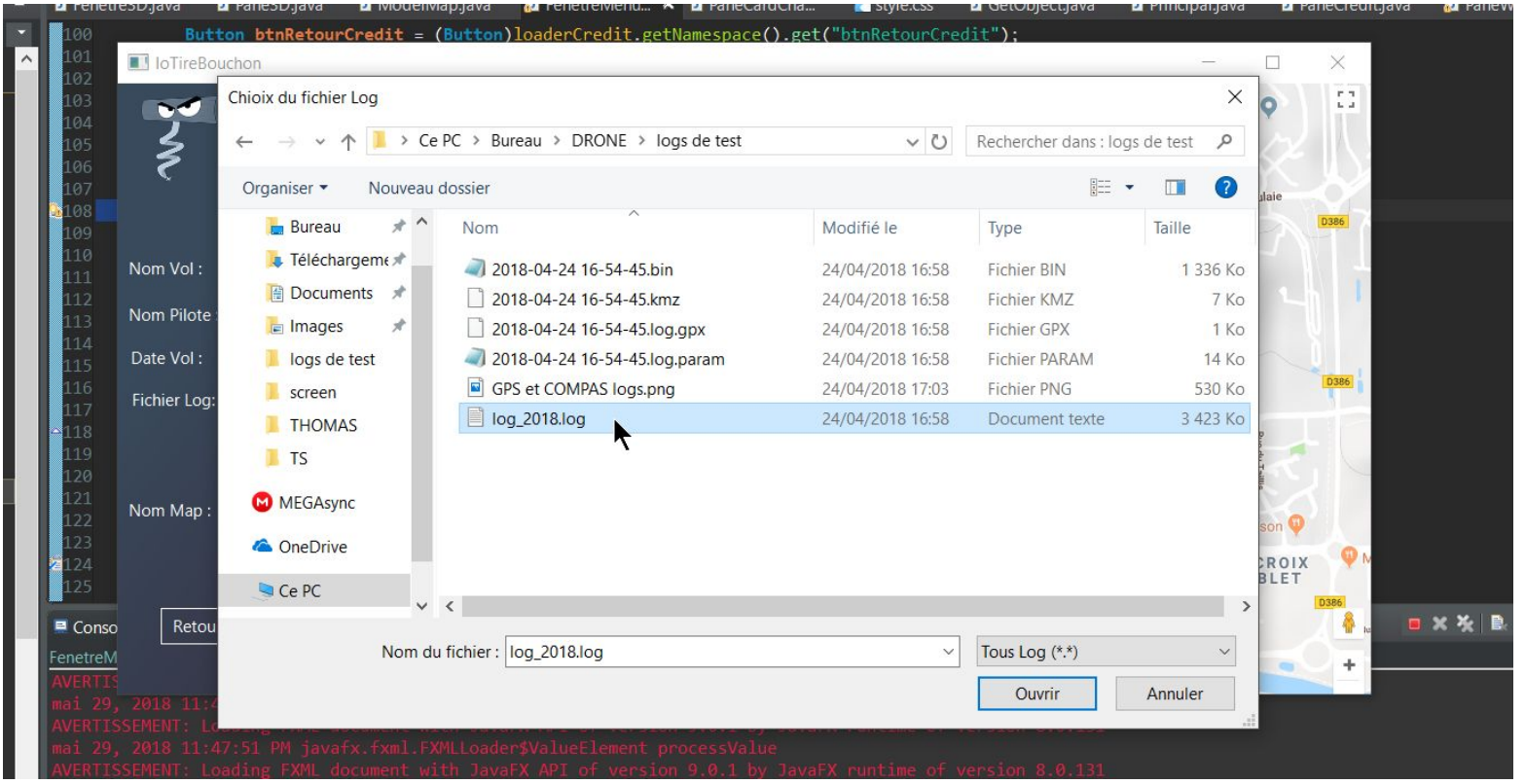
Ici le code qui est incorporé dans un fichier html et qui est implanté dans un objet webview javafx.(Code javascript)

```
        <script>
            var rectangle;
            function initMap() {
              var map = new google.maps.Map(document.getElementById('map'), {
                zoom: 15,
                center: {lat: 48.117932, lng: -1.636422},
                mapTypeId: 'terrain'
              });

              var bounds = {
                north: 48.115455,
                south: 48.113908,
                east: -1.638129,
                west: -1.642593
              };

              // Define a rectangle and set its editable property to true.
                rectangle = new google.maps.Rectangle({
                bounds: bounds,
                editable: true,
                draggable: true,
                geodesic: true
              });

              rectangle.setMap(map);
              var sw = rectangle.getBounds().getSouthEast();
              console.log("Se : " + sw.lat());
            }

           function getNordOuestLat()
           {
              return rectangle.getBounds().getNorthEast().lat();
           }
        ...
</script>
```

Une fois que le WebView est chargé nous pouvons récupérer les coordonnées de longitude et latitude du NO et SE. Ce qui va nous permettre de générer un map à la taille voulu. (Code java)

```
final URL urlGoogleMaps = getClass().getResource("map.html");
webEngine.load(urlGoogleMaps.toExternalForm());
getChildren().add(webView);
public double[] getCoord()
  {

    double nordOuestLat = (double) webEngine.executeScript("getNordOuestLat()");
    double nordOuestLng = (double) webEngine.executeScript("getNordOuestLng()")
    double sudEstLat = (double) webEngine.executeScript("getSudEstLat()");
        double sudEstLng = (double) webEngine.executeScript("getSudEstLng()");

    double res[] = {nordOuestLat,nordOuestLng,sudEstLat,sudEstLng};
    return res; }
```

Ici nous pouvons distinguer la fenêtre qui va ouvrir le gestionnaire de fichier de l'OS ou est lancer l'application.



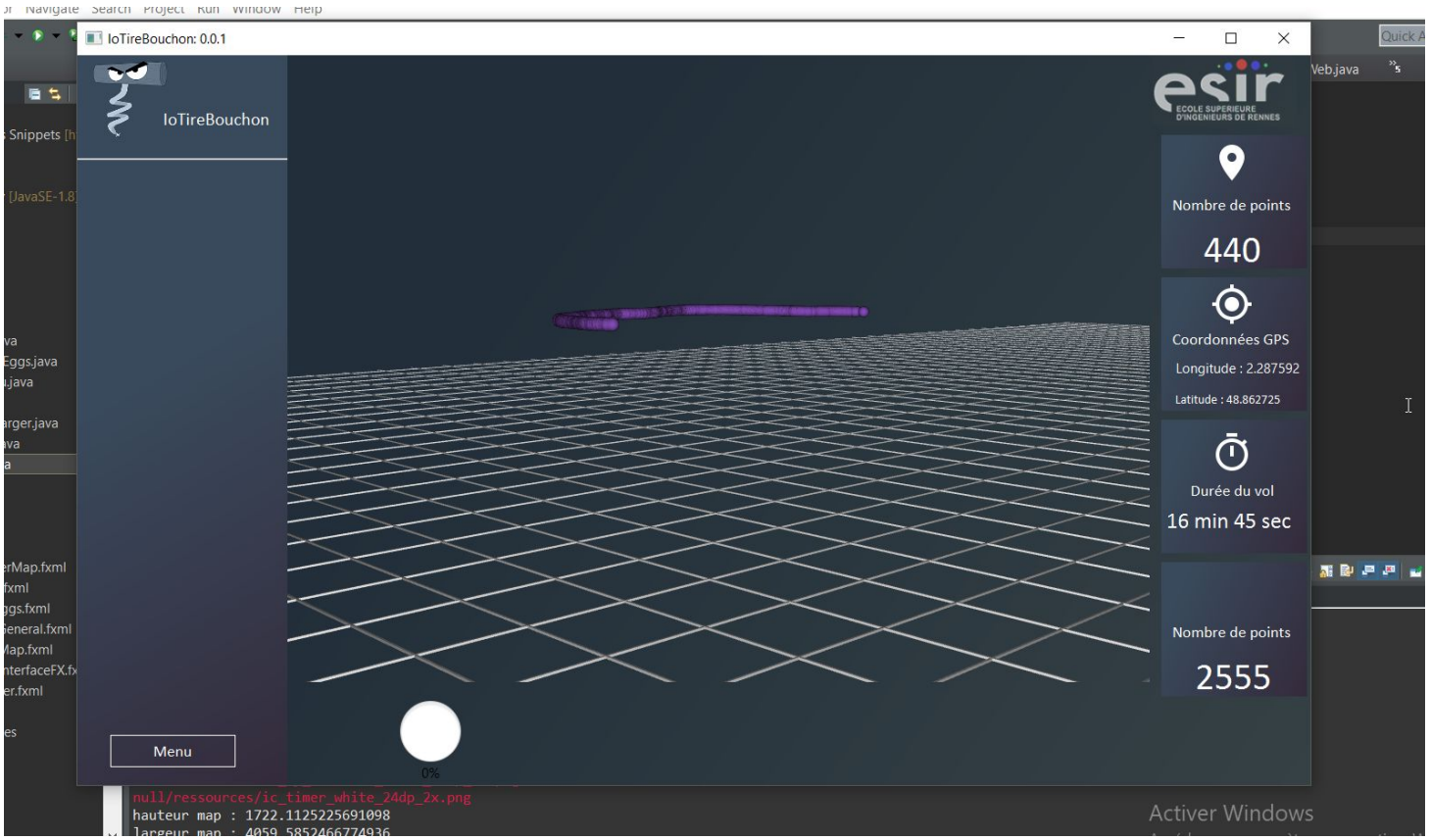
Une fois que le path du fichier log est enregistré dans une variable nous pouvons procéder au décryptage et sélectionner les informations à récupérer.

On récupère les valeurs des inputs pour peupler la base de données avec les informations saisie par l'utilisateur.

On vérifie le format des inputs pour ne pas impacter la base de données par des informations non normalisées.

Fenêtre 3D :

Cette partie du logiciel est la plus compliquée à réaliser, nous avons utilisé la bibliothèque javaFx pour réaliser un vue 3D réaliser par un système de caméra virtuelle dirigé par des Événement de souris.



Ici le layout utilisé nous permet de séparer en 4 partie la fenêtre graphique, sur la gauche devait être affichée les informations sur la map mais par manque de temps cette partie a été retiré des sprints de développement tout comme la partie basse de la fenêtre.

Sur le panel de droit on peut avoir quelques informations sur le vol comme le nombres de points capté durant le vols, les coordonnées gps du premier points capter, la durée du vol.

En ce qui concerne le panel central avec la vu 3D une classe spécial a été développée. On passe dans le constructeur un tableau de points qui va être disposé via triangulation dans le scène de javaFx.

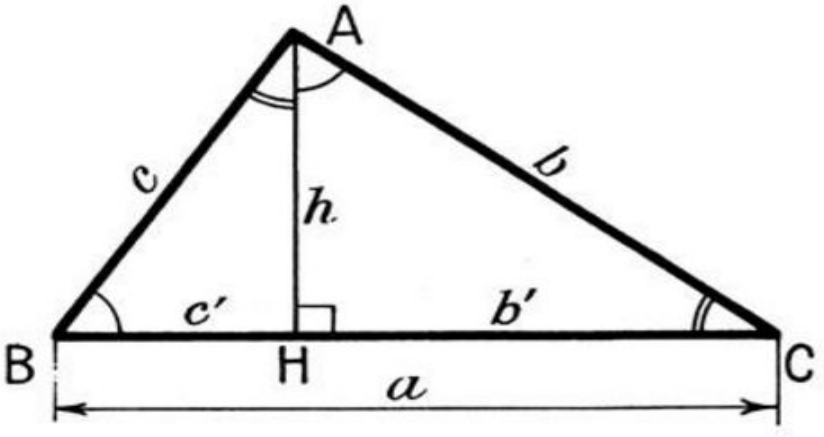
Nous avons créé un algorithme nous permet de calculer la longueur entre deux coordonnées sphériques ce qui va nous permettre de placer les points en 3d. Nous récupérons donc une distance en mètre entre deux points.

```
public static double coordSpheriqueVersMetre(double lngReference,double latReference,double
lng,double lat)
	{
		double R = 6378.137; // Rayon de la terre
		double dLat = Math.abs(latReference * Math.PI / 180 - lat * Math.PI / 180);
		double dLon = Math.abs(lngReference * Math.PI / 180 - lng * Math.PI / 180);
		double a = Math.sin(dLat/2) * Math.sin(dLat/2) +
				Math.cos(lat * Math.PI / 180) * Math.cos(lngReference *
Math.PI / 180) *
				Math.sin(dLon/2) * Math.sin(dLon/2);
		double c = 2 * Math.atan2(Math.sqrt(a), Math.sqrt(1-a));
		double d = R * c * 1000;

		return d;

	}
```

Une fois que cet algo a été développé nous avons pu utiliser les coordonnées gps de l'api google qui nous permet d'avoir une coordonnée de référence (le point C est le point de référence)

```
maxLong,maxLat  A-------B minLong,maxLat
                |       |
                |       |
                |       |
 maxLong,minLat  D-------C minLong,minLat
```

Nous pouvons enfin passer à la triangulation pour placer les points de base sphérique dans un plan orthogonal x,y qui fait la taille de la maps réglé via l'API de google.



Nous passons par une formule assez simple pour trouver l'emplacement du point dans le repère (x = c' par exemple). Le triangle choisit es un triangle rectangle :

Voici l'algo qui nous renvoie un tableau de deux position : x,y à répartir ensuite sur le plan 3D la hauteur z est renseignée lors du traitement du fichier.

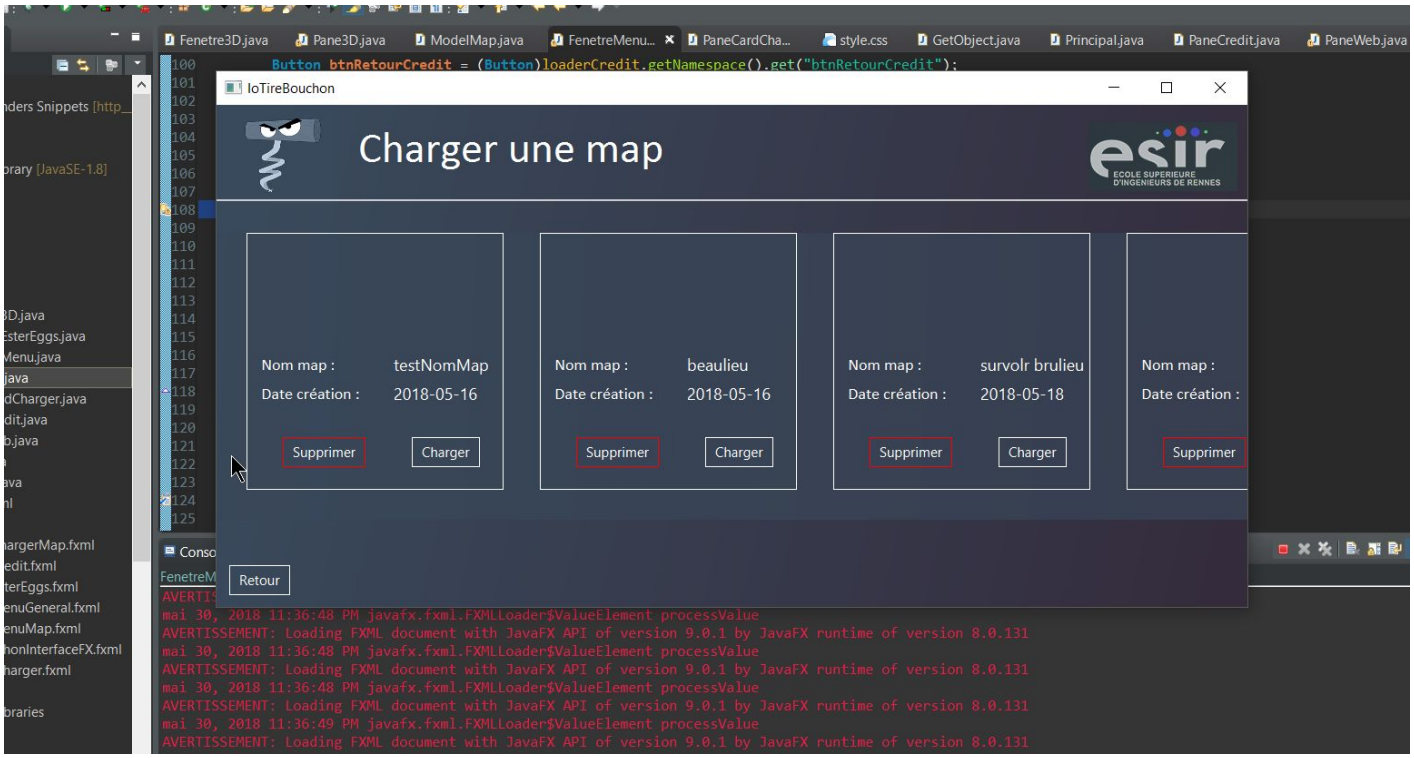
```
public static double[] triangulation(double lonPara,double latPara)
    {

        /** DISTANCE EN METRE **/
        double d1B = coordSpheriqueVersMetre(minLongitude,maxLatitude,lonPara,latPara);
        double d2C = coordSpheriqueVersMetre(minLongitude,minLatitude,lonPara,latPara);
        double d3D = coordSpheriqueVersMetre(maxLongitude,minLatitude,lonPara,latPara);

        double x = (Math.pow(distanceCB,2.0) - Math.pow(d1B, 2.0) + Math.pow(d2C,
2.0))/(2*distanceCB);
        double y = (Math.pow(distanceCD,2.0) - Math.pow(d3D, 2.0) + Math.pow(d2C,
2.0))/(2*distanceCD);

        /** DISTACE EN PIXEL  **/

        double ptsX = 10*x;
        double ptsY = 10*y;

        double[] tab = {ptsX,ptsY};
               return tab;
    }
```

Il n'y a plus qu'à générer des sphères avec une version de couleur en fonction de la hauteur du points (nuancier de 6 couleur en fonction de la hauteur des points).

```
Sphere sphere = new Sphere()
sphere.setRadius(5.0);
double[] tabXY = triangulation(tabPoint.get(i).longitude,tabPoint.get(i).latitude);
sphere.setTranslateX(-tabXY[0]);
sphere.setTranslateY(-10*tabPoint.get(i).hauteur);
sphere.setTranslateZ(-tabXY[1]);
                if(sphere.getTranslateY() > -40){
                       sphere.setMaterial(bleuFoncer);
                 }
                 else if(sphere.getTranslateY() <=- 40 && sphere.getTranslateY() > -100){
                       sphere.setMaterial(bleuClair);
                 }
                 else if(sphere.getTranslateY() <= -100 && sphere.getTranslateY()  >
-250){
                       sphere.setMaterial(vertClair);
                 }
                 else if(sphere.getTranslateY() <= -250 && sphere.getTranslateY()  >
-350){
                       sphere.setMaterial(jaune);
                 }
                 else if(sphere.getTranslateY() <= -350 && sphere.getTranslateY()  >
-450){
                       sphere.setMaterial(orange);
                 }
                 else{
```

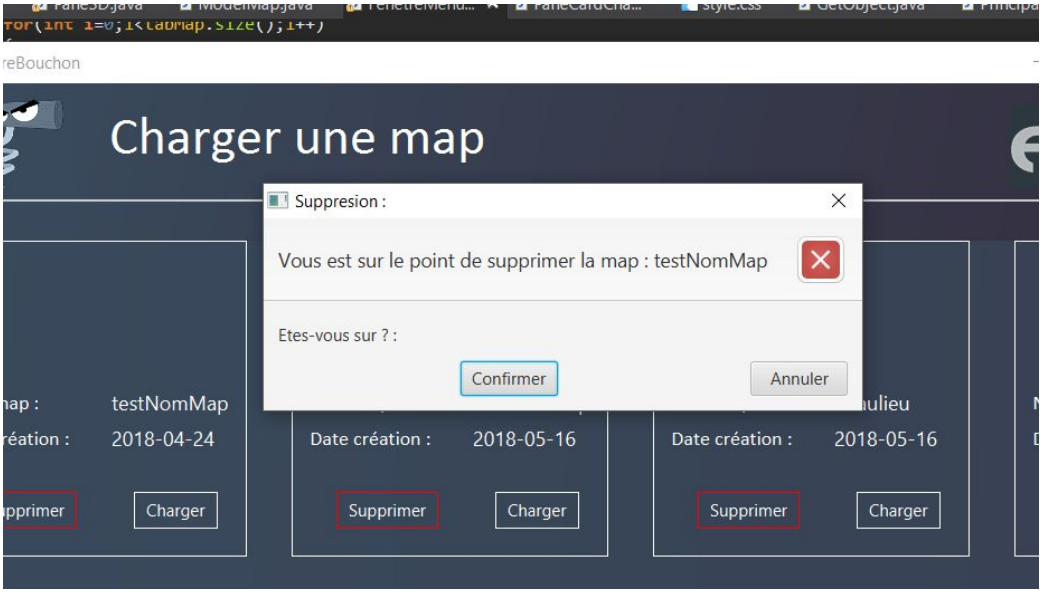
```
        sphere.setMaterial(rouge);
    }
```

Nous avons un bouton pour retour vers le menu principale, depuis celui-ci nous pouvons accéder au panel permettant la gestion des maps déjà générées.



Deux actions sont disponible : soit charger soit supprimer, lors du chargement de la map le logiciel va interroger la base de données pour récupérer tous les points liés à la map et charger une fenêtre 3D.

Lors de la suppression le logiciel va demander une confirmation via une pop-up si l'utilisateur valide la suppression alors le SGBD va s'occuper de supprimer en cascade et déclencher des triggers vu précédemment.



WorkPackage 4.2.2.2 : Liaison interface - données récupérées

Objectif : Le logiciel devra faire des requêtes sur la base de données et récupérer les informations pour afficher les donnés.

Résultat : (Voir wp Récupération de données)

WorkPackage 4.2.2.3 : Différentes fonctionnalités

Objectif : Ce WorkPackage est optionnel il permet d'ajouter des fonctionnalités à l'application si nous respectons nos deadlines.
Nous aimerions développer par exemple :

- Enregistrer l'image vue dans le render au format jpg
- Interface avec un UX | UI agréable
- Personalisation des points (couleur, grosseur, nomenclature)

Résultat : Pas eu le temps de faire ce wp étant optionnel non critique pour le projet

WorkPackage 4.2.2.4 : Lecture des fichiers logs | écriture bdd

Objectif : Le logiciel devra, lorsqu'on lui importe un fichier log, le décrypter, récupérer l'information, la traiter, écrire celle-ci dans la base de données au bon endroit. Une fois les données écrites, le logiciel pourra afficher les points dans une fenêtre 3D.

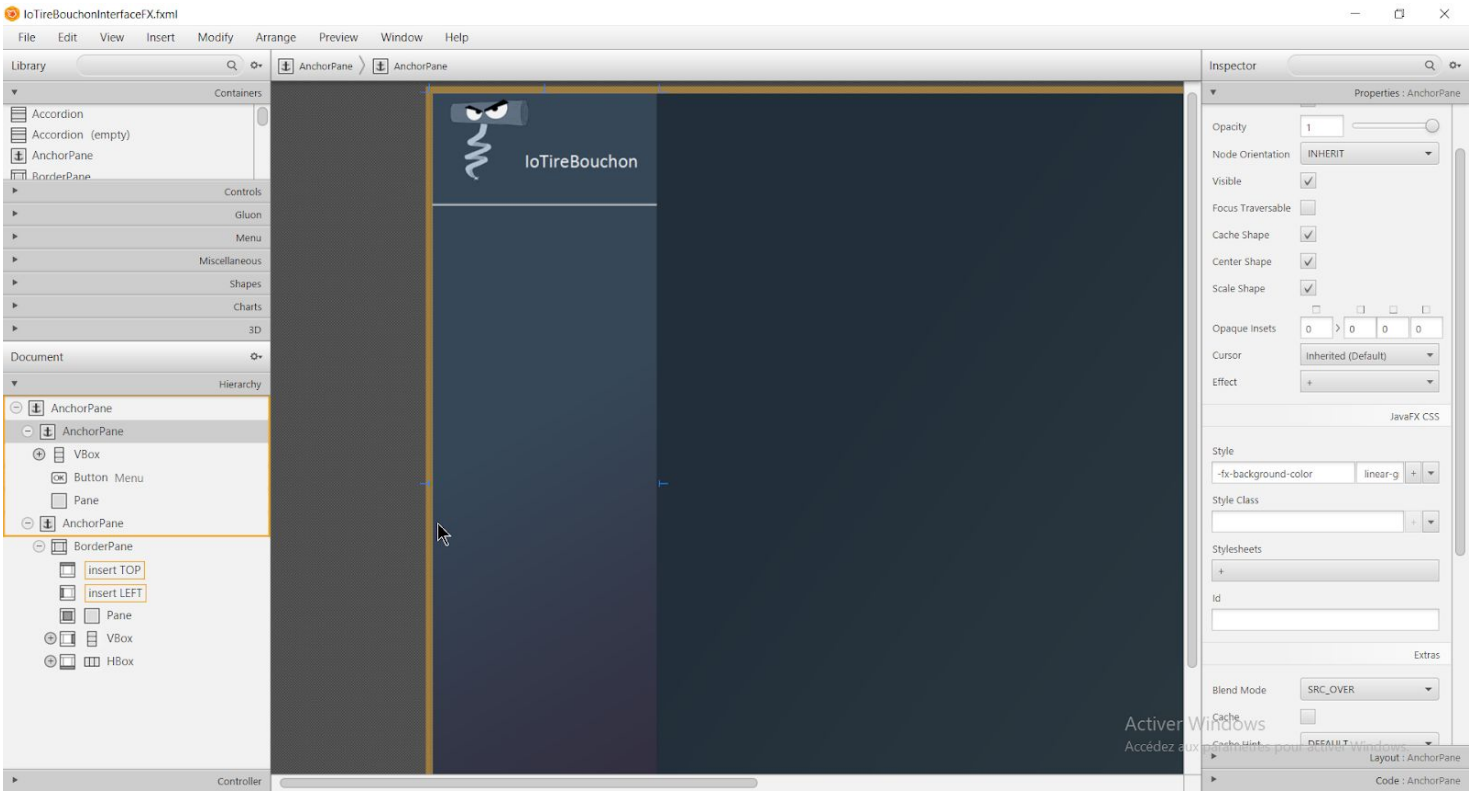
Résultat : Ici nous allons voir le résultat de l'algo de decriptage des fichiers logs :

```
try{
                        InputStream flux=new FileInputStream(url);
                        InputStreamReader lecture=new InputStreamReader(flux);
                        BufferedReader buff=new BufferedReader(lecture);
                        String ligne;

                        ModelPoint mp = new ModelPoint();

                        while ((ligne=buff.readLine())!=null){
                               String[] parts = ligne.split("(?=,)");
                               String typeDeDonne = parts[0];
                               if(typeDeDonne.equals("GPS2"))
                               {
                                      String latitude = parts[7].replace(", ", "");
                                      String longitude = parts[8].replace(", ", "");

mp.ajouterPoint(Double.valueOf(longitude),Double.valueOf(latitude), 10, idMap);
                               }
                        }
                        buff.close();
                        }
                        catch (Exception e){
                        System.out.println(e.toString());
                        }
```

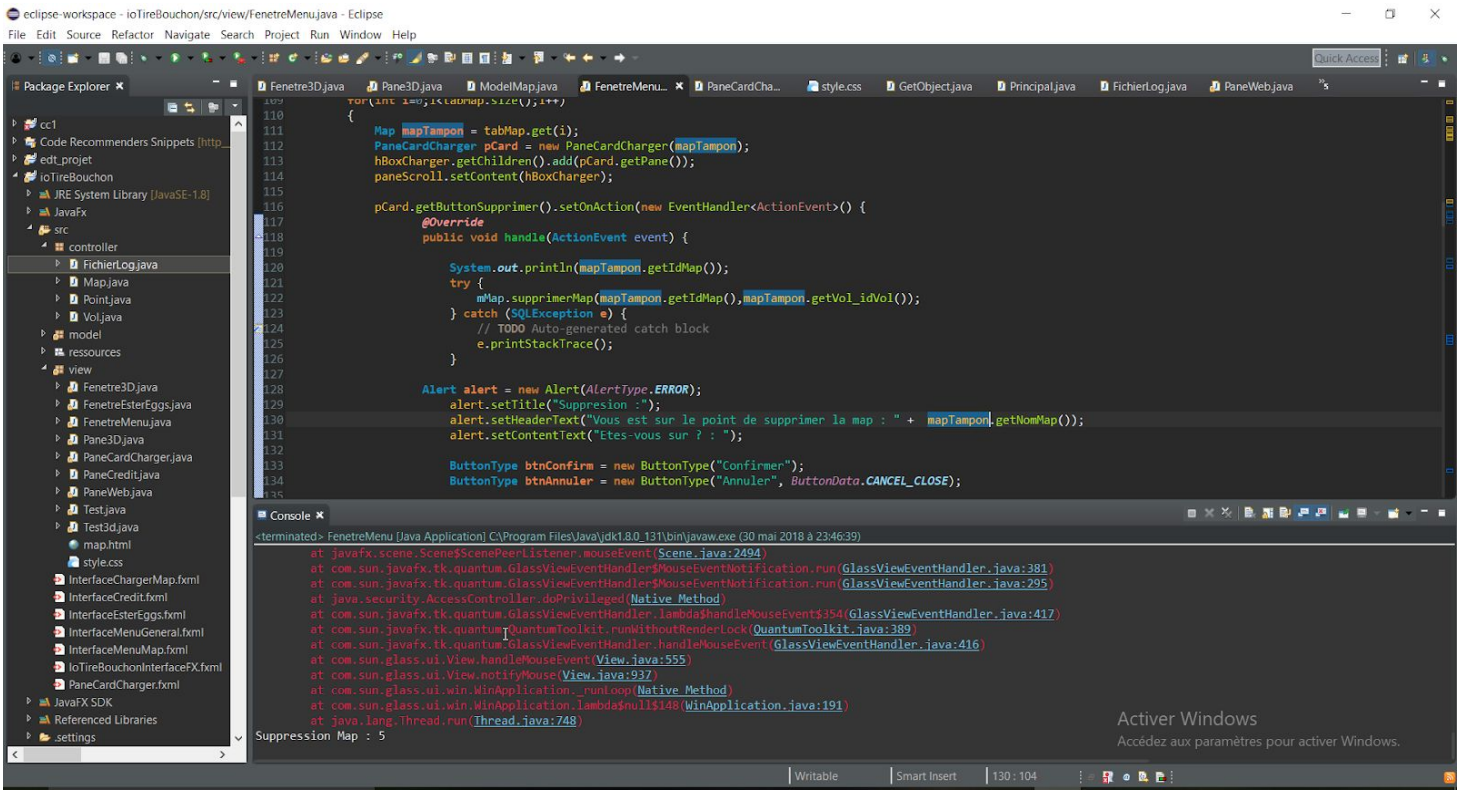
On peut constater que le modelMap récupère les données des points et l'ajouter dans la base de données avec l'id de la map correspondant.
Ici le cas du lidar n'est pas traité car le fichier log du lidar n'a pas eu le temps d'être analysé.

Annexe Outils utilisés :

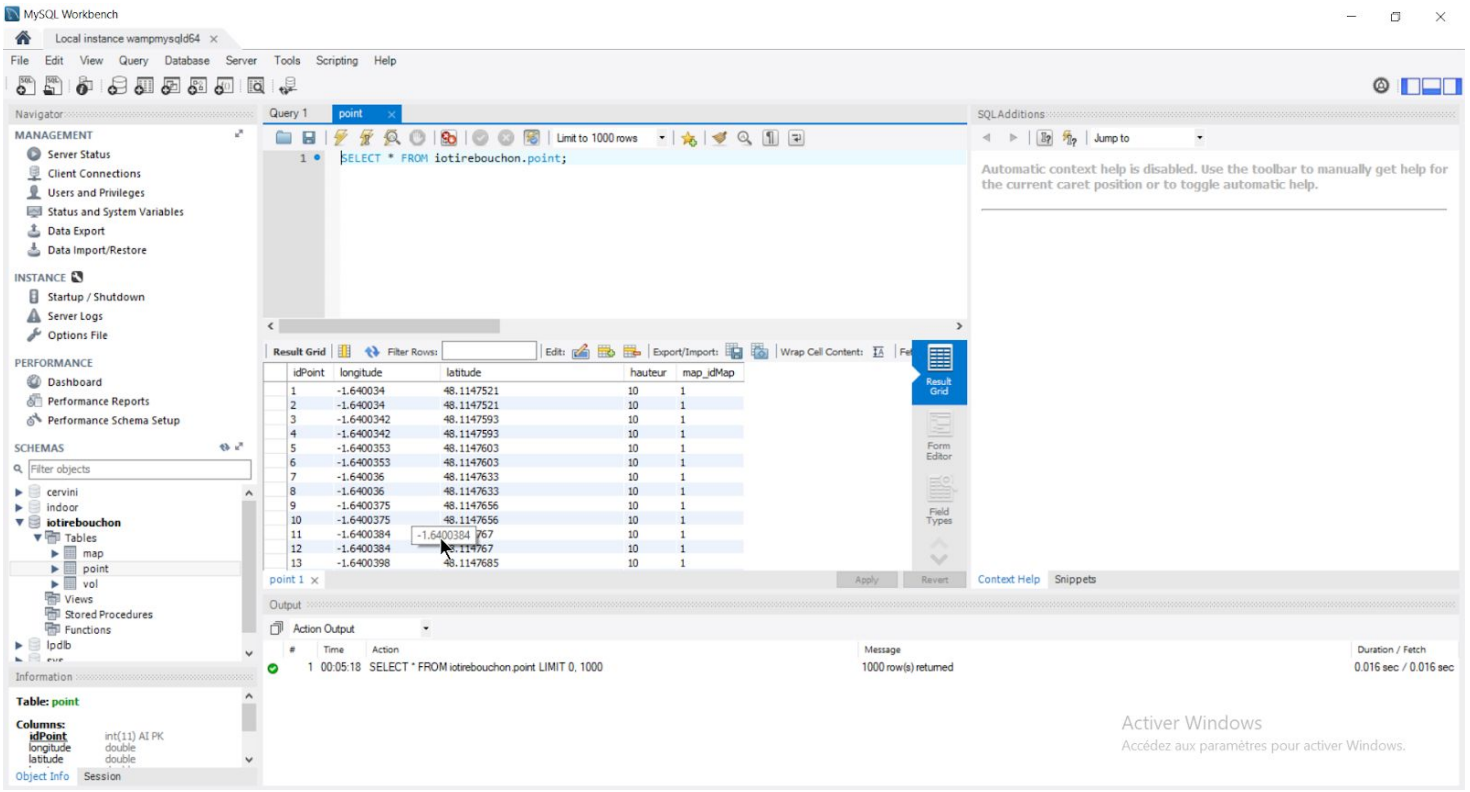
Logiciel Scene builder pour générer des fichier FXML permettant de gagner du temps lors de la disposition des éléments dans les fenêtres graphique :



Eclipse a été utilisé comme IDE pendant la période de développement.



MySQL WorkBench nous permettait de gérer la base de données.



Bonus : un easter egg est caché dans le software … juste un indice : iot