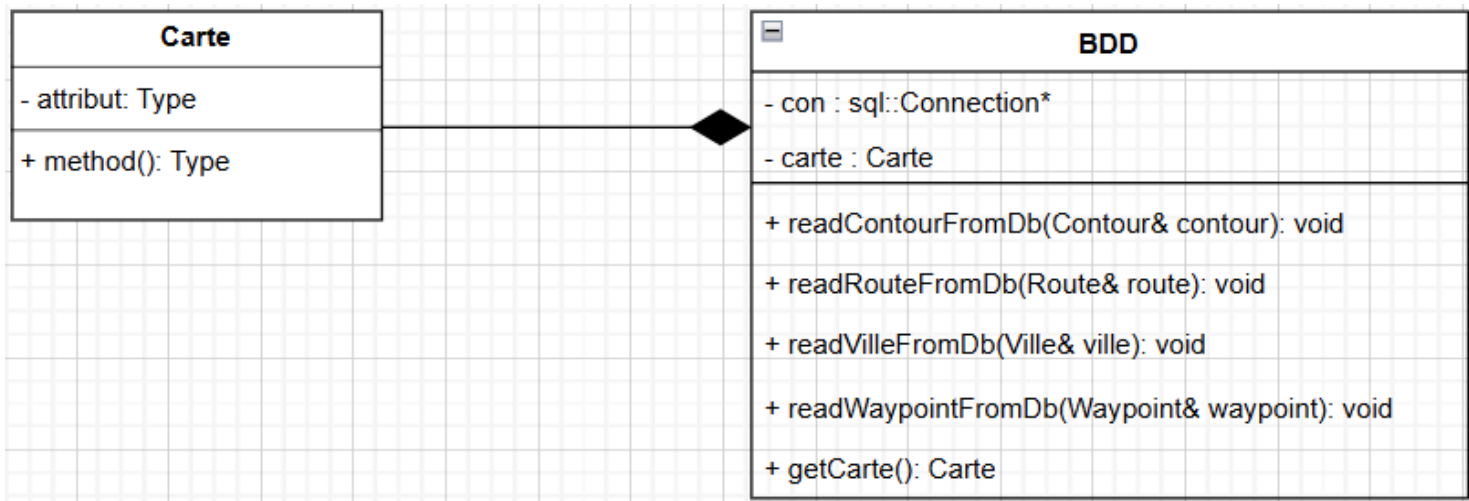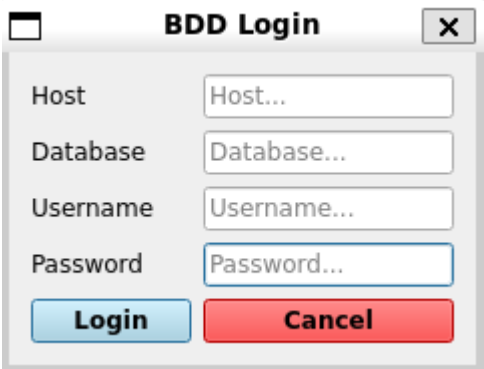# RoutePlanner Roadmap

1. Database

We will need the database information (Route, Waypoint, Contour, etc.) to have something to visualize. Therefore, the implementation of the **BDD** <class> must be completed and tested first.

The person responsible for its implementation can test the **BDD** <class> using the **LoginDialog** <class> for testing in main.cpp .
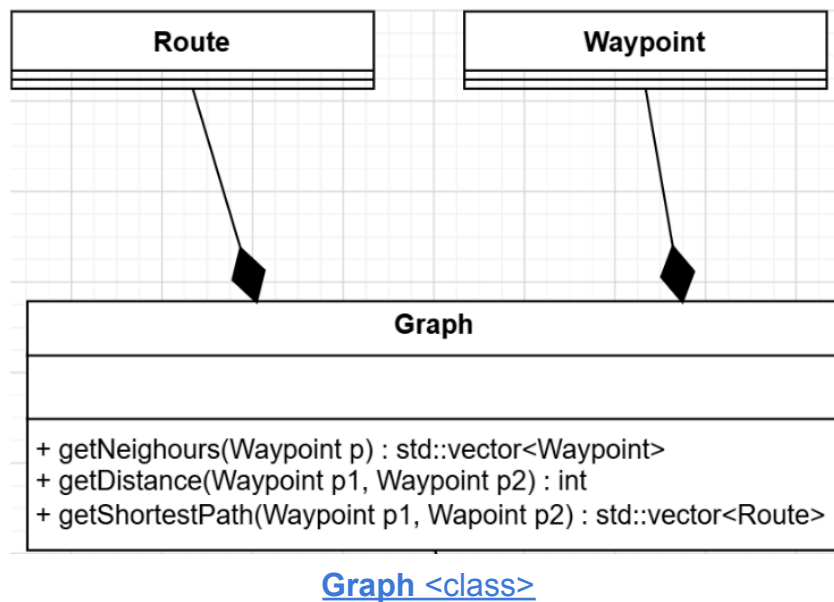

BDD <class>

LoginDialog <class>

2. **Graph**

The class **Graph** <class> required the Route and Waypoint data so that we can test its efficiency. Completed, it can be tested in the cmd line.
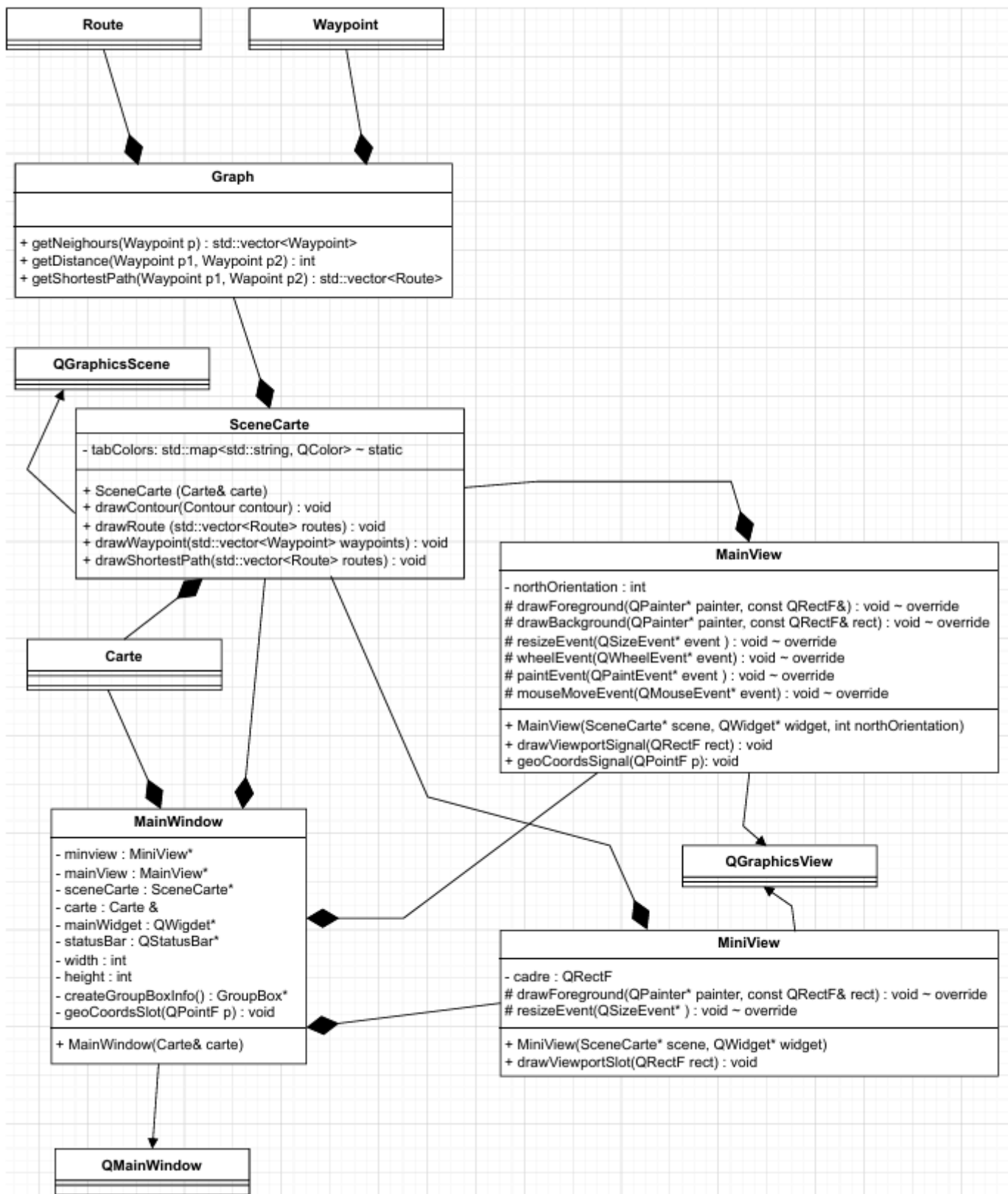
Here is the UML diagram of **Graph** <class>.

Graph <class>

3. <u>User Interface</u>

Implementing the user interface will be all necessary functionalities will be our main challenge. But by going step by step, we can achieve it without issues of grouping things together.

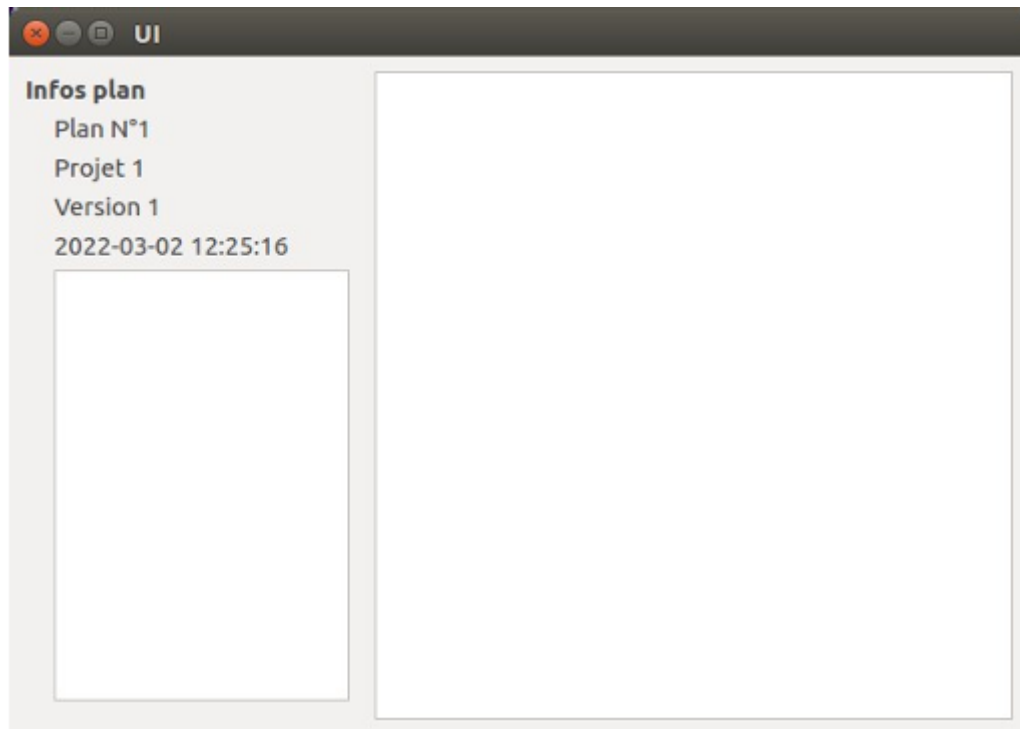First of all, take a look at the UML diagram of this part.

Route

Waypoint

Graph

```
+ getNeighours(Waypoint p) : std::vector<Waypoint>
+ getDistance(Waypoint p1, Waypoint p2) : int
+ getShortestPath(Waypoint p1, Wapoint p2) : std::vector<Route>
```

QGraphicsScene

SceneCarte

```
- tabColors: std::map<std::string, QColor> ~ static

+ SceneCarte (Carte& carte)
+ drawContour(Contour contour) : void
+ drawRoute (std::vector<Route> routes) : void
+ drawWaypoint(std::vector<Waypoint> waypoints) : void
+ drawShortestPath(std::vector<Route> routes) : void
```

MainView

```
- northOrientation : int
# drawForeground(QPainter* painter, const QRectF&) : void ~ override
# drawBackground(QPainter* painter, const QRectF& rect) : void ~ override
# resizeEvent(QSizeEvent* event ) : void ~ override
# wheelEvent(QWheelEvent* event) : void ~ override
# paintEvent(QPaintEvent* event ) : void ~ override
# mouseMoveEvent(QMouseEvent* event) : void ~ override

+ MainView(SceneCarte* scene, QWidget* widget, int northOrientation)
+ drawViewportSignal(QRectF rect) : void
+ geoCoordsSignal(QPointF p): void
```

Carte

MainWindow

```
- minview : MiniView*
- mainView : MainView*
- sceneCarte : SceneCarte*
- carte : Carte &
- mainWidget : QWigdet*
- statusBar : QStatusBar*
- width : int
- height : int
- createGroupBoxInfo() : GroupBox*
- geoCoordsSlot(QPointF p) : void

+ MainWindow(Carte& carte)
```

QGraphicsView

MiniView

```
- cadre : QRectF
# drawForeground(QPainter* painter, const QRectF& rect) : void ~ override
# resizeEvent(QSizeEvent* ) : void ~ override

+ MiniView(SceneCarte* scene, QWidget* widget)
+ drawViewportSlot(QRectF rect) : void
```

QMainWindow

a. Implementation of the **MainWindow** <class>
    Methods on which we should be focus on are:
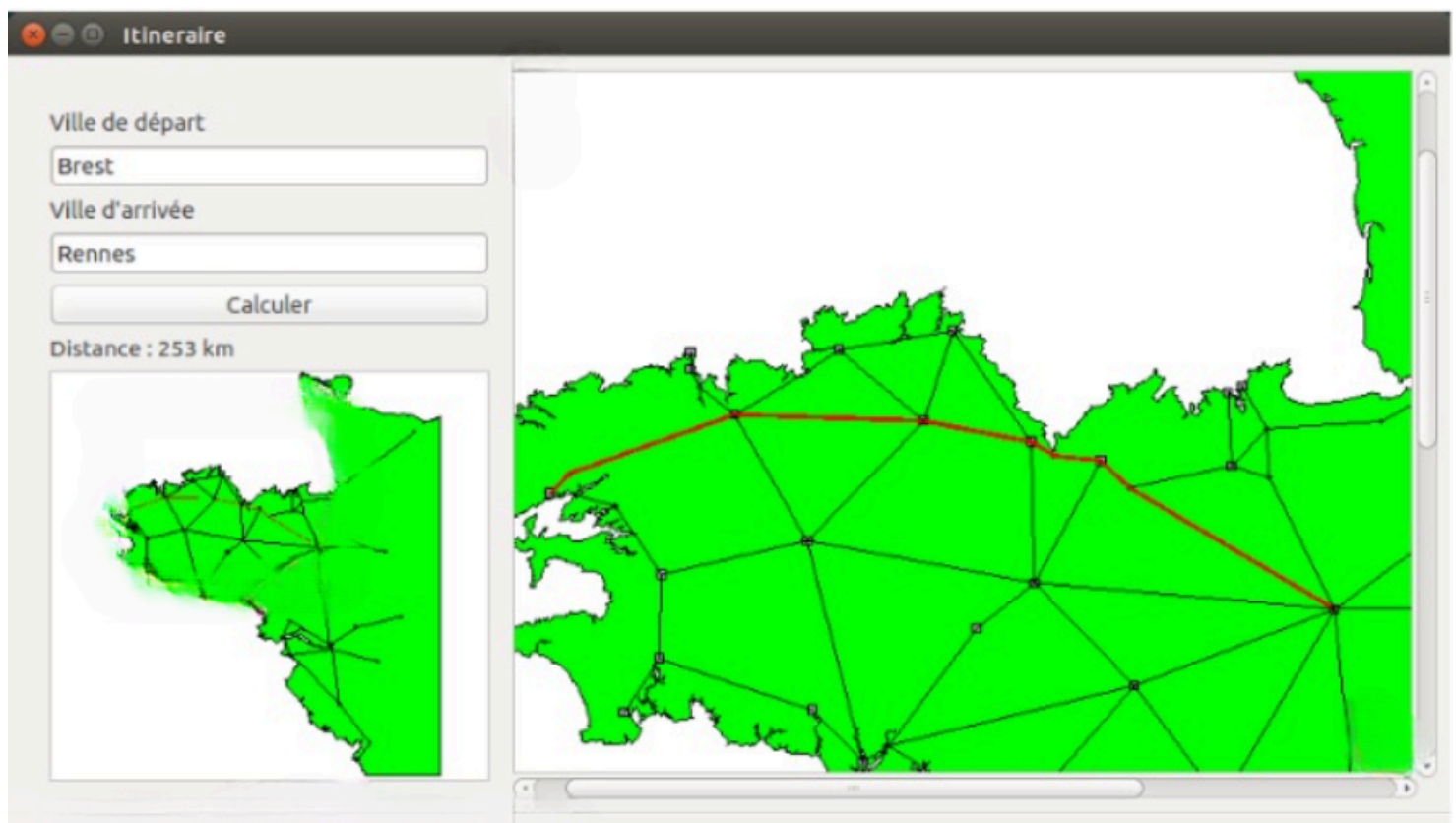   -   MainWindow(Carte& carte)
    -   createGroupBoxInfo() : GroupBox*

She should have something similar to this:

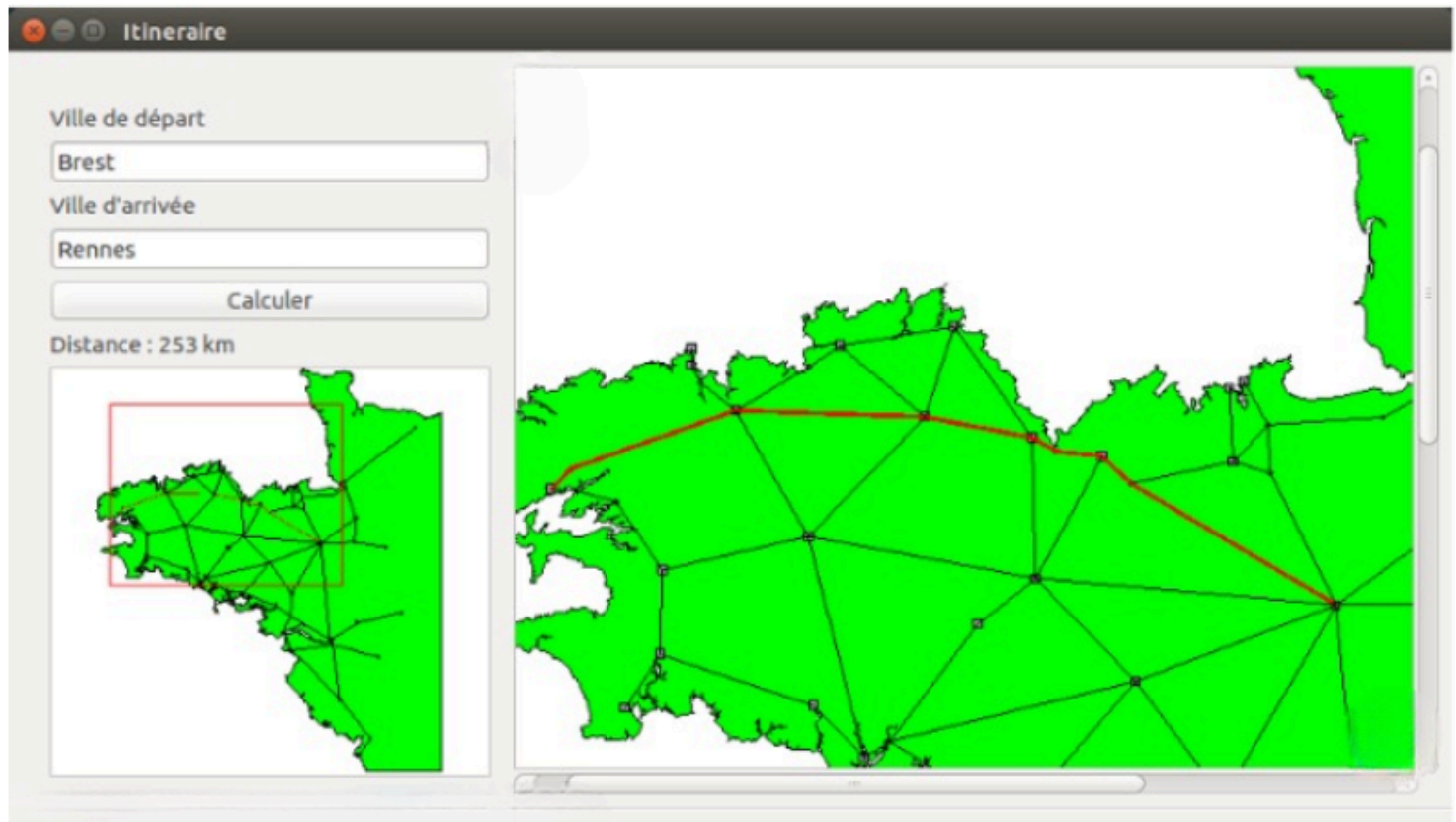b. Implementation of the **SceneCarte** <class>
At this part of the project, we have data ready to be displayed 😃. All demanded <class> should be ready. So we could implement all methods of the **SceneCarte** <class>.

At the end, we should have something similar to this:

c. Implementation of the **MiniView** <class> & **MainView** <class>
Here we will add the zoom effect and the viewport drawing.

d. Implement additional functionalities
- Mouse tracking : it shows the geographical coordinate of the mouse pointer to the map. This functionality will need us to add geoCoordsSlot(QPointF*) to MainWindow <class>, geoCoordsSignal(QPointF* p) to MainView <class>.
- Add the North direction image: it will require the implementation of northOrientation(): int method in MainView <class>.
- Display the scale: it shows the scale on the right-bottom of the MainView.

Final view of RoutePlanner