

Tristan Leduc

Victor Perez

Hugo Robert

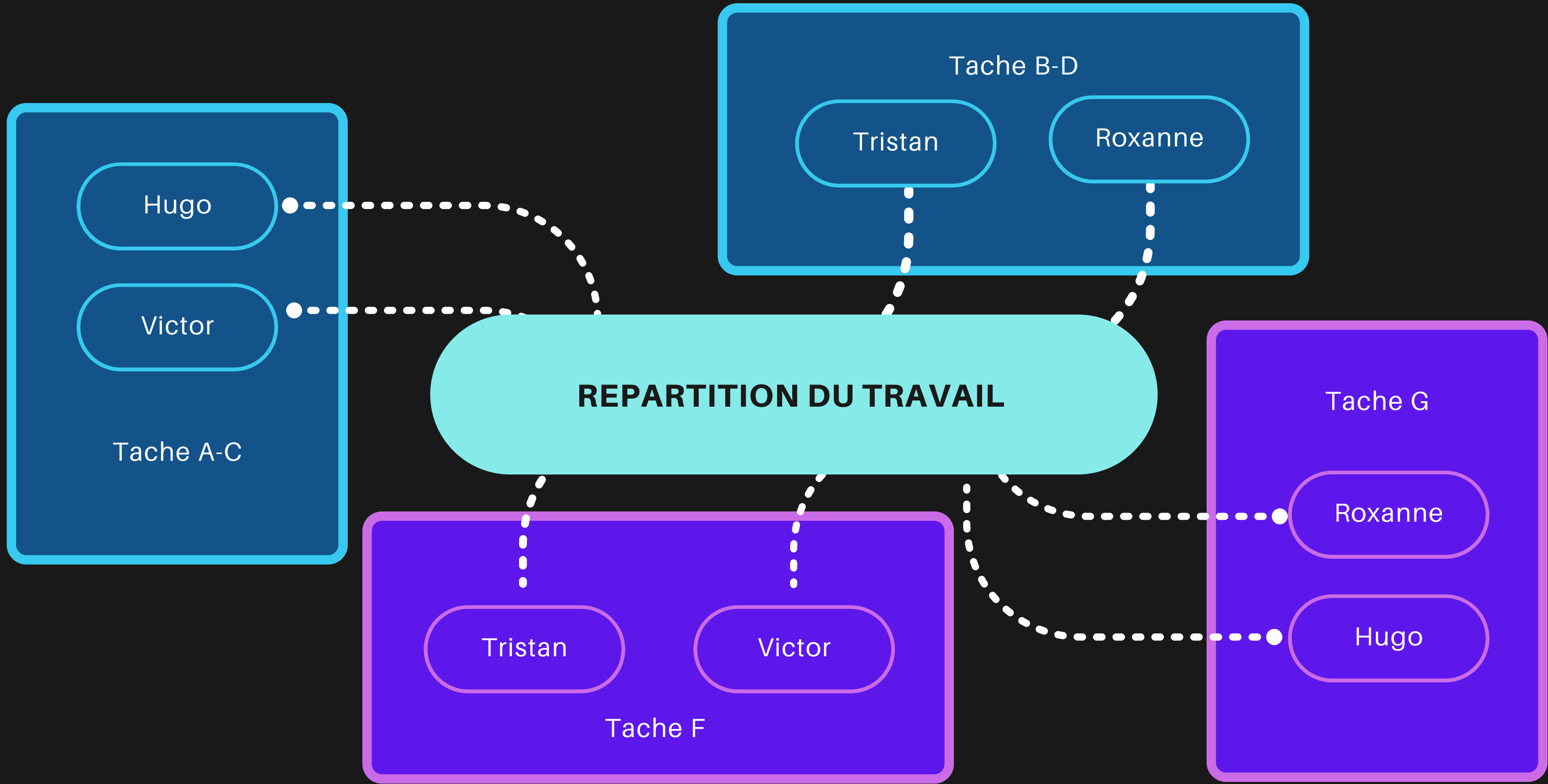
Roxanne Xu

Projet de bases de données
conception de l'application



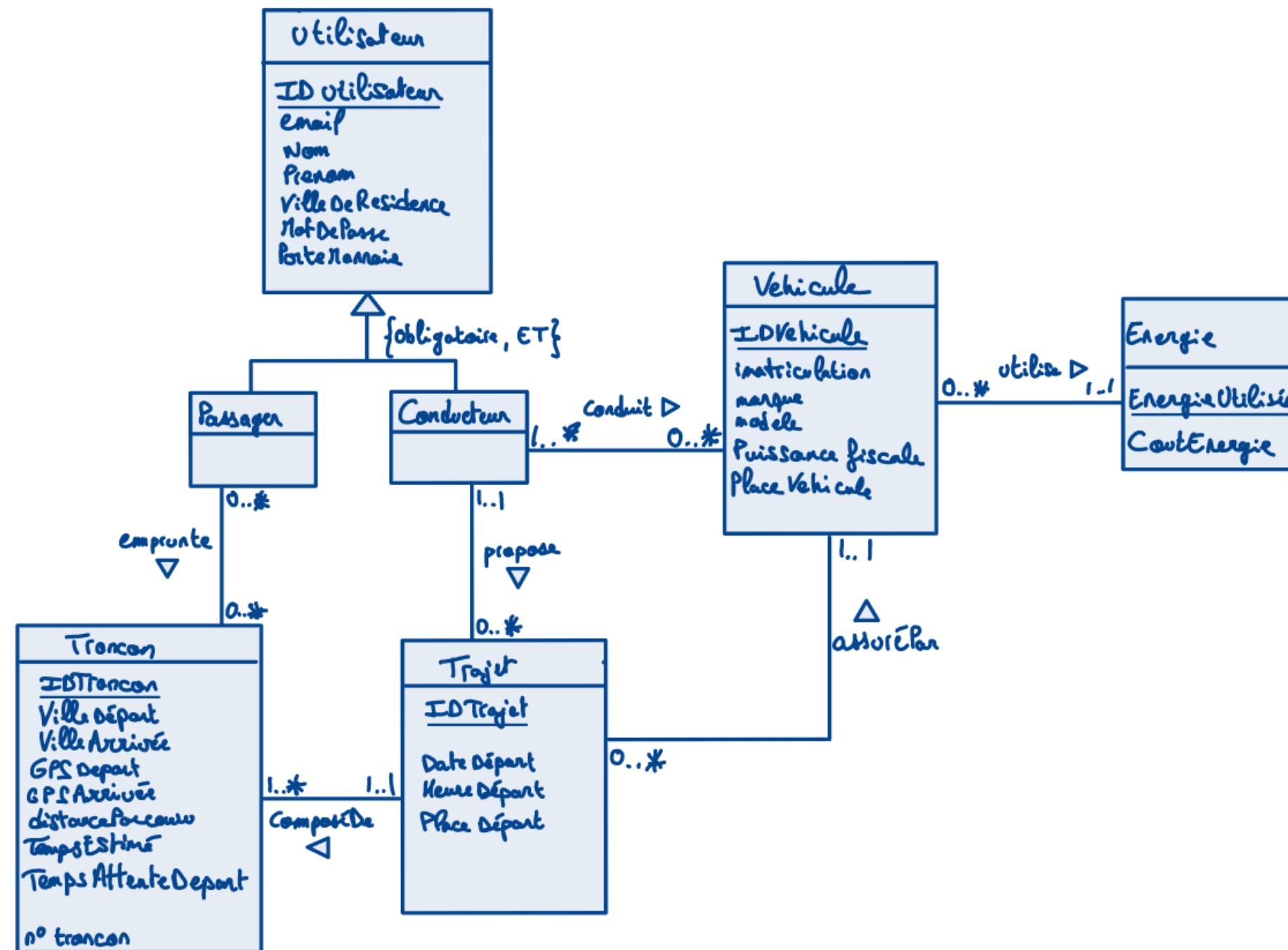
 **VerbiageVoiture**

Stratégie organisationnelle



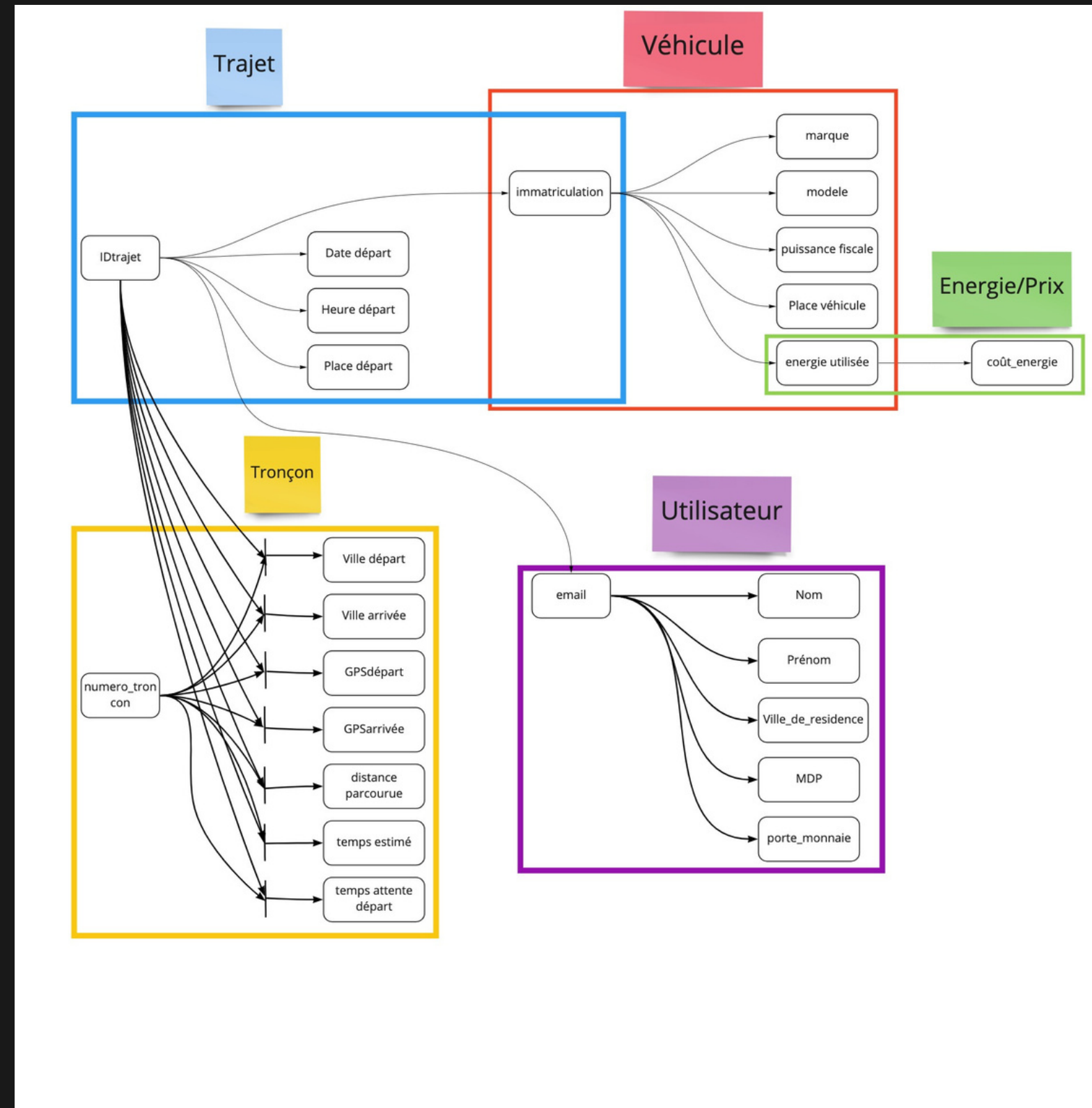
Analyse de la structure de BD

Schéma Entités/Associations

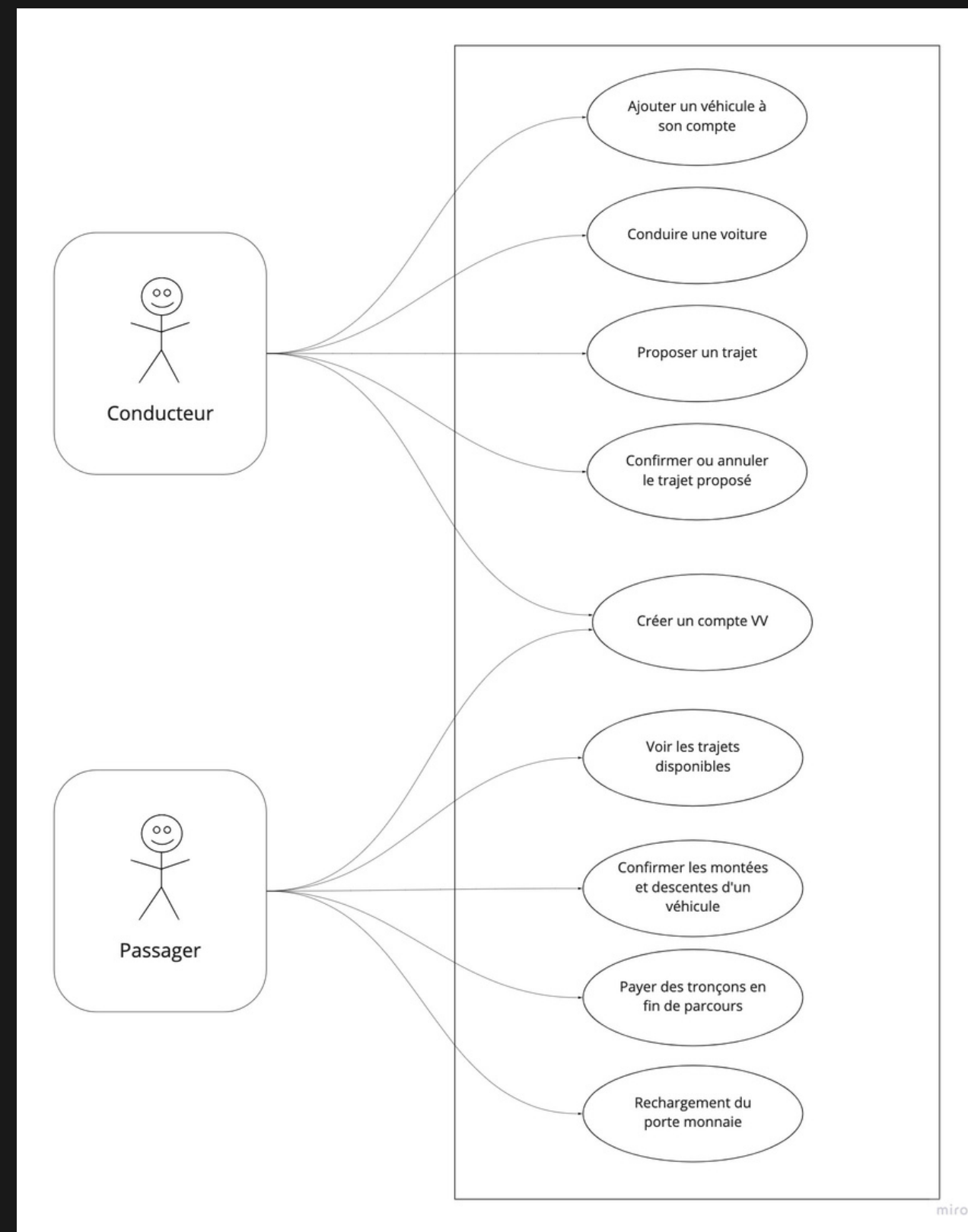


Analyse de la structure de BD

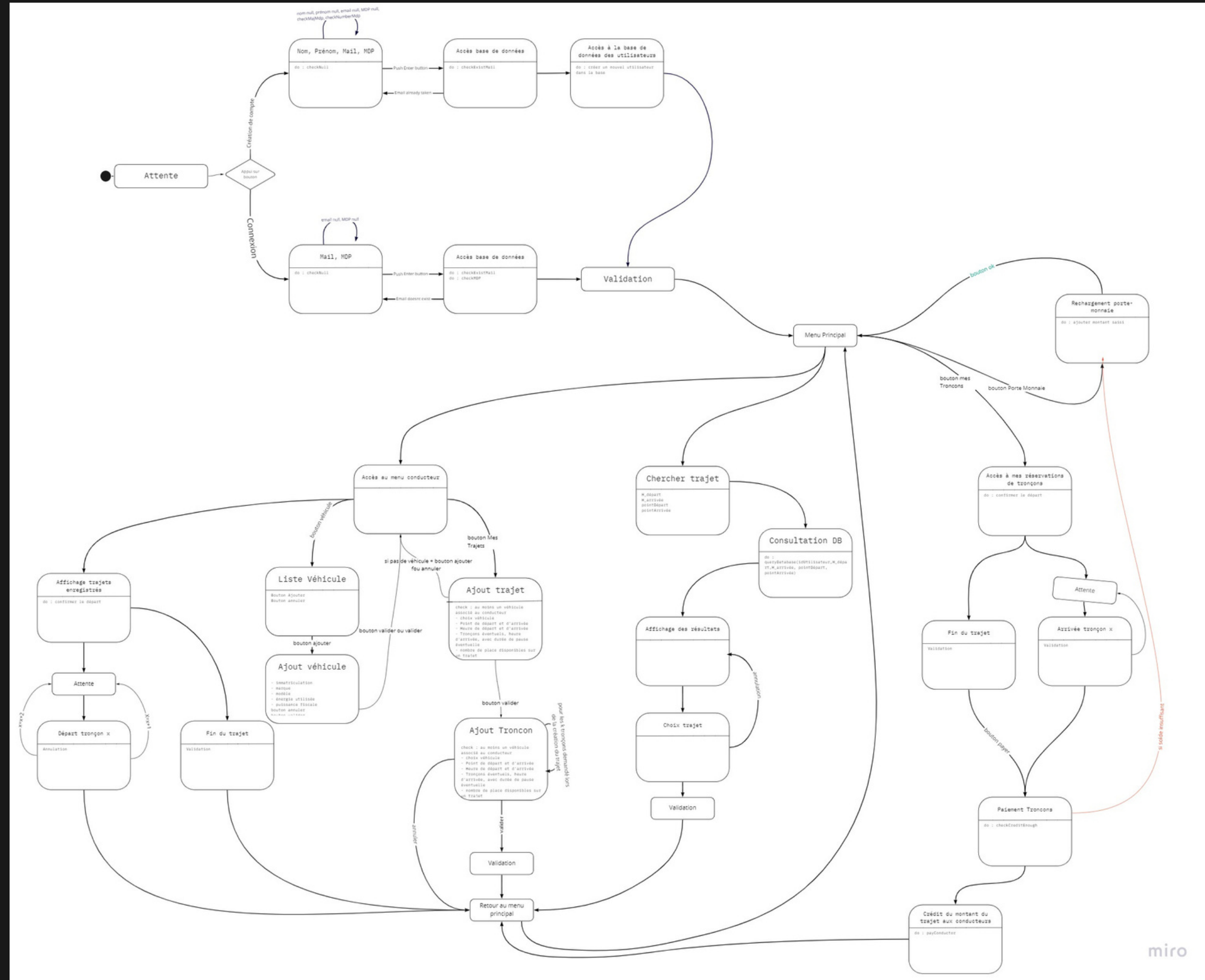
Schéma relationnel



Analyse fonctionnelle de l'app

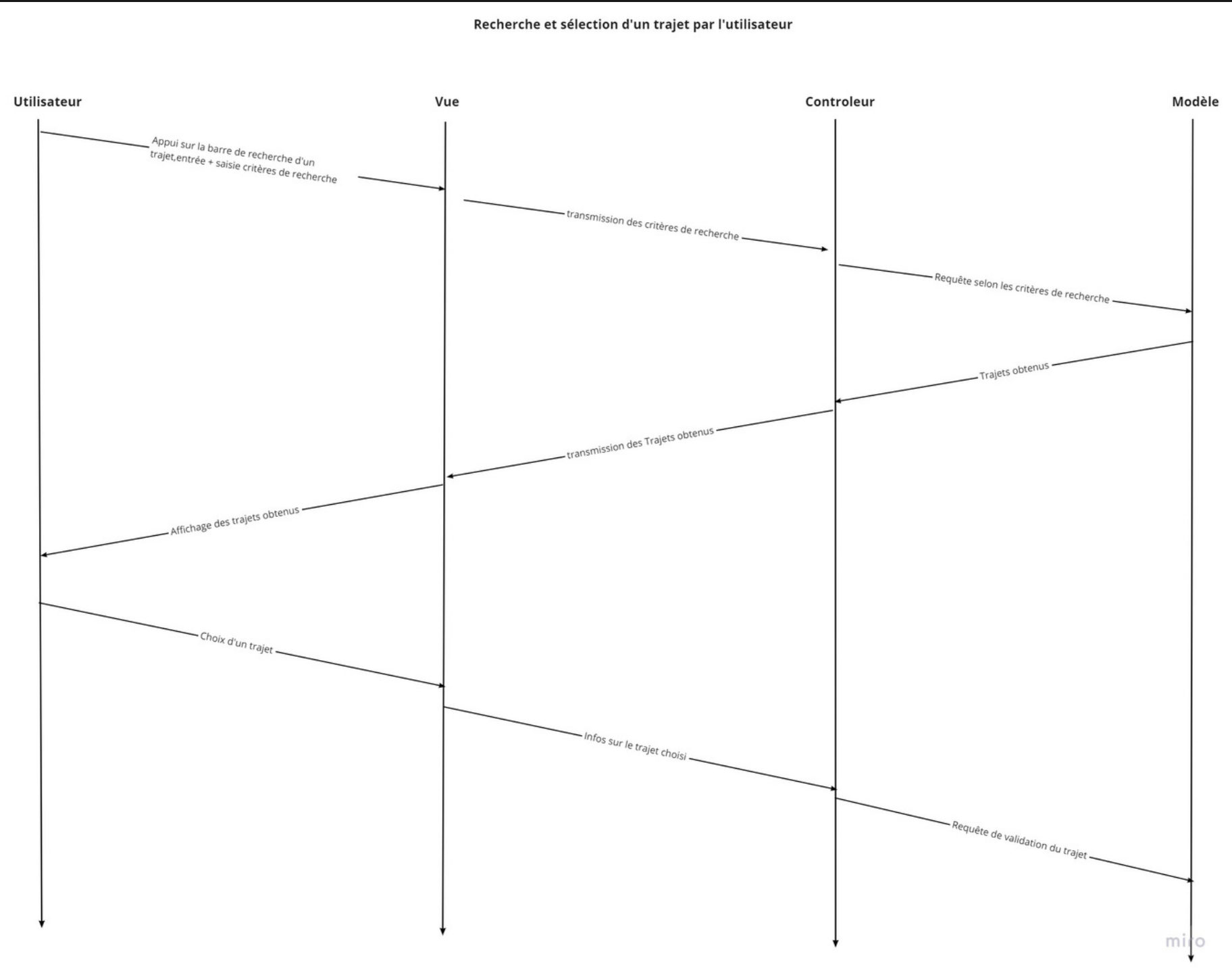
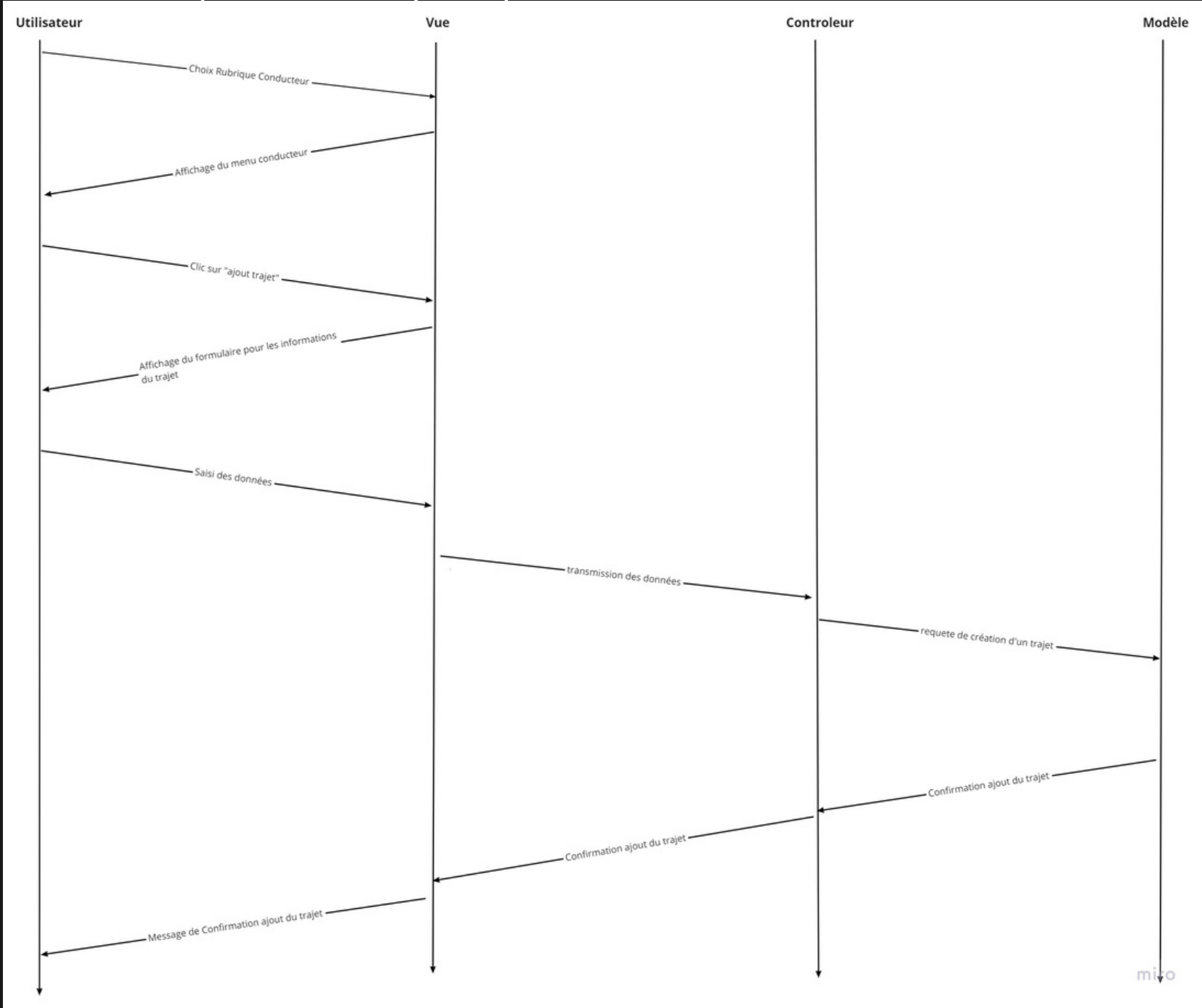


Analyse fonctionnelle de l'app



Analyse fonctionnelle de l'app

Ajout trajet par un conducteur



Implémentation des accès à la Base De Donnée

MyConnection	
getMyVehicule()	ArrayList<String>
deleteTrajetwithTroncon(int)	boolean
addEmprunte(int, int)	boolean
validerDescenteTroncon(int, int)	boolean
coutTroncon(int, int)	float
ajoutTrajet(int, String, String, Timestamp, Timestamp, ArrayList<AjoutTroncon>)	boolean
CheckEmail(String)	boolean
RechargerSolde(float, String)	boolean
getMyVehicule(String)	ArrayList<String>
getTronconEmprunte(String)	ArrayList<int[]>
getTronconEmprunte()	ArrayList<int[]>
validerMonteeTroncon(int, int)	boolean
deleteEmprunte(int)	boolean
AfficherSolde(String)	String
deleteEmprunte(int, int)	boolean
addEmprunte(int, int, String)	boolean
deleteTroncon(int, int)	boolean
RechargerSolde(float)	boolean
addVehicule(String, String, String, String, int, int, String)	boolean
addVehicule(String, String, String, int, int, String)	boolean
deleteTrajet(int)	boolean
creerUtilisateur(String, String, String, String, String)	boolean
getMyTrajet()	ArrayList<String[]>
CheckEmailAndMDP(String, String)	boolean
getNumberTroncon(int)	int
addTrajet(int, String, String, Timestamp, Timestamp)	int
getMyTrajet(String)	ArrayList<String[]>
ajoutTrajet(int, String, Timestamp, Timestamp, ArrayList<AjoutTroncon>)	boolean
findTrajet(String, String)	ArrayList<String[]>
addTroncon(int, int, String, String, String, String, int, int)	int
AfficherSolde()	String
closeConnection()	void

UtilisateurController	
RechargerSolde(float)	boolean
RechargerSolde(float, String)	boolean
creerUtilisateur(String, String, String, String, String)	boolean
CheckEmail(String)	boolean
AfficherSolde()	String
AfficherSolde(String)	String
CheckEmailAndMDP(String, String)	boolean

TronconController	
deleteEmprunte(int, int)	boolean
getTronconEmprunte(String)	ArrayList<int[]>
calculeDistanceTroncon(String, String)	int
coutTroncon(int, int)	float
validerMonteeTroncon(int, int)	boolean
addTroncon(int, int, String, String, String, String, int, int)	int
addEmprunte(int, int, String)	boolean
deleteTroncon(int, int)	boolean
deleteEmprunte(int)	boolean
validerDescenteTroncon(int, int)	boolean
getNumberTroncon(int)	int

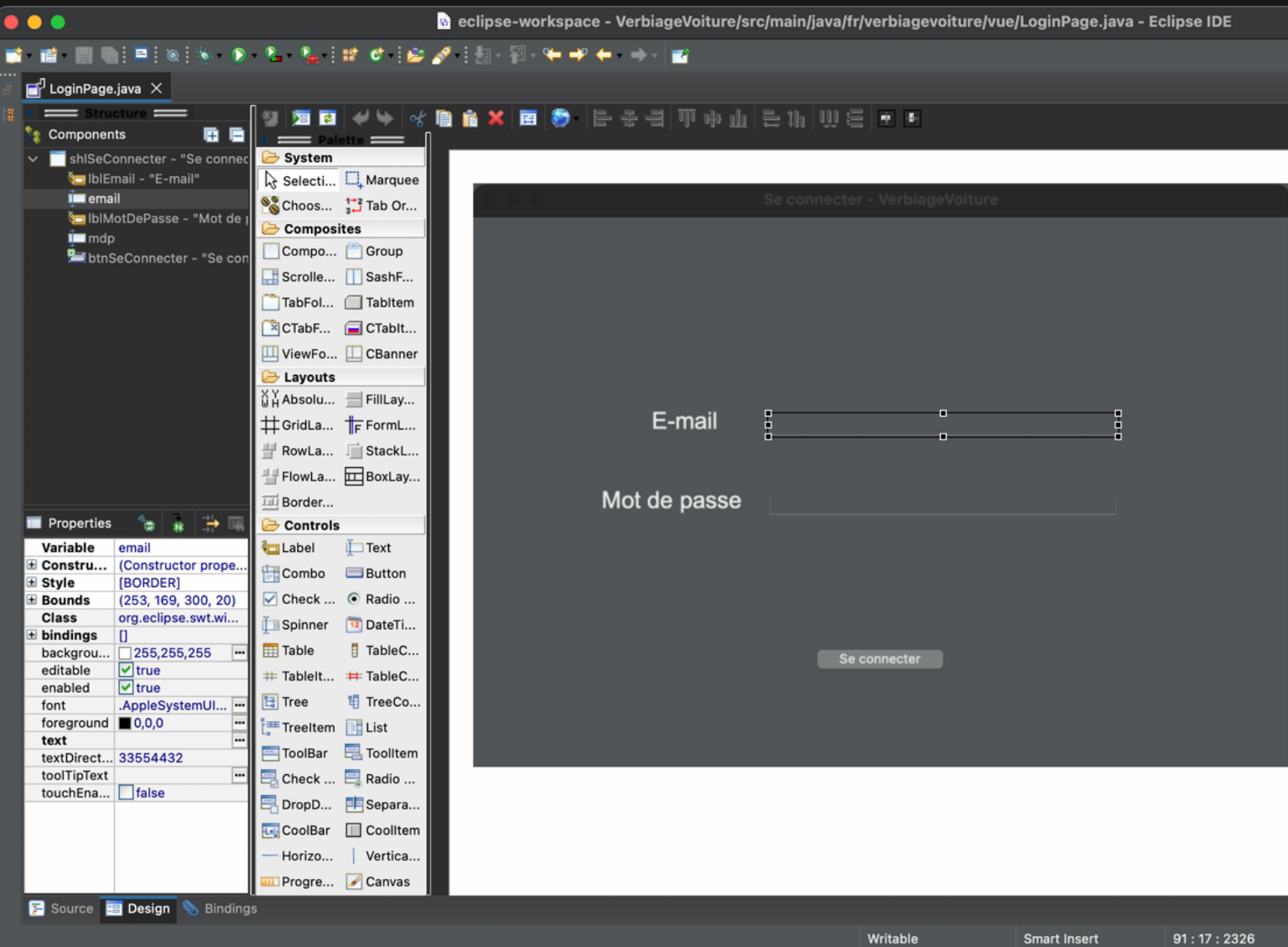
TrajetController	
addTrajet(int, String, String, Timestamp, Timestamp)	int
deleteTroncon(int, int)	boolean
getMyTrajet(String)	ArrayList<String[]>
addTroncon(int, int, String, String, String, String, int, int)	int
findTrajet(String, String)	ArrayList<String[]>
deleteTrajet(int)	boolean

VehiculeController	
getMyVehicule(String)	ArrayList<String>
VehiculeAlreadyExist(String)	boolean
addVehicule(String, String, String, String, int, int, String)	boolean

```
23 // Méthode qui crée une connexion à la BD
24 public MyConnection() {
25     try{
26         // Chargement du driver Oracle
27         System.out.print("Loading Oracle driver... ");
28         DriverManager.registerDriver(new oracle.jdbc.driver.OracleDriver());
29         System.out.println("loaded");
30
31         // Connection à la BD
32         System.out.print("Connecting to the database... ");
33         this.conn = (Connection) DriverManager.getConnection(URL, USERNAME, PASSWD);
34         System.out.println("connected");
35
36         // Demarrage de la transaction (implicite)
37         this.conn.setAutoCommit(false); // Pas de autocommit
38         this.conn.setTransactionIsolation(Connection.TRANSACTION_SERIALIZABLE);
39     } catch (SQLException e) {
40         System.err.println("failed");
41         e.printStackTrace(System.err);
42         this.conn = null;
43     }
44     user = new UtilisateurController(conn);
45     vehicule = new VehiculeController(conn);
46     troncon = new TronconController(conn);
47     trajet = new TrajetController(conn);
48     energie = new EnergieController(conn);
49 }
50
```

```
51 //fermeture de la connexion
52 public void closeConnection() throws SQLException {
53     try {
54         System.out.print("closing connection to the database... ");
55         conn.close();
56         System.out.println("closed ");
57     } catch (SQLException e) {
58         System.err.println("failed");
59         e.printStackTrace(System.err);
60         this.conn = null;
61     }
62 }
```


Interface graphique



```
Label lblMotDePasse = new Label(shlSeConnector, SWT.NONE);
lblMotDePasse.setText("Mot de passe");
lblMotDePasse.setFont(SWTResourceManager.getFont("Arial", 20, SWT.NORMAL));
lblMotDePasse.setAlignment(SWT.CENTER);
lblMotDePasse.setBounds(106, 232, 129, 29);

mdp = new Text(shlSeConnector, SWT.BORDER | SWT.PASSWORD);
mdp.setBounds(253, 237, 300, 20);

Button btnSeConnector = new Button(shlSeConnector, SWT.NONE);
btnSeConnector.addMouseListener(new MouseAdapter() {
    @Override
    public void mouseUp(MouseEvent e) {
        if (Login()) { //if connection success
            ChangeWindow();
            MenuPrincipal window = new MenuPrincipal(myco);
            window.open();
        }
        else {
            System.out.println("connexion impossible : adresse mail ou mot de passe incorrect");
            //TODO : afficher un message d'erreur du type (creation impossible car...)
        }
    }
});
btnSeConnector.setBounds(289, 368, 120, 27);
btnSeConnector.setText("Se connecter");

protected boolean Login() {
    return myco.CheckEmailAndMDP(email.getText(), mdp.getText());
}
```

Bilan du projet

