app-routing.module.ts

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import {TaxisComponent} from './components/taxis/taxis.component';
import {LocationsComponent} from
'./components/locations/locations.component';
import {ClientsComponent} from './components/clients/clients.component';
import {AdressesComponent} from
'./components/adresses/adresses.component';
import {NewTaxiComponent} from "./components/newtaxi/newtaxi.component";
import {NewClientComponent} from
"./components/newclient/newclient.component";
import {NewAdresseComponent} from
"./components/newadresse/newadresse.component";
import {NewLocationComponent} from
"./components/newlocation/newlocation.component";
import {EditTaxiComponent} from
"./components/edittaxi/edittaxi.component";
import {EditClientComponent} from
"./components/editclient/editclient.component";
import {EditAdresseComponent} from
"./components/editadresse/editadresse.component";
import {EditLocationComponent} from
"./components/editlocation/editlocation.component";


import {HomeComponent} from './components/home/home.component';



export const routes: Routes = [
  {path: 'taxis', component: TaxisComponent},
  {path: 'locations', component: LocationsComponent},
  {path: 'clients', component: ClientsComponent},
  {path: 'clients/:idtaxi', component: ClientsComponent},
  {path: 'adresses', component: AdressesComponent},
  {path:'newtaxi',component:NewTaxiComponent},
  {path:"newclient",component:NewClientComponent},
  {path:"newadresse",component:NewAdresseComponent},
  {path:"newlocation",component:NewLocationComponent},
  {path:"edittaxi/:id",component:EditTaxiComponent},
  {path:"editclient/:id",component:EditClientComponent},
  {path:"editadresse/:id",component:EditAdresseComponent},
  {path:"editlocation/:id",component:EditLocationComponent},
  {path: '', component: HomeComponent}
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
```

```
  exports: [RouterModule]
})

export class AppRoutingModule { }
```

app.component.html

```
<app-main-menu></app-main-menu>
<router-outlet ></router-outlet>
```

app.component.spec.ts

```
import { TestBed } from '@angular/core/testing';
import { AppComponent } from './app.component';

describe('AppComponent', () => {
  beforeEach(async () => {
    await TestBed.configureTestingModule({
      imports: [AppComponent],
    }).compileComponents();
  });

  it('should create the app', () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.componentInstance;
    expect(app).toBeTruthy();
  });

  it(`should have the 'HaniniAngularProject' title`, () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.componentInstance;
    expect(app.title).toEqual('HaniniAngularProject');
  });

  it('should render title', () => {
    const fixture = TestBed.createComponent(AppComponent);
    fixture.detectChanges();
    const compiled = fixture.nativeElement as HTMLElement;
    expect(compiled.querySelector('h1')?.textContent).toContain('Hello,
HaniniAngularProject');
  });
});
```

app.component.ts

```
import { Component } from '@angular/core';
import { CommonModule } from '@angular/common';
import { RouterOutlet } from '@angular/router';
import {MainMenuComponent} from "./components/main-menu/main-
menu.component";

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
})
export class AppComponent {
  title = 'HaniniAngularProject';
}
```

app.module.ts

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppRoutingModule } from './app-routing.module';
import {FormsModule, ReactiveFormsModule} from '@angular/forms';
import { HttpClientModule } from '@angular/common/http';
import {AppComponent} from "./app.component";
import {TaxisComponent} from "./components/taxis/taxis.component";
import {NewTaxiComponent} from "./components/newtaxi/newtaxi.component";
import {LocationsComponent} from
"./components/locations/locations.component";
import {ClientsComponent} from "./components/clients/clients.component";
import {AdressesComponent} from
"./components/adresses/adresses.component";
import {HomeComponent} from "./components/home/home.component";
import {MainMenuComponent} from "./components/main-menu/main-
menu.component";
import {EditTaxiComponent} from
"./components/edittaxi/edittaxi.component";
import { NewClientComponent } from
'./components/newclient/newclient.component';
import { NewAdresseComponent } from
'./components/newadresse/newadresse.component';
import {NewLocationComponent} from
"./components/newlocation/newlocation.component";
import {EditLocationComponent} from
"./components/editlocation/editlocation.component";
import { EditClientComponent } from
'./components/editclient/editclient.component';
import { EditAdresseComponent } from
'./components/editadresse/editadresse.component';

@NgModule({
  imports: [
    BrowserModule,
    AppRoutingModule,
    HttpClientModule,
    FormsModule,
    ReactiveFormsModule,
  ],

  declarations: [
    AppComponent,
    MainMenuComponent,
    HomeComponent,
    TaxisComponent,
    LocationsComponent,
    ClientsComponent,
    AdressesComponent,
```

```
    NewTaxiComponent,
    EditTaxiComponent,
    NewLocationComponent,
    EditLocationComponent,
    NewClientComponent,
    NewAdresseComponent,
    EditClientComponent,
    EditAdresseComponent

  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule {}
```

adresses.component.html

```
<div class="container">
  <div class="card mt-4">
    <div class="card-body">
      <h3 class="card-title">Adresse</h3>

      <form #f="ngForm" (ngSubmit)="getAdresseByLocalite(f.value)"
class="mb-3">
        <div class="input-group">
            <input type="text" class="form-control" ngModel
name="localite" placeholder="Entrez la localitÃ©">
          <button type="submit" class="btn btn-
primary">Rechercher</button>
        </div>
      </form>

      <div class="d-flex justify-content-between align-items-center">
        <button class="btn btn-primary" (click)="showAll()">Afficher
toutes les adresses</button>
        <button class="btn btn-success" (click)="onNewAdresse()">Ajouter
une nouvelle adresse</button>
      </div>
    </div>
  </div>

  <ng-container *ngIf="adresses">
    <ng-container *ngIf="adresses.length > 0; else inconnu">
      <div class="card mt-4">
        <div class="card-body">
          <h3 class="card-title">Adresses trouvÃ©es :</h3>

          <table class="table table-striped">
            <thead>
              <tr>
                <th scope="col">ID</th>
                <th scope="col">Code Postal</th>
                <th scope="col">LocalitÃ©</th>
                <th scope="col">Rue</th>
                <th scope="col">NumÃ©ro</th>
                <th scope="col">Actions</th>
              </tr>
            </thead>
            <tbody>
              <tr *ngFor="let adresse of adresses">
                <td>{{ adresse.idadresse }}</td>
                <td>{{ adresse.cp }}</td>
                <td>{{ adresse.localite }}</td>
                <td>{{ adresse.rue }}</td>
                <td>{{ adresse.num }}</td>
```

```
                <td>
                  <button (click)="onEdit(adresse)" class="btn btn-
primary btn-sm me-2">
                    <span class="fa fa-edit"></span> Modifier
                  </button>
                  <button (click)="onDelete(adresse)" class="btn btn-
danger btn-sm">
                    <span class="fa fa-trash-o"></span> Supprimer
                  </button>
                </td>
              </tr>
            </tbody>
          </table>
        </div>
      </div>
    </ng-container>

    <ng-template #inconnu>
      <div class="card mt-4">
        <div class="card-body">
          <h3 class="card-title">Aucune adresse trouvÃ©e</h3>
        </div>
      </div>
    </ng-template>
  </ng-container>
</div>
```

adresses.component.spec.ts

```
import { ComponentFixture, TestBed } from '@angular/core/testing';

import { AdressesComponent } from './adresses.component';

describe('AdressesComponent', () => {
  let component: AdressesComponent;
  let fixture: ComponentFixture<AdressesComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      imports: [AdressesComponent]
    })
    .compileComponents();

    fixture = TestBed.createComponent(AdressesComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

adresses.component.ts

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { Adresse } from '../../entities/adresse.entities';
import { AdressesService } from '../../services/adresses.service';

@Component({
  selector: 'app-adresses',
  templateUrl: './adresses.component.html',
})
export class AdressesComponent implements OnInit {

  adresses?: Adresse[];

  constructor(private adressesService: AdressesService, private router:
Router) {}

  ngOnInit(): void {
  }

  onSearch(value: { id: number }) {
    this.adressesService.getAdresse(value.id).subscribe({
      next: data => {
        this.adresses = data ? [data] : [];
      }
    });
  }

  getAdresseByLocalite(value: { localite: string }) {
    this.adressesService.getAdresseByLocalite(value.localite).subscribe(
{
      next: data => {
        this.adresses = data;
      }
    });
  }

  showAll() {
    this.adressesService.getAllAdresses().subscribe({
      next: data => {
        this.adresses = data;
      }
    });
  }

  onEdit(adresse: Adresse) {
    this.router.navigate(['/editadresse', adresse.idadresse]);
  }
```

```
  onDelete(adresse: Adresse) {
    const confirmation = confirm('Are you sure you want to delete?');

    if (confirmation) {
      this.adressesService.deleteAdresse(adresse).subscribe({
        next: () => {
          const index = this.adresses?.indexOf(adresse);
          if (index !== undefined && index !== -1) {
            this.adresses?.splice(index, 1);
          }
          (window as any).sendAlert('success', 'Adresse deleted!');
        }
      });
    }
  }

  onNewAdresse() {
    this.router.navigate(['newadresse']);
  }
}
```

clients.component.html

```
<div class="container">
  <div class="card mt-4">
    <div class="card-body">
      <h3 class="card-title">Client</h3>

      <form #f="ngForm"
(ngSubmit)="getClientByNomAndPrenomAndTel(f.value)" class="mb-3">
        <div class="input-group">
          <input type="text" class="form-control" ngModel name="nom"
placeholder="Nom du client">
          <input type="text" class="form-control" ngModel name="prenom"
placeholder="PrÃ©nom du client">
          <input type="text" class="form-control" ngModel name="tel"
placeholder="NumÃ©ro de tÃ©lÃ©phone">
          <button type="submit" class="btn btn-
primary">Rechercher</button>
        </div>
      </form>


      <div class="d-flex justify-content-between align-items-center">
        <button class="btn btn-primary" (click)="showAll()">Afficher
tous les clients</button>
        <button class="btn btn-success" (click)="onNewClient()">Ajouter
un nouveau client</button>
      </div>
    </div>
  </div>

  <ng-container *ngIf="clients">
    <ng-container *ngIf="clients.length > 0; else inconnu">
      <div class="card mt-4">
        <div class="card-body">
          <h3 class="card-title">Clients trouvÃ©s :</h3>

          <table class="table table-striped">
            <thead>
              <tr>
                <th scope="col">ID</th>
                <th scope="col">Mail</th>
                <th scope="col">Nom</th>
                <th scope="col">PrÃ©nom</th>
                <th scope="col">TÃ©lÃ©phone</th>
                <th scope="col">Locations</th>
                <th scope="col">Actions</th>
              </tr>
            </thead>
            <tbody>
```

```
              <tr *ngFor="let client of clients">
                <td>{{ client.idclient }}</td>
                <td>{{ client.mail }}</td>
                <td>{{ client.nom }}</td>
                <td>{{ client.prenom }}</td>
                <td>{{ client.tel }}</td>
                <td>
                  <button
(click)="getLocationsForClient(client.idclient)" class="btn btn-primary
btn-sm">
                    <span class="fa fa-users"></span>
                    {{ selectedClientId === client.idclient ? 'Cacher' :
'Afficher' }} Locations
                  </button>
                </td>
                <td>
                  <button (click)="onEdit(client)" class="btn btn-
primary btn-sm me-2">
                    <span class="fa fa-edit"></span> Modifier
                  </button>
                  <button (click)="onDelete(client)" class="btn btn-
danger btn-sm">
                    <span class="fa fa-trash-o"></span> Supprimer
                  </button>
                </td>
              </tr>
            </tbody>
          </table>
        </div>
      </div>
    </ng-container>

    <ng-template #inconnu>
      <div class="card mt-4">
        <div class="card-body">
          <h3 class="card-title">Aucun client trouvÃ©</h3>
        </div>
      </div>
    </ng-template>
  </ng-container>
</div>
```

clients.component.spec.ts

```
import { ComponentFixture, TestBed } from '@angular/core/testing';

import { ClientsComponent } from './clients.component';

describe('ClientsComponent', () => {
  let component: ClientsComponent;
  let fixture: ComponentFixture<ClientsComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      imports: [ClientsComponent]
    })
    .compileComponents();

    fixture = TestBed.createComponent(ClientsComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

clients.component.ts

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute, Router } from '@angular/router';
import { Client } from '../../entities/client.entities';
import { ClientsService } from '../../services/clients.service';
import { TaxisService } from '../../services/taxis.service';

@Component({
  selector: 'app-clients',
  templateUrl: './clients.component.html',
})
export class ClientsComponent implements OnInit {

  clients?: Client[];
  loading = false;
  idtaxi: number;
  selectedClientId?: number;


  constructor(private clientsService: ClientsService, private router:
Router, private activeroute: ActivatedRoute, private taxisService:
TaxisService) {
    this.idtaxi = this.activeroute.snapshot.params.taxiId;
      if (this.idtaxi) {
        this.loading = true;
        this.taxisService.getClientsForTaxi(this.idtaxi).subscribe({
          next: data => {
            this.clients = data;
          },
          complete: () => {
            this.loading = false;
          }
        });
      }
  }

  ngOnInit(): void {
  }

  onSearch(value: { id: number }) {
    this.clientsService.getClient(value.id).subscribe({
      next: data => {
        this.clients = data ? [data] : [];
      }
    });
  }

    getClientByNomAndPrenomAndTel(value: { nom: string, prenom: string,
tel: string }) {
```

```
      this.loading = true;

      this.clientsService.getClientByNomAndPrenomAndTel(value.nom,
value.prenom, value.tel).subscribe({
        next: data => {
          this.clients = data ? [data] : [];
        },
        complete: () => {
          this.loading = false;
        }
      });
    }

    getLocationsForClient(clientId: number): void {
      if (clientId) {
        this.loading = true;
        this.router.navigate(['/locations', { clientId }]);
      }
    }


  showAll() {
    this.clientsService.getAllClients().subscribe({
      next: data => {
        this.clients = data;
      }
    });
  }


  onEdit(client: Client) {
    this.router.navigate(['/editclient', client.idclient]);
  }


  onDelete(client: Client) {
    const confirmation = confirm('Are you sure you want to delete?');

    if (confirmation) {
      this.clientsService.deleteClient(client).subscribe({
        next: () => {
          const index = this.clients?.indexOf(client);
          if (index !== undefined && index !== -1) {
            this.clients?.splice(index, 1);
          }
          (window as any).sendAlert('success', 'Client deleted!');
        }
      });
    }
```

```
  }

  onNewClient() {
    this.router.navigate(['newclient']);
  }
}
```

editadresse.component.html

```
<div class="container">
  <form [formGroup]="adresseFormGroup" *ngIf="adresseFormGroup">
    <div class="form-group">
      <label>ID</label>
      <input type="text" class="form-control"
formControlName="idadresse" readonly>
    </div>

    <div class="form-group">
      <label>Code postal</label>
      <input type="text" class="form-control" formControlName="cp"
             [ngClass]="{'is-invalid': submitted &&
adresseFormGroup.controls.cp.errors}">
      <div *ngIf="submitted && adresseFormGroup.controls.cp.errors"
class="invalid-feedback">
        <div *ngIf="adresseFormGroup.controls.cp.errors.required">
          Code postal obligatoire
        </div>
      </div>
    </div>

    <div class="form-group">
      <label>LocalitÃ©</label>
      <input type="text" class="form-control" formControlName="localite"
             [ngClass]="{'is-invalid': submitted &&
adresseFormGroup.controls.localite.errors}">
      <div *ngIf="submitted &&
adresseFormGroup.controls.localite.errors" class="invalid-feedback">
        <div *ngIf="adresseFormGroup.controls.localite.errors.required">
          LocalitÃ© obligatoire
        </div>
      </div>
    </div>

    <div class="form-group">
      <label>Rue</label>
      <input type="text" class="form-control" formControlName="rue"
             [ngClass]="{'is-invalid': submitted &&
adresseFormGroup.controls.rue.errors}">
      <div *ngIf="submitted && adresseFormGroup.controls.rue.errors"
class="invalid-feedback">
        <div *ngIf="adresseFormGroup.controls.rue.errors.required">
          Rue obligatoire
        </div>
      </div>
    </div>

    <div class="form-group">
```

```
      <label>NumÃ©ro</label>
      <input type="text" class="form-control" formControlName="num"
             [ngClass]="{'is-invalid': submitted &&
adresseFormGroup.controls.num.errors}">
      <div *ngIf="submitted && adresseFormGroup.controls.num.errors"
class="invalid-feedback">
        <div *ngIf="adresseFormGroup.controls.num.errors.required">
          NumÃ©ro obligatoire
        </div>
      </div>
    </div>

    <nav class="navbar navbar-expand-sm bg-light navbar-light">
      <ul class="navbar navbar-nav">
        <li class="nav-item">
          <button (click)="onUpdateAdresse()" class="btn btn-success">
            Save
          </button>
        </li>
      </ul>
    </nav>
  </form>
</div>
```

editadresse.component.spec.ts

```
import { ComponentFixture, TestBed } from '@angular/core/testing';

import { EditAdresseComponent } from './editadresse.component';

describe('EditAdresseComponent', () => {
  let component: EditAdresseComponent;
  let fixture: ComponentFixture<EditAdresseComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      imports: [EditAdresseComponent]
    })
    .compileComponents();

    fixture = TestBed.createComponent(EditAdresseComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

editadresse.component.ts

```
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { AdressesService } from '../../services/adresses.service';
import { ActivatedRoute } from '@angular/router';
import { Router } from '@angular/router';

@Component({
  selector: 'app-editadresse',
  templateUrl: './editadresse.component.html',
})
export class EditAdresseComponent implements OnInit {
  adresseFormGroup?: FormGroup;
  submitted = false;
  idadresse: number;

  constructor(private fb: FormBuilder, private adressesService:
AdressesService, private route: ActivatedRoute, private router: Router)
{
    this.idadresse = route.snapshot.params.id;
  }

  ngOnInit(): void {
    this.adressesService.getAdresse(this.idadresse).subscribe(adresse =>
{
    this.adresseFormGroup = this.fb.group({
        idadresse: [adresse.idadresse],
        cp: [adresse.cp, [Validators.required, Validators.pattern('^[0-
9]+$')]],
        localite: [adresse.localite, [Validators.required,
Validators.pattern('^[a-zA-ZÃ€-Å¾Ã€-Ã¿ ]*$')]],
        rue: [adresse.rue, [Validators.required,
Validators.pattern('^[a-zA-ZÃ€-Å¾Ã€-Ã¿ ]*$')]],
        num: [adresse.num, [Validators.required]],
        });
      }, error => {
          console.error('Error fetching client details:', error);
      });

  }

  onUpdateAdresse(): void {
    this.submitted = true;
    if (this.adresseFormGroup?.invalid) { return; }


this.adressesService.updateAdresse(this.adresseFormGroup?.value).subscri
be(
      data => {
```

```
        alert('Mise Ã  jour rÃ©ussie');
        this.router.navigate(['/adresses']);
      },
      err => {
        alert(err.headers.get('error'));
      }
    );
  }
}
```

editclient.component.html

```
<div class="container">
  <form [formGroup]="clientFormGroup" *ngIf="clientFormGroup">
    <div class="form-group">
      <label>ID</label>
      <input type="text" class="form-control" formControlName="idclient"
readonly>
    </div>

    <div class="form-group">
      <label>Email</label>
      <input type="email" class="form-control" formControlName="mail"
             [ngClass]="{'is-invalid': submitted &&
clientFormGroup.controls.mail.errors}">
      <div *ngIf="submitted && clientFormGroup.controls.mail.errors"
class="invalid-feedback">
        <div *ngIf="clientFormGroup.controls.mail.errors.required">
          Email obligatoire
        </div>
        <div *ngIf="clientFormGroup.controls.mail.errors.email">
          Entrez un email valide
        </div>
      </div>
    </div>

    <div class="form-group">
      <label>Nom</label>
      <input type="text" class="form-control" formControlName="nom"
             [ngClass]="{'is-invalid': submitted &&
clientFormGroup.controls.nom.errors}">
      <div *ngIf="submitted && clientFormGroup.controls.nom.errors"
class="invalid-feedback">
        <div *ngIf="clientFormGroup.controls.nom.errors.required">
          Nom obligatoire
        </div>
      </div>
    </div>

    <div class="form-group">
      <label>PrÃ©nom</label>
      <input type="text" class="form-control" formControlName="prenom"
             [ngClass]="{'is-invalid': submitted &&
clientFormGroup.controls.prenom.errors}">
      <div *ngIf="submitted && clientFormGroup.controls.prenom.errors"
class="invalid-feedback">
        <div *ngIf="clientFormGroup.controls.prenom.errors.required">
          PrÃ©nom obligatoire
        </div>
      </div>
```

```
    </div>

    <div class="form-group">
      <label>TÃ©lÃ©phone</label>
      <input type="tel" class="form-control" formControlName="tel"
             [ngClass]="{'is-invalid': submitted &&
clientFormGroup.controls.tel.errors}">
      <div *ngIf="submitted && clientFormGroup.controls.tel.errors"
class="invalid-feedback">
        <div *ngIf="clientFormGroup.controls.tel.errors.required">
          TÃ©lÃ©phone obligatoire
        </div>
      </div>
    </div>

    <nav class="navbar navbar-expand-sm bg-light navbar-light">
      <ul class="navbar navbar-nav">
        <li class="nav-item">
          <button (click)="onUpdateClient()" class="btn btn-success">
            Save
          </button>
        </li>
      </ul>
    </nav>
  </form>
</div>
```

editclient.component.spec.ts

```
import { ComponentFixture, TestBed } from '@angular/core/testing';

import { EditClientComponent } from './editclient.component';

describe('EditClientComponent', () => {
  let component: EditClientComponent;
  let fixture: ComponentFixture<EditClientComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      imports: [EditClientComponent]
    })
    .compileComponents();

    fixture = TestBed.createComponent(EditClientComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

editclient.component.ts

```
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { ClientsService } from '../../services/clients.service';
import { ActivatedRoute } from '@angular/router';
import { Router } from '@angular/router';



@Component({
  selector: 'app-editclient',
  templateUrl: './editclient.component.html',
})
export class EditClientComponent implements OnInit {
  clientFormGroup?: FormGroup;
  idclient: number
  submitted = false;

  constructor(private fb: FormBuilder, private clientsService:
ClientsService,  private route: ActivatedRoute, private router: Router)
{
    this.idclient = route.snapshot.params.id;
  }

  ngOnInit(): void {

    this.clientsService.getClient(this.idclient).subscribe(client => {
      this.clientFormGroup = this.fb.group({
        idclient: [client.idclient],
        mail: [client.mail, [Validators.required, Validators.email]],
        nom: [client.nom, [Validators.required, Validators.pattern('[a-
zA-Z ]*')]],
        prenom: [client.prenom, [Validators.required,
Validators.pattern('[a-zA-Z ]*')]],
        tel: [client.tel, [Validators.required, Validators.pattern('[0-
9]*'), Validators.minLength(10)]],
      });
    }, error => {
      console.error('Error fetching client details:', error);
    });
  }


  onUpdateClient(): void {
    this.submitted = true;
    if (this.clientFormGroup?.invalid) { return; }


this.clientsService.updateClient(this.clientFormGroup?.value).subscribe(
```

```
      data => {
        alert('Mise Ã  jour rÃ©ussie');
        this.router.navigate(['/clients']);
      },
      err => {
        alert(err.headers.get('error'));
      }
    );
  }
}
```

editlocation.component.html

```
<div class="container">
  <form [formGroup]="locationFormGroup" *ngIf="locationFormGroup">
    <div class="form-group">
      <label>ID</label>
      <input type="text" class="form-control"
formControlName="idlocation" readonly>
    </div>

    <div class="form-group">
      <label>Date de location</label>
      <input type="date" class="form-control" formControlName="dateloc"
             [ngClass]="{'is-invalid': submitted &&
locationFormGroup.controls.dateloc.errors}">
      <div *ngIf="submitted &&
locationFormGroup.controls.dateloc.errors" class="invalid-feedback">
        <div *ngIf="locationFormGroup.controls.dateloc.errors.required">
          Date de location obligatoire
        </div>
      </div>
    </div>

    <div class="form-group">
      <label>KilomÃ©trage total</label>
      <input type="number" class="form-control"
formControlName="kmtotal"
             [ngClass]="{'is-invalid': submitted &&
locationFormGroup.controls.kmtotal.errors}">
      <div *ngIf="submitted &&
locationFormGroup.controls.kmtotal.errors" class="invalid-feedback">
        <div *ngIf="locationFormGroup.controls.kmtotal.errors.required">
          KilomÃ©trage total obligatoire
        </div>
        <div *ngIf="locationFormGroup.controls.kmtotal.errors.min">
          KilomÃ©trage total doit Ãªtre supÃ©rieur Ã  0
        </div>
      </div>
    </div>

    <div class="form-group">
      <label>Acompte</label>
      <input type="number" class="form-control"
formControlName="acompte"
             [ngClass]="{'is-invalid': submitted &&
locationFormGroup.controls.acompte.errors}">
      <div *ngIf="submitted &&
locationFormGroup.controls.acompte.errors" class="invalid-feedback">
        <div *ngIf="locationFormGroup.controls.acompte.errors.required">
          Acompte obligatoire
```

```
        </div>
        <div *ngIf="locationFormGroup.controls.acompte.errors.min">
          Acompte doit Ãªtre supÃ©rieur ou Ã©gal 0
        </div>
      </div>
    </div>

    <div class="row">
      <div class="col-md-6">
        <label for="clientfk">Client</label>
        <select id="clientfk" formControlName="clientfk" class="form-
control">
          <option *ngFor="let client of clients"
[value]="client.idclient">{{ client.mail }}</option>
        </select>
      </div>

      <div class="col-md-6">
        <label for="taxifk">Taxi</label>
        <select id="taxifk" formControlName="taxifk" class="form-
control">
          <option *ngFor="let taxi of taxis" [value]="taxi.idtaxi">{{
taxi.immatriculation }}</option>
        </select>
      </div>

      <div class="col-md-6">
        <label for="adressedepart">Adresse de DÃ©part</label>
        <select id="adressedepart" formControlName="adressedepart"
class="form-control">
          <option *ngFor="let adresse of adresses"
[value]="adresse.idadresse">{{ adresse.rue }} , {{adresse.num}} , {{
adresse.cp}} {{adresse.localite}}</option>
        </select>
      </div>

      <div class="col-md-6">
        <label for="adressefin">Adresse de Fin</label>
        <select id="adressefin" formControlName="adressefin"
class="form-control">
          <option *ngFor="let adresse of adresses"
[value]="adresse.idadresse">{{ adresse.rue }} , {{adresse.num}} , {{
adresse.cp}} {{adresse.localite}}</option>
        </select>
      </div>
    </div>
```

```
    <nav class="navbar navbar-expand-sm bg-light navbar-light">
      <ul class="navbar navbar-nav">
        <li class="nav-item">
          <button (click)="updateLocation()" class="btn btn-success">
            Save
          </button>
        </li>
      </ul>
    </nav>
  </form>
</div>
```

editlocation.component.spec.ts

```
import { ComponentFixture, TestBed } from '@angular/core/testing';

import { EditLocationComponent } from './editlocation.component';


describe('EditLocationComponent', () => {
  let component: EditLocationComponent;
  let fixture: ComponentFixture<EditLocationComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      imports: [EditLocationComponent]
    })
    .compileComponents();

    fixture = TestBed.createComponent(EditLocationComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

editlocation.component.ts

```
import { Component, OnInit } from '@angular/core';
import { AbstractControl, FormBuilder, FormGroup, Validators } from
'@angular/forms';
import { LocationsService } from '../../services/locations.service';
import { ActivatedRoute, Router } from '@angular/router';
import { Client } from '../../entities/client.entities';
import { Taxi } from '../../entities/taxi.entities';
import { Adresse } from '../../entities/adresse.entities';
import { ClientsService } from '../../services/clients.service';
import { TaxisService } from '../../services/taxis.service';
import { AdressesService } from '../../services/adresses.service';
import { forkJoin } from 'rxjs';

@Component({
  selector: 'app-editlocation',
  templateUrl: './editlocation.component.html',
})
export class EditLocationComponent implements OnInit {
  locationFormGroup?: FormGroup;
  submitted = false;
  idlocation: number;
  clients:Client[]=[];
  taxis:Taxi[]=[];
  adresses:Adresse[]=[];

  constructor(private fb: FormBuilder, private locationService:
LocationsService, private route: ActivatedRoute, private router: Router,
private clientService:ClientsService, private
taxiService:TaxisService,private adresseService:AdressesService) {
    this.idlocation = route.snapshot.params.id;

    this.clientService.getAllClients().subscribe((clients)=>{
          this.clients=clients;
        });
        this.taxiService.getAllTaxis().subscribe((taxis)=>{
          this.taxis=taxis;
        });
        this.adresseService.getAllAdresses().subscribe((adresses)=>{
          this.adresses=adresses;
        });
   }

   ngOnInit(): void {
    this.locationService.getLocation(this.idlocation).subscribe(location
=> {
      this.locationFormGroup = this.fb.group({
        idlocation: [location.idlocation],
        dateloc: [location.dateloc, [Validators.required,
```

```
this.validateDate]],
        kmtotal: [location.kmtotal, [Validators.required,
Validators.min(1)]],
        acompte: [location.acompte, [Validators.required,
Validators.min(0)]],
        taxifk: [location.taxifk.idtaxi],
        clientfk: [location.clientfk.idclient],
        adressedepart: [location.adressedepart.idadresse],
        adressefin: [location.adressefin.idadresse],
      });
    }, error => {
      console.error('Error fetching location details:', error);
    });
  }


  updateLocation(): void {
    this.submitted = true;

    if(this.locationFormGroup && this.locationFormGroup.valid){

      const datelocValue = new
Date(this.locationFormGroup.get('dateloc')?.value);
      datelocValue.setHours(0, 0, 0, 0);
      this.locationFormGroup.patchValue({ dateloc: datelocValue });

      const taxifkId = this.locationFormGroup.get('taxifk')?.value;
      const clientfkId = this.locationFormGroup.get('clientfk')?.value;
      const adressedepartId =
this.locationFormGroup.get('adressedepart')?.value;
      const adressefinId =
this.locationFormGroup.get('adressefin')?.value;

      forkJoin([
        this.taxiService.getTaxi(taxifkId),
        this.clientService.getClient(clientfkId),
        this.adresseService.getAdresse(adressedepartId),
        this.adresseService.getAdresse(adressefinId)
      ]).subscribe(([taxi, client, adresseDepart, adresseFin]) => {

        if (this.locationFormGroup) {
          this.locationFormGroup.patchValue({
            taxifk: taxi,
            clientfk: client,
            adressedepart: adresseDepart,
            adressefin: adresseFin
          });
```

```
this.locationService.updateLocation(this.locationFormGroup?.value).subsc
ribe(
            data => {
              alert('Mise Ã  jour rÃ©ussie');
              this.router.navigate(['/locations']);
            },
            err => {
              alert(err.headers.get('error'));
            }
          );
        }
      }, (error: any) => {
        console.error('Error fetching entities:', error);
      });
    }
  }



  validateDate(control: AbstractControl): { [key: string]: any } | null
{
    const selectedDate = new Date(control.value);
    const currentDate = new Date();

    if (selectedDate < currentDate) {
      return { 'invalidDate': true };
    }

    return null;
  }
}
```

edittaxi.component.html

```
<div class="container">
  <form [formGroup]="taxiFormGroup" *ngIf="taxiFormGroup">
    <div class="form-group">
      <label>ID</label>
      <input type="text" class="form-control" formControlName="idtaxi"
readonly>
    </div>

    <div class="form-group">
      <label>Immatriculation</label>
      <input type="text" class="form-control"
formControlName="immatriculation"
             [ngClass]="{'is-invalid': submitted &&
taxiFormGroup.controls.immatriculation.errors}">
      <div *ngIf="submitted &&
taxiFormGroup.controls.immatriculation.errors" class="invalid-feedback">
        <div
*ngIf="taxiFormGroup.controls.immatriculation.errors.required">
          Immatriculation obligatoire
        </div>
      </div>
    </div>

    <div class="form-group">
      <label>Carburant</label>
      <input type="text" class="form-control"
formControlName="carburant"
             [ngClass]="{'is-invalid': submitted &&
taxiFormGroup.controls.carburant.errors}">
      <div *ngIf="submitted && taxiFormGroup.controls.carburant.errors"
class="invalid-feedback">
        <div *ngIf="taxiFormGroup.controls.carburant.errors.required">
          Carburant obligatoire
        </div>
      </div>
    </div>

  <div class="form-group">
      <label>Prix au km</label>
      <input type="text" class="form-control" formControlName="prixkm"
                  [ngClass]="{'is-invalid': submitted &&
taxiFormGroup.controls.prixkm.errors}">
      <div *ngIf="submitted && taxiFormGroup.controls.prixkm.errors"
class="invalid-feedback">
          <div *ngIf="taxiFormGroup.controls.prixkm.errors.required">
              Prix au km obligatoire
          </div>
      </div>
```

```
  </div>

    <nav class="navbar navbar-expand-sm bg-light navbar-light">
      <ul class="navbar navbar-nav">
        <li class="nav-item">
          <button (click)="onUpdateTaxi()" class="btn btn-success">
            Save
          </button>
        </li>
      </ul>
    </nav>
  </form>
</div>
```

edittaxi.component.spec.ts

```
import { ComponentFixture, TestBed } from '@angular/core/testing';

import { EditTaxiComponent } from './edittaxi.component';

describe('EdittaxiComponent', () => {
  let component: EditTaxiComponent;
  let fixture: ComponentFixture<EditTaxiComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      imports: [EditTaxiComponent]
    })
    .compileComponents();

    fixture = TestBed.createComponent(EditTaxiComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

edittaxi.component.ts

```
import { Component } from '@angular/core';
import { TaxisService } from '../../services/taxis.service';
import { Taxi } from '../../entities/taxi.entities';
import { ActivatedRoute } from '@angular/router';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { Router } from '@angular/router';


@Component({
  selector: 'app-edittaxi',
  templateUrl: './edittaxi.component.html',
})
export class EditTaxiComponent {
  taxiFormGroup?: FormGroup;
  submitted = false;
  idtaxi: number;

  constructor(private fb: FormBuilder, private taxiService:
TaxisService, private route: ActivatedRoute, private router: Router) {
    this.idtaxi = route.snapshot.params.id;
  }

  ngOnInit(): void {

    this.taxiService.getTaxi(this.idtaxi).subscribe(taxi => {
      this.taxiFormGroup = this.fb.group({
        idtaxi: [taxi.idtaxi],
        immatriculation: [taxi.immatriculation, [Validators.required,
Validators.pattern('^[0-9]{1}-[A-Z]{3}-[0-9]{3}$')]],
        carburant: [taxi.carburant, [Validators.required,
Validators.pattern('[a-zA-Z ]*')]],
        prixkm: [taxi.prixkm, [Validators.required,
Validators.pattern('^[0-9]*\.[0-9]{0,2}$')]],

      });
    }, error => {
      console.error('Error fetching taxi details:', error);
    });
  }



  onUpdateTaxi(): void {
    this.submitted = true;
    if (this.taxiFormGroup?.invalid) {
      console.log('Form is invalid');
      return;
    }
```

```
    if (this.taxiFormGroup) {
      this.taxiService.updateTaxi(this.taxiFormGroup.value).subscribe(
      data => {
        alert('Mise Ã  jour rÃ©ussie');
        this.router.navigate(['/taxis']);
      },
      err => {
        alert(err.headers.get('error'));
      }
    );
  }
}

}
```

home.component.html

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>My Angular Project</title>
</head>
<body>
  <div class="container mt-5">
    <div class="text-center">
      <h1 class="display-4">Welcome to My Angular Project</h1>
      <p class="lead">for the API3 Course</p>
      <p>I wish you a great navigation on my site!</p>
    </div>
  </div>
</body>
</html>
```

home.component.spec.ts

```
import { ComponentFixture, TestBed } from '@angular/core/testing';

import { HomeComponent } from './home.component';

describe('HomeComponent', () => {
  let component: HomeComponent;
  let fixture: ComponentFixture<HomeComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      imports: [HomeComponent]
    })
    .compileComponents();

    fixture = TestBed.createComponent(HomeComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

home.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-home',
  templateUrl: './home.component.html',
})
export class HomeComponent {

}
```

locations.component.html

```
<div class="container">
  <div class="card mt-4">
    <div class="card-body">
      <h3 class="card-title">Location</h3>

      <form #f="ngForm" (ngSubmit)="onSearchDate(f.value)" class="mb-3">
        <div class="input-group">
          <input type="date" class="form-control" ngModel name="dateloc"
placeholder="Date des locations" required>
          <button type="submit" class="btn btn-
primary">Rechercher</button>
        </div>
      </form>

      <form #g="ngForm" (ngSubmit)="onSearchDatesAndTaxi(g.value)"
class="mb-3">
        <div class="input-group">
          <select class="form-select" ngModel name="idTaxi" required>
            <option *ngFor="let taxi of taxis" [value]="taxi.idtaxi">{{
taxi.immatriculation }}</option>
          </select>
          <input type="date" class="form-control" ngModel
name="startDate" placeholder="Date de dÃ©but" required>
          <input type="date" class="form-control" ngModel name="endDate"
placeholder="Date de fin" required>
          <button type="submit" class="btn btn-primary">Rechercher par
taxi et pÃ©riode</button>
        </div>
      </form>


      <div class="d-flex justify-content-between align-items-center">
        <button class="btn btn-primary" (click)="showAll()">Afficher
toutes les locations</button>
        <button class="btn btn-success"
(click)="onNewLocation()">Ajouter une nouvelle location</button>
      </div>
    </div>
  </div>

  <ng-container *ngIf="locations">
    <ng-container *ngIf="locations.length > 0; else inconnu">
      <div class="card mt-4">
        <div class="card-body">
          <h3 class="card-title">Locations trouvÃ©es :</h3>

          <table class="table table-striped">
```

```
            <thead>
              <tr>
                <th scope="col">ID</th>
                <th scope="col">Date de location</th>
                <th scope="col">KilomÃ©trage total</th>
                <th scope="col">Acompte</th>
                <th scope="col">Total</th>
                <th scope="col">Client</th>
                <th scope="col">Taxi</th>
                <th scope="col">DÃ©part</th>
                <th scope="col">ArrivÃ©e</th>
                <th scope="col">Actions</th>
              </tr>
            </thead>
            <tbody>
              <tr *ngFor="let location of locations">
                <td>{{ location.idlocation }}</td>
                <td>{{ location.dateloc }}</td>
                <td>{{ location.kmtotal }}</td>
                <td>{{ location.acompte }}</td>
                <td>{{ location.total }}</td>
                <td>{{ location.clientfk.mail }} , {{
location.clientfk.nom }}, {{ location.clientfk.prenom }}, {{
location.clientfk.tel }}</td>
                <td>{{ location.taxifk.immatriculation }}, Carburant : {
{ location.taxifk.carburant }}, Prix km : {{ location.taxifk.prixkm }}
</td>
                <td>{{ location.adressedepart.rue }} {{
location.adressedepart.num }}  {{ location.adressedepart.cp }} {{
location.adressedepart.localite }}</td>
                <td>{{ location.adressefin.rue }} {{
location.adressefin.num }}, {{ location.adressefin.cp }} {{
location.adressefin.localite }}</td>
                <td>
                  <button (click)="onEdit(location)" class="btn btn-
primary btn-sm me-2">
                    <span class="fa fa-edit"></span> Modifier
                  </button>
                  <button (click)="onDelete(location)" class="btn btn-
danger btn-sm">
                    <span class="fa fa-trash-o"></span> Supprimer
                  </button>
                </td>
              </tr>
            </tbody>
          </table>
        </div>
      </div>
    </ng-container>
```

```
    <ng-template #inconnu>
      <div class="card mt-4">
        <div class="card-body">
          <h3 class="card-title">Aucune location trouvÃ©e</h3>
        </div>
      </div>
    </ng-template>
  </ng-container>
</div>
```

locations.component.spec.ts

```
import { ComponentFixture, TestBed } from '@angular/core/testing';

import { LocationsComponent } from './locations.component';

describe('LocationsComponent', () => {
  let component: LocationsComponent;
  let fixture: ComponentFixture<LocationsComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      imports: [LocationsComponent]
    })
    .compileComponents();

    fixture = TestBed.createComponent(LocationsComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

locations.component.ts

```
import { Component, OnInit } from '@angular/core';
import { ActivatedRoute, Router } from '@angular/router';
import { LocationEntity } from '../../entities/location.entities';
import { LocationsService } from '../../services/locations.service';
import { TaxisService } from '../../services/taxis.service';
import { ClientsService } from '../../services/clients.service';
import { Taxi } from '../../entities/taxi.entities';

@Component({
  selector: 'app-locations',
  templateUrl: './locations.component.html',
})
export class LocationsComponent implements OnInit {
  locations?: LocationEntity[];
  idtaxi: number;
  idclient: number;
  loading = false;
  taxis?: Taxi[];



  constructor(private locationsService: LocationsService, private
router: Router, private activeroute: ActivatedRoute, private
taxisService: TaxisService, private clientsService: ClientsService) {
    this.idtaxi = this.activeroute.snapshot.params.taxiId;
      if (this.idtaxi) {
        this.loading = true;
        this.taxisService.getLocationsForTaxi(this.idtaxi).subscribe({
          next: data => {
            this.locations = data;
          },
          complete: () => {
            this.loading = false;
          }
        });
      }

      this.idclient = this.activeroute.snapshot.params.clientId;
      if (this.idclient) {
        this.loading = true;

this.clientsService.getLocationsForClient(this.idclient).subscribe({
          next: data => {
            this.locations = data;
          },
          complete: () => {
            this.loading = false;
          }
```

```
      });
    }
  }

  ngOnInit(): void {
      this.loading = true;
      this.taxisService.getAllTaxis().subscribe({
      next: data => {
        this.taxis = data;
      },
      complete: () => {
        this.loading = false;
      }
   });
  }

  onSearch(value: { id: number }) {
    this.locationsService.getLocation(value.id).subscribe({
      next: data => {
        this.locations = data ? [data] : [];
      }
    });
  }

  onSearchDatesAndTaxi(formValue: any) {
    const idTaxi = formValue.idTaxi;
    const startDate = formValue.startDate;
    const endDate = formValue.endDate;

    if (idTaxi && startDate && endDate) {
      this.locationsService.getLocationsForTaxiInPeriod(idTaxi,
startDate, endDate).subscribe({
        next: data => {
          this.locations = data;
        },
        error: err => {
          console.error('Error fetching locations', err);
        }
      });
    } else {
      console.log('Invalid input values');
    }
  }


  onSearchDate(value: any) {
    if(value.dateloc) {
      this.locationsService.getLocationByDate(value.dateloc).subscribe({
        next: data => {
```

```
        this.locations = data;
      }
    });
  }
}

showAll() {
  this.locationsService.getAllLocations().subscribe({
    next: data => {
      this.locations = data;
    }
  });
}

onEdit(location: LocationEntity) {
  this.router.navigate(['/editlocation', location.idlocation]);

}

onDelete(location: LocationEntity) {
  const confirmation = confirm('Are you sure you want to delete?');

  if (confirmation) {
    this.locationsService.deleteLocation(location).subscribe({
      next: () => {
        const index = this.locations?.indexOf(location);
        if (index !== undefined && index !== -1) {
          this.locations?.splice(index, 1);
        }
        (window as any).sendAlert('success', 'Location deleted!');
                },
       });
      }
    }

  onNewLocation() {
      this.router.navigate(['newlocation']);
      }
   }
```

main-menu.component.html

```
<nav class="navbar navbar-custom navbar-expand-sm">
  <ul class="navbar-nav">
    <li class="nav-item">
      <a class="nav-link" routerLink="/">HOME</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" routerLink="/taxis">TAXIS</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" routerLink="/locations">LOCATIONS</a>
    </li>
    <li class="nav-item">
    <a class="nav-link" routerLink="/clients">CLIENTS</a>
    </li>
    <li class="nav-item">
    <a class="nav-link" routerLink="/adresses">ADRESSES</a>
    </li>
  </ul>
</nav>
```

main-menu.component.spec.ts

```
import { ComponentFixture, TestBed } from '@angular/core/testing';

import { MainMenuComponent } from './main-menu.component';

describe('MainMenuComponent', () => {
  let component: MainMenuComponent;
  let fixture: ComponentFixture<MainMenuComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      imports: [MainMenuComponent]
    })
    .compileComponents();

    fixture = TestBed.createComponent(MainMenuComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

main-menu.component.ts

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-main-menu',
  templateUrl: './main-menu.component.html',
  styleUrl: './main-menu.component.css'
})
export class MainMenuComponent {

}
```

newadresse.component.html

```
<div class="container">

  <form [formGroup]="adresseFormGroup" *ngIf="adresseFormGroup">

    <div class="form-group">
      <label>Code Postal</label>
      <input type="number" class="form-control" formControlName="cp"
             [ngClass]="{'is-invalid': submitted &&
adresseFormGroup.controls.cp.errors}">
      <div *ngIf="submitted && adresseFormGroup.controls.cp.errors"
class="invalid-feedback">
        <div *ngIf="adresseFormGroup.controls.cp.errors.required">
          Code Postal obligatoire
        </div>
      </div>
    </div>

    <div class="form-group">
      <label>LocalitÃ©</label>
      <input type="text" class="form-control" formControlName="localite"
             [ngClass]="{'is-invalid': submitted &&
adresseFormGroup.controls.localite.errors}">
      <div *ngIf="submitted &&
adresseFormGroup.controls.localite.errors" class="invalid-feedback">
        <div *ngIf="adresseFormGroup.controls.localite.errors.required">
          LocalitÃ© obligatoire
        </div>
      </div>
    </div>

    <div class="form-group">
      <label>Rue</label>
      <input type="text" class="form-control" formControlName="rue"
             [ngClass]="{'is-invalid': submitted &&
adresseFormGroup.controls.rue.errors}">
      <div *ngIf="submitted && adresseFormGroup.controls.rue.errors"
class="invalid-feedback">
        <div *ngIf="adresseFormGroup.controls.rue.errors.required">
          Rue obligatoire
        </div>
      </div>
    </div>

    <div class="form-group">
      <label>NumÃ©ro</label>
      <input type="text" class="form-control" formControlName="num"
             [ngClass]="{'is-invalid': submitted &&
adresseFormGroup.controls.num.errors}">
```

```
      <div *ngIf="submitted && adresseFormGroup.controls.num.errors"
class="invalid-feedback">
        <div *ngIf="adresseFormGroup.controls.num.errors.required">
          NumÃ©ro obligatoire
        </div>
      </div>
    </div>

    <button (click)="onSaveAdresse()" class="btn btn-success">
      Save
    </button>
  </form>

</div>
```

newadresse.component.spec.ts

```
import { ComponentFixture, TestBed } from '@angular/core/testing';

import { NewAdresseComponent } from './newadresse.component';

describe('NewAdresseComponent', () => {
  let component: NewAdresseComponent;
  let fixture: ComponentFixture<NewAdresseComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      imports: [NewAdresseComponent]
    })
    .compileComponents();

    fixture = TestBed.createComponent(NewAdresseComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

newadresse.component.ts

```
import { Component, EventEmitter, OnInit } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { AdressesService } from '../../services/adresses.service';
import { Router } from '@angular/router';
import { Adresse } from '../../entities/adresse.entities';
import { Output } from '@angular/core';

@Component({
  selector: 'app-newadresse',
  templateUrl: './newadresse.component.html',
})

export class NewAdresseComponent implements OnInit {

  @Output() addedAdresse = new EventEmitter<Adresse>();
  adresseFormGroup?: FormGroup;
  submitted = false;

  constructor(private fb: FormBuilder, private adressesService:
AdressesService, private router: Router) {}

  ngOnInit(): void {
      this.adresseFormGroup = this.fb.group({
      cp: ['', [Validators.required, Validators.pattern('^[0-9]+$')]],
      localite: ['', [Validators.required, Validators.pattern('^[a-zA-
ZÃ€-Å¾Ã€-Ã¿ ]*$')]],
      rue: ['', [Validators.required, Validators.pattern('^[a-zA-ZÃ€-
Å¾Ã€-Ã¿ ]*$')]],
      num: ['', [Validators.required]],
    });
  }

  onSaveAdresse(): void {
    this.submitted = true;
    if (this.adresseFormGroup?.invalid) { return; }

this.adressesService.saveAdresse(this.adresseFormGroup?.value).subscribe
(
      data => {
        this.addedAdresse.emit(data);
        alert('Sauvegarde rÃ©ussie');
        this.router.navigate(['/adresses']);
      }
    );

    this.adresseFormGroup?.reset();
  }
}
```

newclient.component.html

```
<div class="container">

  <form [formGroup]="clientFormGroup" *ngIf="clientFormGroup">

    <div class="form-group">
      <label>Email</label>
      <input type="email" class="form-control" formControlName="mail"
             [ngClass]="{'is-invalid': submitted &&
clientFormGroup.controls.mail.errors}">
      <div *ngIf="submitted && clientFormGroup.controls.mail.errors"
class="invalid-feedback">
        <div *ngIf="clientFormGroup.controls.mail.errors.required">
          Email obligatoire
        </div>
        <div *ngIf="clientFormGroup.controls.mail.errors.email">
          Entrez un email valide
        </div>
      </div>
    </div>

    <div class="form-group">
      <label>Nom</label>
      <input type="text" class="form-control" formControlName="nom"
             [ngClass]="{'is-invalid': submitted &&
clientFormGroup.controls.nom.errors}">
      <div *ngIf="submitted && clientFormGroup.controls.nom.errors"
class="invalid-feedback">
        <div *ngIf="clientFormGroup.controls.nom.errors.required">
          Nom obligatoire
        </div>
      </div>
    </div>

    <div class="form-group">
      <label>PrÃ©nom</label>
      <input type="text" class="form-control" formControlName="prenom"
             [ngClass]="{'is-invalid': submitted &&
clientFormGroup.controls.prenom.errors}">
      <div *ngIf="submitted && clientFormGroup.controls.prenom.errors"
class="invalid-feedback">
        <div *ngIf="clientFormGroup.controls.prenom.errors.required">
          PrÃ©nom obligatoire
        </div>
      </div>
    </div>

    <div class="form-group">
      <label>TÃ©lÃ©phone</label>
```

```
      <input type="tel" class="form-control" formControlName="tel"
             [ngClass]="{'is-invalid': submitted &&
clientFormGroup.controls.tel.errors}">
      <div *ngIf="submitted && clientFormGroup.controls.tel.errors"
class="invalid-feedback">
        <div *ngIf="clientFormGroup.controls.tel.errors.required">
          TÃ©lÃ©phone obligatoire
        </div>
      </div>
    </div>


    <button (click)="onSaveClient()" class="btn btn-success">
      Save
    </button>
  </form>

</div>
```

newclient.component.spec.ts

```
import { ComponentFixture, TestBed } from '@angular/core/testing';

import { NewClientComponent } from './newclient.component';

describe('NewClientComponent', () => {
  let component: NewClientComponent;
  let fixture: ComponentFixture<NewClientComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      imports: [NewClientComponent]
    })
    .compileComponents();

    fixture = TestBed.createComponent(NewClientComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

newclient.component.ts

```
import { Component, EventEmitter, OnInit, Output } from '@angular/core';
import { FormBuilder, FormGroup, Validators } from '@angular/forms';
import { ClientsService } from '../../services/clients.service';
import { Router } from '@angular/router';
import { Client } from '../../entities/client.entities';

@Component({
  selector: 'app-newclient',
  templateUrl: './newclient.component.html',
})

export class NewClientComponent implements OnInit {

  @Output() addedClient = new EventEmitter<Client>();
  clientFormGroup?: FormGroup;
  submitted = false;

  constructor(private fb: FormBuilder, private clientsService:
ClientsService, private router: Router) {}

  ngOnInit(): void {
    this.clientFormGroup = this.fb.group({
      mail: ['', [Validators.required, Validators.email]],
      nom: ['', [Validators.required, Validators.pattern('[a-zA-Z
]*')]],
      prenom: ['', [Validators.required, Validators.pattern('[a-zA-Z
]*')]],
      tel: ['', [Validators.required, Validators.pattern('[0-9]*'),
Validators.minLength(10)]],
    });
  }

  onSaveClient(): void {
    this.submitted = true;
    if (this.clientFormGroup?.invalid) { return; }

this.clientsService.saveClient(this.clientFormGroup?.value).subscribe(
      data => {
        this.addedClient.emit(data);
        alert('Sauvegarde rÃ©ussie');
        this.router.navigateByUrl('/clients');
      }
    );

    this.clientFormGroup?.reset();
  }
}
```

newlocation.component.html

```
<div class="container">

  <form [formGroup]="locationFormGroup" *ngIf="locationFormGroup">

    <div class="form-group">
      <label>Date de la location</label>
      <input type="date" class="form-control" formControlName="dateloc"
             [ngClass]="{'is-invalid': submitted &&
locationFormGroup.controls.dateloc.errors}">
      <div *ngIf="submitted &&
locationFormGroup.controls.dateloc.errors" class="invalid-feedback">
        <div *ngIf="locationFormGroup.controls.dateloc.errors.required">
          Date obligatoire
        </div>
      </div>
    </div>

    <div class="form-group">
      <label>Nombre de kilomÃ¨tres total</label>
      <input type="number" class="form-control"
formControlName="kmtotal"
             [ngClass]="{'is-invalid': submitted &&
locationFormGroup.controls.kmtotal.errors}">
      <div *ngIf="submitted &&
locationFormGroup.controls.kmtotal.errors" class="invalid-feedback">
        <div *ngIf="locationFormGroup.controls.kmtotal.errors.required">
          Nombre de km total obligatoire
        </div>
      </div>
    </div>

    <div class="form-group">
      <label>Acompte</label>
      <input type="number" class="form-control"
formControlName="acompte"
             [ngClass]="{'is-invalid': submitted &&
locationFormGroup.controls.acompte.errors}">
      <div *ngIf="submitted &&
locationFormGroup.controls.acompte.errors" class="invalid-feedback">
        <div *ngIf="locationFormGroup.controls.acompte.errors.required">
          Acompte obligatoire
        </div>
      </div>
    </div>

      <div class="row">
        <div class="col-md-6">
          <label for="clientfk">Client</label>
```

```
          <select id="clientfk" formControlName="clientfk" class="form-
control">
            <option *ngFor="let client of clients"
[value]="client.idclient">{{ client.mail }}</option>
          </select>
        </div>

        <div class="col-md-6">
          <label for="taxifk">Taxi</label>
          <select id="taxifk" formControlName="taxifk" class="form-
control">
            <option *ngFor="let taxi of taxis" [value]="taxi.idtaxi">{{
taxi.immatriculation }}</option>
          </select>
        </div>

        <div class="col-md-6">
          <label for="adressedepart">Adresse de DÃ©part</label>
          <select id="adressedepart" formControlName="adressedepart"
class="form-control">
            <option *ngFor="let adresse of adresses"
[value]="adresse.idadresse">{{ adresse.rue }} , {{adresse.num}} , {{
adresse.cp}} {{adresse.localite}}</option>
          </select>
        </div>

        <div class="col-md-6">
          <label for="adressefin">Adresse de Fin</label>
          <select id="adressefin" formControlName="adressefin"
class="form-control">
            <option *ngFor="let adresse of adresses"
[value]="adresse.idadresse">{{ adresse.rue }} , {{adresse.num}} , {{
adresse.cp}} {{adresse.localite}}</option>
          </select>
        </div>
      </div>
      <div *ngIf="locationFormGroup?.hasError('addressesSame') &&
submitted">
        Les adresses de dÃ©part et d'arrivÃ©e ne peuvent pas Ãªtre
identiquese.
      </div>
      <button (click)="onSaveLocation()" class="btn btn-success">
        Save
      </button>
    </form>
    </div>
```

newlocation.component.spec.ts

```
import { ComponentFixture, TestBed } from '@angular/core/testing';

import { NewLocationComponent } from './newlocation.component';

describe('NewLocationComponent', () => {
  let component: NewLocationComponent;
  let fixture: ComponentFixture<NewLocationComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      imports: [NewLocationComponent]
    })
    .compileComponents();

    fixture = TestBed.createComponent(NewLocationComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

newlocation.component.ts

```
import { Component, EventEmitter, OnInit, Output } from '@angular/core';
import { FormBuilder, FormGroup, Validators, AbstractControl,
ValidationErrors } from '@angular/forms';
import { LocationsService } from '../../services/locations.service';
import { ActivatedRoute, Router } from '@angular/router';
import { Client } from '../../entities/client.entities';
import { Taxi } from '../../entities/taxi.entities';
import { Adresse } from '../../entities/adresse.entities';
import { ClientsService } from '../../services/clients.service';
import { TaxisService } from '../../services/taxis.service';
import { AdressesService } from '../../services/adresses.service';
import { forkJoin } from 'rxjs';

@Component({
  selector: 'app-newlocation',
  templateUrl: './newlocation.component.html',
})
export class NewLocationComponent implements OnInit {

  @Output() addedLocation = new EventEmitter<any>();
  locationFormGroup?: FormGroup;
  submitted = false;
  idlocation: number;
  clients: Client[] = [];
  taxis: Taxi[] = [];
  adresses: Adresse[] = [];

  constructor(
    private fb: FormBuilder,
    private locationService: LocationsService,
    private route: ActivatedRoute,
    private router: Router,
    private clientService: ClientsService,
    private taxiService: TaxisService,
    private adresseService: AdressesService
  ) {
    this.idlocation = route.snapshot.params.id;

    this.clientService.getAllClients().subscribe((clients) => {
      this.clients = clients;
    });
    this.taxiService.getAllTaxis().subscribe((taxis) => {
      this.taxis = taxis;
    });
    this.adresseService.getAllAdresses().subscribe((adresses) => {
      this.adresses = adresses;
    });
  }
```

```
  ngOnInit(): void {
    this.locationFormGroup = this.fb.group({
      dateloc: ['', [Validators.required, this.validateDate]],
      kmtotal: ['', [Validators.required, Validators.min(1)]],
      acompte: ['', [Validators.required, Validators.min(0)]],
      taxifk: ['', Validators.required],
      clientfk: ['', Validators.required],
      adressedepart: ['', Validators.required],
      adressefin: ['', Validators.required],
    }, { validator: this.addressesNotSame });

  }

  onSaveLocation(): void {
    this.submitted = true;
    if (this.locationFormGroup?.invalid) {
      return;
    }
    if (this.locationFormGroup) {
      const datelocValue = new
Date(this.locationFormGroup.get('dateloc')?.value);
      datelocValue.setHours(0, 0, 0, 0);
      this.locationFormGroup.patchValue({ dateloc: datelocValue });

      const taxifkId = this.locationFormGroup.get('taxifk')?.value;
      const clientfkId = this.locationFormGroup.get('clientfk')?.value;
      const adressedepartId =
this.locationFormGroup.get('adressedepart')?.value;
      const adressefinId =
this.locationFormGroup.get('adressefin')?.value;

      forkJoin([
        this.taxiService.getTaxi(taxifkId),
        this.clientService.getClient(clientfkId),
        this.adresseService.getAdresse(adressedepartId),
        this.adresseService.getAdresse(adressefinId)
      ]).subscribe(([taxi, client, adresseDepart, adresseFin]) => {
        if (this.locationFormGroup) {
          this.locationFormGroup.patchValue({
            taxifk: taxi,
            clientfk: client,
            adressedepart: adresseDepart,
            adressefin: adresseFin
          });


this.locationService.saveLocation(this.locationFormGroup?.value).subscri
be(
```

```
            data => {
              this.addedLocation.emit(data);
              alert('Sauvegarde rÃ©ussie');
              this.router.navigate(['/locations']);
            },
            err => {
              console.error('Error saving location:', err);
            }
          );

          this.locationFormGroup?.reset();
        }
      }, error => {
        console.error('Error fetching entities:', error);
      });
    }
  }

  validateDate(control: AbstractControl): ValidationErrors | null {
    const selectedDate = new Date(control.value);
    const currentDate = new Date();

    if (selectedDate < currentDate) {
      return { 'invalidDate': true };
    }

    return null;
  }

  addressesNotSame(group: FormGroup): ValidationErrors | null {
    const adressedepart = group.get('adressedepart')?.value;
    const adressefin = group.get('adressefin')?.value;

    if (adressedepart && adressefin && adressedepart === adressefin) {
      return { 'addressesSame': true };
    }

    return null;
  }
}
```

newtaxi.component.html

```
<div class="container">

  <form [formGroup]="taxiFormGroup" *ngIf="taxiFormGroup">

    <div class="form-group">
      <label>Immatriculation</label>
      <input type="text" class="form-control"
formControlName="immatriculation"
             [ngClass]="{'is-invalid': submitted &&
taxiFormGroup.controls.immatriculation.errors}">
      <div *ngIf="submitted &&
taxiFormGroup.controls.immatriculation.errors" class="invalid-feedback">
        <div
*ngIf="taxiFormGroup.controls.immatriculation.errors.required">
          Immatriculation obligatoire
        </div>
      </div>
    </div>

    <div class="form-group">
      <label>Carburant</label>
      <input type="text" class="form-control"
formControlName="carburant"
             [ngClass]="{'is-invalid': submitted &&
taxiFormGroup.controls.carburant.errors}">
      <div *ngIf="submitted && taxiFormGroup.controls.carburant.errors"
class="invalid-feedback">
        <div *ngIf="taxiFormGroup.controls.carburant.errors.required">
          Carburant obligatoire
        </div>
      </div>
    </div>

    <div class="form-group">
      <label>Prix par kilomÃ¨tre</label>
      <input type="number" class="form-control" formControlName="prixkm"
             [ngClass]="{'is-invalid': submitted &&
taxiFormGroup.controls.prixkm.errors}">
      <div *ngIf="submitted && taxiFormGroup.controls.prixkm.errors"
class="invalid-feedback">
        <div *ngIf="taxiFormGroup.controls.prixkm.errors.required">
          Prix par kilomÃ¨tre obligatoire
        </div>
      </div>
    </div>

    <button (click)="onSaveTaxi()" class="btn btn-success">
      Save
```

```
    </button>
  </form>

</div>
```

newtaxi.component.spec.ts

```
import { ComponentFixture, TestBed } from '@angular/core/testing';

import { NewTaxiComponent } from './newtaxi.component';

describe('NewTaxiComponent', () => {
  let component: NewTaxiComponent;
  let fixture: ComponentFixture<NewTaxiComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      imports: [NewTaxiComponent]
    })
    .compileComponents();

    fixture = TestBed.createComponent(NewTaxiComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

newtaxi.component.ts

```
import { Component, EventEmitter, OnInit, Output } from '@angular/core';
import {FormBuilder, FormGroup, Validators} from '@angular/forms';
import { TaxisService } from '../../services/taxis.service';
import { Router } from '@angular/router';
import { Taxi } from '../../entities/taxi.entities';

@Component({
  selector: 'app-newtaxi',
  templateUrl: './newtaxi.component.html',
})

export class NewTaxiComponent {
  @Output() addedTaxi = new EventEmitter<Taxi>();
  taxiFormGroup?: FormGroup;
  submitted = false;
  id: number | null = null;

  constructor(private fb: FormBuilder, private taxiService:
TaxisService, private router: Router) {
  }

  ngOnInit(): void {
    this.taxiFormGroup = this.fb.group({
      immatriculation: ['', Validators.required],
      carburant: ['', Validators.required],
      prixkm: ['', [Validators.required, Validators.min(0)]],
    });
  }

  onSaveTaxi(): void {
    this.submitted = true;
    if (this.taxiFormGroup?.invalid) { return; }
    this.taxiService.saveTaxi(this.taxiFormGroup?.value).subscribe(
      data => {
        this.addedTaxi.emit(data);
        alert('Sauvegarde rÃ©ussie');
        this.router.navigateByUrl('/taxis');
      }
    );

    this.taxiFormGroup?.reset();
  }
}
```

taxis.component.html

```
<div class="container">
  <div class="card mt-4">
    <div class="card-body">
      <h3 class="card-title">Taxi</h3>

      <form #f="ngForm" (ngSubmit)="getTaxisByImmatriculation(f.value)"
class="mb-3">
        <div class="input-group">
          <input type="text" class="form-control" ngModel
name="immatriculation" placeholder="Immatriculation du taxi">
          <button type="submit" class="btn btn-
primary">Rechercher</button>
        </div>
      </form>

      <div class="d-flex justify-content-between align-items-center">
        <button class="btn btn-primary" (click)="showAll()">Afficher
tous les taxis</button>
        <button class="btn btn-success" (click)="onNewTaxi()">Ajouter un
nouveau taxi</button>
      </div>
    </div>
  </div>

  <ng-container *ngIf="taxis">
    <ng-container *ngIf="taxis.length > 0; else inconnu">
      <div class="card mt-4">
        <div class="card-body">
          <h3 class="card-title">Taxis trouvÃ©s :</h3>

          <table class="table table-striped">
            <thead>
            <tr>
              <th scope="col">ID</th>
              <th scope="col">Immatriculation</th>
              <th scope="col">Carburant</th>
              <th scope="col">Prix par kilomÃ¨tre</th>
              <th scope="col">KilomÃ¨tres total</th>
              <th scope="col">Montant total des locations</th>
              <th scope="col">Clients</th>
              <th scope="col">Locations</th>
              <th scope="col">Actions</th>
            </tr>
            </thead>
            <tbody>
            <tr *ngFor="let taxi of taxis; let i = index">
              <td>{{ taxi.idtaxi }}</td>
              <td>{{ taxi.immatriculation }}</td>
```

```
              <td>{{ taxi.carburant }}</td>
              <td>{{ taxi.prixkm }}</td>
              <td>{{ this.kmtotal ? this.kmtotal[i]:0 }}</td>
              <td>{{ this.montanttotal ? this.montanttotal[i]:0 }}</td>
              <td>
                <button (click)="getClientsForTaxi(taxi.idtaxi)"
class="btn btn-primary btn-sm">
                  <span class="fa fa-users"></span>
                  {{ selectedTaxiId === taxi.idtaxi ? 'Cacher' :
'Afficher' }} Clients
                </button>
              </td>
              <td>
                <button (click)="getLocationsForTaxi(taxi.idtaxi)"
class="btn btn-primary btn-sm">
                  <span class="fa fa-users"></span>
                  {{ selectedTaxiId === taxi.idtaxi ? 'Cacher' :
'Afficher' }} Locations
                </button>
              </td>
              <td>
                <button (click)="onEdit(taxi)" class="btn btn-primary
btn-sm me-2">
                  <span class="fa fa-edit"></span> Modifier
                </button>
                <button (click)="onDelete(taxi)" class="btn btn-danger
btn-sm">
                  <span class="fa fa-trash-o"></span> Supprimer
                </button>
              </td>
            </tr>
            </tbody>
          </table>
        </div>
      </div>
    </ng-container>

    <ng-template #inconnu>
      <div class="card mt-4">
        <div class="card-body">
          <h3 class="card-title">Aucun taxi trouvÃ©</h3>
        </div>
      </div>
    </ng-template>
  </ng-container>
</div>
```

taxis.component.spec.ts

```
import { ComponentFixture, TestBed } from '@angular/core/testing';

import { TaxisComponent } from './taxis.component';

describe('TaxisComponent', () => {
  let component: TaxisComponent;
  let fixture: ComponentFixture<TaxisComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      imports: [TaxisComponent]
    })
    .compileComponents();

    fixture = TestBed.createComponent(TaxisComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create', () => {
    expect(component).toBeTruthy();
  });
});
```

taxis.component.ts

```
import { Component, OnInit } from '@angular/core';
import { Router } from '@angular/router';
import { Taxi } from '../../entities/taxi.entities';
import { TaxisService } from '../../services/taxis.service';
import { Client } from '../../entities/client.entities';
import { catchError, map } from 'rxjs/operators';

@Component({
  selector: 'app-taxis',
  templateUrl: './taxis.component.html',
})
export class TaxisComponent implements OnInit {
  taxis?: Taxi[];
  loading = false;
  selectedTaxiId?: number;
  kmtotal?: number[];
  montanttotal?: number[];

  constructor(private taxisService: TaxisService, private router:
Router) {

  }

  ngOnInit(): void {

  }

  onSearch(value: { id: number }) {
    this.taxisService.getTaxi(value.id).subscribe({
      next: data => {
        this.taxis = data ? [data] : [];
      }
    });
  }

  getTaxisByImmatriculation(value: { immatriculation: string }) {
    this.loading = true;

this.taxisService.getTaxisByImmatriculation(value.immatriculation).subsc
ribe({
      next: data => {
        this.taxis = data ? [data] : [];
      },
      complete: () => {
        this.loading = false;
      }
    });
  }
```

```
  getClientsForTaxi(taxiId: number): void {
    if (taxiId) {
      this.loading = true;
      this.router.navigate(['/clients', { taxiId }]);
    }
  }

  getLocationsForTaxi(taxiId: number): void {
    if (taxiId) {
      this.loading = true;
      this.router.navigate(['/locations', { taxiId }]);
    }
  }

  getKmtotal(): void {

    this.loading = true;
    if(this.taxis){
    this.taxisService.getKmTotal(this.taxis.map(t=>t.idtaxi)).subscribe(
{
      next: data => {
        console.log(data);
        this.kmtotal = data;
      },
      complete: () => {
        this.loading = false;
      }
    });
    }
  }

  getMontantTotal(): void {

    this.loading = true;
    if(this.taxis){

this.taxisService.getMontantTotal(this.taxis.map(t=>t.idtaxi)).subscribe
({
      next: data => {
        console.log(data);
        this.montanttotal = data;
      },
      complete: () => {
        this.loading = false;
      }
    });
    }
```

```
  }

  showAll() {

    this.loading = true;
    this.taxisService.getAllTaxis().subscribe({
      next: data => {
        this.taxis = data;
      },
      complete: () => {
        this.loading = false;
        this.getKmtotal();
        this.getMontantTotal();
      }
    });
  }

  onEdit(taxi: Taxi) {
    this.router.navigate(['/edittaxi', taxi.idtaxi]);
  }

  onDelete(taxi: Taxi) {
    const confirmation = confirm('Are you sure you want to delete?');

    if (confirmation) {
      this.taxisService.deleteTaxi(taxi).subscribe({
        next: () => {
          const index = this.taxis?.indexOf(taxi);
          if (index !== undefined && index !== -1) {
            this.taxis?.splice(index, 1);
          }
          (window as any).sendAlert('success', 'Taxi deleted!');
        }
      });
    }
  }

  onNewTaxi() {
    this.router.navigate(['newtaxi']);
  }
}
```

adresse.entities.ts

```
export interface Adresse{
    idadresse: number;
    cp: number;
    localite: string;
    rue: string;
    num: string;
}
```

client.entities.ts

```
export interface Client{
  idclient: number;
  mail: string;
  nom: string;
  prenom: string;
  tel: string;
}
```

location.entities.ts

```
import {Adresse} from './adresse.entities';
import { Client } from './client.entities';
import { Taxi } from './taxi.entities';

export interface LocationEntity{
    idlocation: number;
    dateloc: string;
    kmtotal: number;
    acompte: number;
    total: number;
    taxifk: Taxi;
    clientfk: Client;
    adressedepart: Adresse;
    adressefin: Adresse;

}
```

taxi.entities.ts

```
export interface Taxi {

  idtaxi: number;
  immatriculation: string;
  carburant:string;
  prixkm: number;
}
```

adresses.service.ts

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { environment } from '../../environments/environment';
import { Observable } from 'rxjs';
import { Adresse } from '../entities/adresse.entities';

@Injectable({ providedIn: 'root' })
export class AdressesService {
  private host = environment.host+"/adresses";

  constructor(private http: HttpClient) {}

  getAdresse(id: number): Observable<Adresse> {
    return this.http.get<Adresse>(`${this.host}/${id}`);
  }

  getAdresseByLocalite(localite: string): Observable<Adresse[]> {
    return this.http.get<Adresse[]>(`${this.host}/localite/${localite}
`);
  }

  getAllAdresses(): Observable<Adresse[]> {
    return this.http.get<Adresse[]>(`${this.host}/all`);
  }

  deleteAdresse(a: Adresse): Observable<void> {
    return this.http.delete<void>(`${this.host}/${a.idadresse}`);
  }

  saveAdresse(a: Adresse): Observable<Adresse> {
    return this.http.post<Adresse>(`${this.host}/create`, a);
  }

  updateAdresse(a: Adresse): Observable<Adresse> {
    return this.http.put<Adresse>(`${this.host}/${a.idadresse}`, a);
  }


}
```

clients.service.ts

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { environment } from '../../environments/environment';
import { Observable } from 'rxjs';
import { Client } from '../entities/client.entities';
import { LocationEntity } from '../entities/location.entities';

@Injectable({ providedIn: 'root' })
export class ClientsService {
  private host = environment.host+"/clients";

  constructor(private http: HttpClient) {}

  getClient(id: number): Observable<Client> {
    return this.http.get<Client>(`${this.host}/${id}`);
  }


  getAllClients(): Observable<Client[]> {
    return this.http.get<Client[]>(`${this.host}/all`);
  }

  deleteClient(c: Client): Observable<void> {
    return this.http.delete<void>(`${this.host}/${c.idclient}`);
  }


  saveClient(c: Client): Observable<Client> {
    return this.http.post<Client>(`${this.host}/create`, c );
  }

  updateClient(c: Client): Observable<Client> {
    return this.http.put<Client>(`${this.host}/${c.idclient}`, c);
  }


  getClientByNomAndPrenomAndTel(nom: string, prenom: string, tel:
string): Observable<Client> {
    return this.http.get<Client>(`${this.host}/${nom}/${prenom}/${tel}
`);
  }


  getLocationsForClient(id: number): Observable<LocationEntity[]> {
    return this.http.get<LocationEntity[]>(`${this.host}
/identifiantloc/${id}`);
  }
}
```

locations.service.ts

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { environment } from '../../environments/environment';
import { Observable } from 'rxjs';
import { LocationEntity } from '../entities/location.entities';

@Injectable({ providedIn: 'root' })
export class LocationsService {
  private host = environment.host+"/locations";

  constructor(private http: HttpClient) {}

  getLocation(id: number): Observable<LocationEntity> {
    return this.http.get<LocationEntity>(`${this.host}/${id}`);
  }

  getLocationByDate(dateloc: Date): Observable<LocationEntity[]> {
    return this.http.get<LocationEntity[]>(`${this.host}/datelocation/${
dateloc}`);
  }

  getAllLocations(): Observable<LocationEntity[]> {
    return this.http.get<LocationEntity[]>(this.host + '/all');
  }

  deleteLocation(l: LocationEntity): Observable<void> {
    return this.http.delete<void>(`${this.host}/${l.idlocation}`);
  }

  saveLocation(l: LocationEntity): Observable<LocationEntity> {
    return this.http.post<LocationEntity>(this.host+'/create', l);
  }

  updateLocation(l: LocationEntity): Observable<LocationEntity> {
    return this.http.put<LocationEntity>(`${this.host}/${l.idlocation}`,
l);
  }

  getLocationsForTaxi(taxiId: number): Observable<LocationEntity[]> {
    return this.http.get<LocationEntity[]>(`${this.host}/taxi/${taxiId}
`);
  }

  getLocationsForClient(clientId: number): Observable<LocationEntity[]>
{
    return this.http.get<LocationEntity[]>(`${this.host}/client/${
clientId}`);
  }
```

```
  getLocationsForTaxiInPeriod(taxiId: number, startDate: Date, endDate:
Date): Observable<LocationEntity[]>{
    return this.http.get<LocationEntity[]>(`${this.host}/${taxiId}/${
startDate}/${endDate}`);
  }
}
```

taxis.service.ts

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { environment } from '../../environments/environment';
import { Observable } from 'rxjs';
import { Taxi } from '../entities/taxi.entities';
import { Client } from '../entities/client.entities';
import { LocationEntity } from '../entities/location.entities';

@Injectable({ providedIn: 'root' })
export class TaxisService {
  private host = environment.host+"/taxis";

  constructor(private http: HttpClient) {}

  saveTaxi(t: Taxi): Observable<Taxi> {
    return this.http.post<Taxi>(`${this.host}/create`, t );
  }

  getTaxi(id: number): Observable<Taxi> {
    return this.http.get<Taxi>(`${this.host}/${id}`);
  }

  getAllTaxis(): Observable<Taxi[]> {
    return this.http.get<Taxi[]>(this.host + '/all');
  }

  getTaxisByImmatriculation(immatriculation: string): Observable<Taxi> {
    return this.http.get<Taxi>(`${this.host}/immatriculation/${
immatriculation}`);
  }

  deleteTaxi(t: Taxi): Observable<void> {
    return this.http.delete<void>(`${this.host}/${t.idtaxi}`);
  }

  updateTaxi(t: Taxi): Observable<Taxi> {
    return this.http.put<Taxi>(`${this.host}/${t.idtaxi}`, t);
  }

  getClientsForTaxi(id: number): Observable<Client[]> {
    return this.http.get<Client[]>(`${this.host}/identifiant/${id}`);
  }

  getLocationsForTaxi(id: number): Observable<LocationEntity[]> {
    return this.http.get<LocationEntity[]>(`${this.host}
/identifiantloc/${id}`);
  }
```

```
  getKmTotal(taxiIds: number[]): Observable<number[]> {
    return this.http.post<number[]>(`${this.host}/identifiantkmtot`,
taxiIds);
  }

  getMontantTotal(taxiIds: number[]): Observable<number[]> {
    return this.http.post<number[]>(`${this.host}/identifiantmontant`,
taxiIds);
  }

}
```