Mini-Projet (Atelier Java / Ingénierie des Bases de Données)
Location des voitures

Encadré par :Mme Saloua Zammeli

Mme Hajer Dammak

Réalisé par :Amen Allah Hajji

Mohame Amine zeaibi

Introduction

Dans le cadre de notre formation en informatique, nous avons développé une application complète de gestion de location de voitures. Ce projet, réalisé avec **Java (NetBeans)** pour la partie interface et **PL/SQL** pour la base de données, a pour but de mettre en œuvre nos compétences en développement logiciel, modélisation de bases de données, programmation orientée objet et interaction avec un SGBD via **JDBC**.

Objectifs du projet

- Gérer les opérations de location de voitures.
- Fournir une interface graphique pour les **clients** et pour les **administrateurs** (gérant ou "boss").
- Assurer la sécurité des accès via un système de **connexion par mot de passe**.
- Permettre une interaction fluide entre l'interface Java et la base de données via **JDBC**.
- Implémenter une logique métier fiable via des **procédures PL/SQL** et des **triggers**.

Modèle relationnel de la base de données

Table Voiture

Nom de champ	Type	Description
matricule	VARCHAR(20)	Clé primaire, identifiant unique
marque	VARCHAR(20)	Marque de la voiture
model	VARCHAR(20)	Modèle
annee_fabrication	NUMBER	Année de fabrication
couleur	VARCHAR(20)	Couleur de la voiture
kilometrage	NUMBER	Kilométrage actuel

carburant	VARCHAR(10)	Type de carburant
puissance	NUMBER	Puissance en chevaux
prix_par_jour	NUMBER	Tarif de location par jour
etat	NUMBER	État de la voiture (0: dispo, 1: louée)

Table Client

Nom de champ	Type	Description
cin	NUMBER	Clé primaire
nom	VARCHAR(10)	Nom du client
prenom	VARCHAR(10)	Prénom du client
date_naiss	DATE	Date de naissance
numero_tel	NUMBER	Numéro de téléphone
num_conduite	NUMBER	Numéro de permis de conduire
gender	VARCHAR(10)	Sexe
adresse	VARCHAR(20)	Adresse

Table location

Nom de champ	Type	Description
matricule	VARCHAR(20)	Clé étrangère vers voiture(matricule)
cin	NUMBER	Clé étrangère vers client(cin)
date_debut	DATE	Date de début de location
date_fin	DATE	Date de fin prévue
lieu_retrait	VARCHAR(20)	Lieu de retrait du véhicule
lieu_restitution	VARCHAR(20)	Lieu de retour prévu du véhicule

Diagramme de Classe

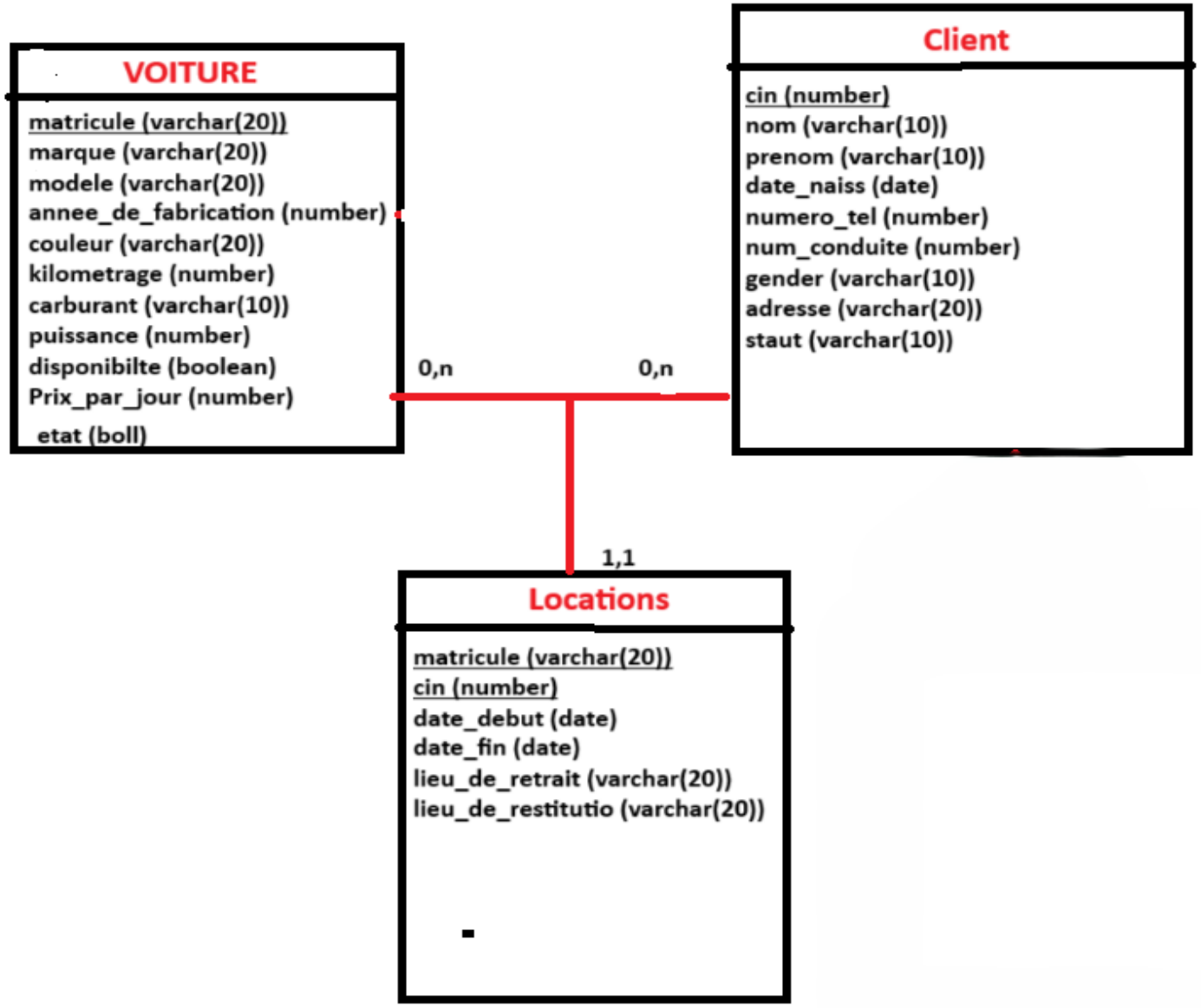
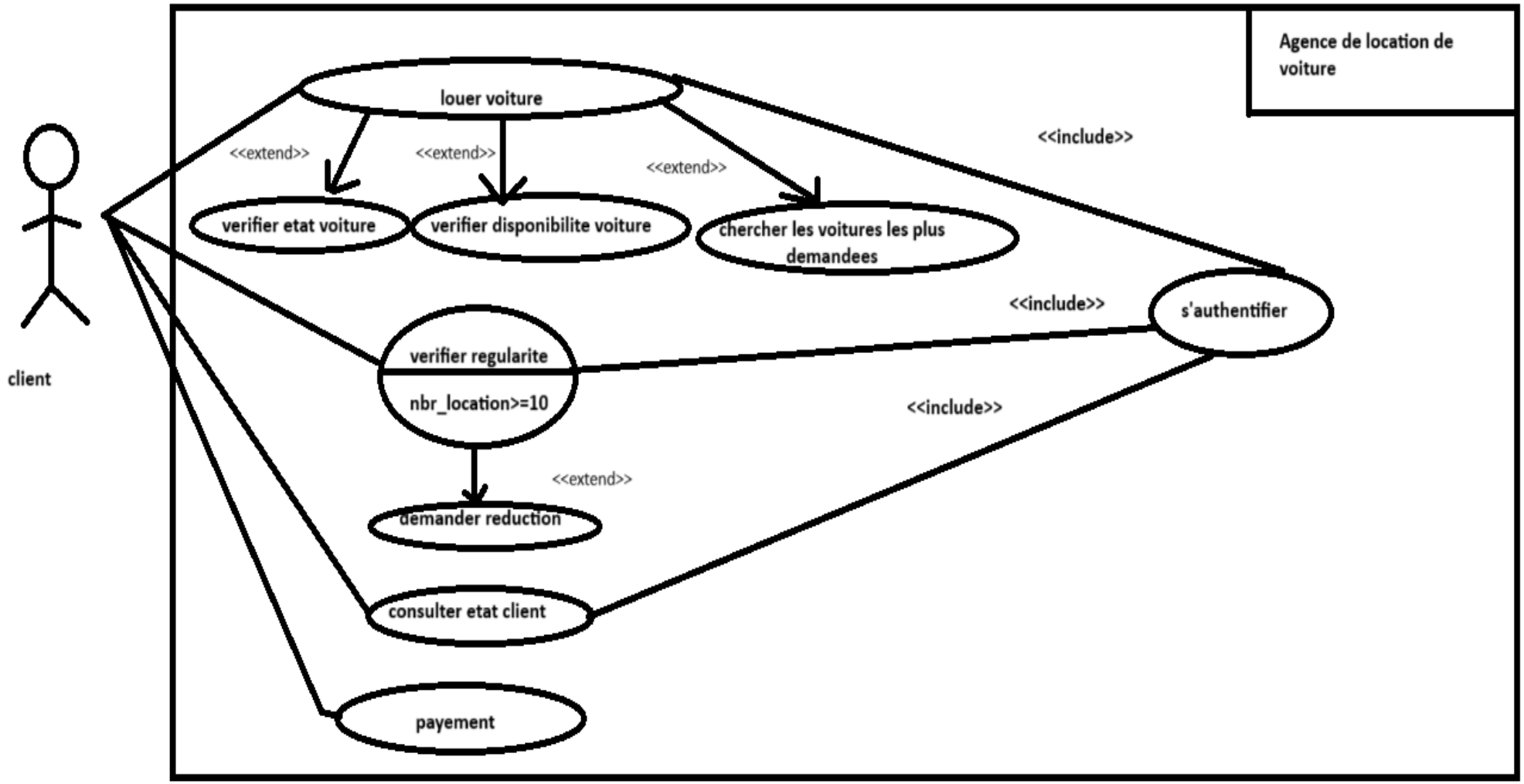
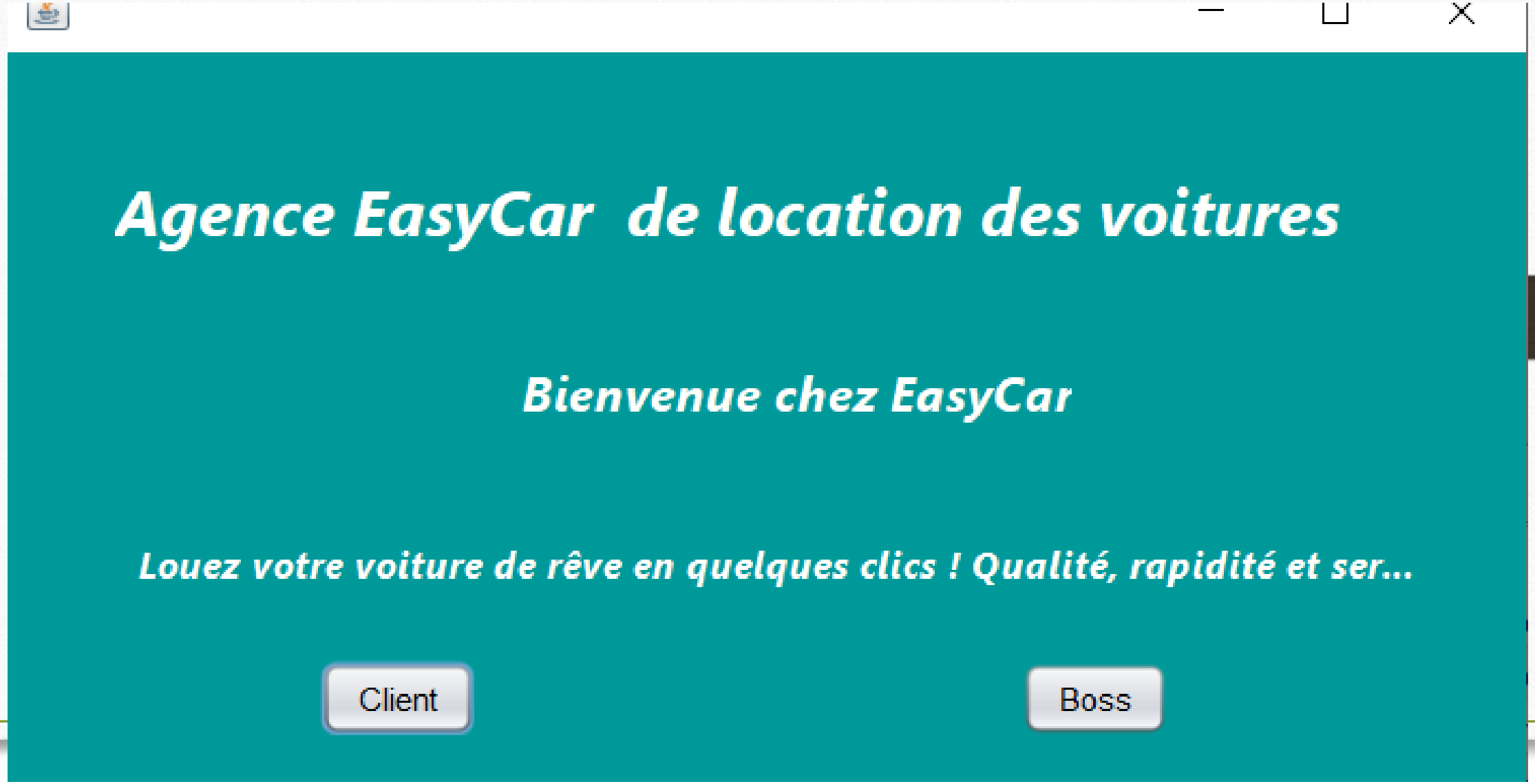
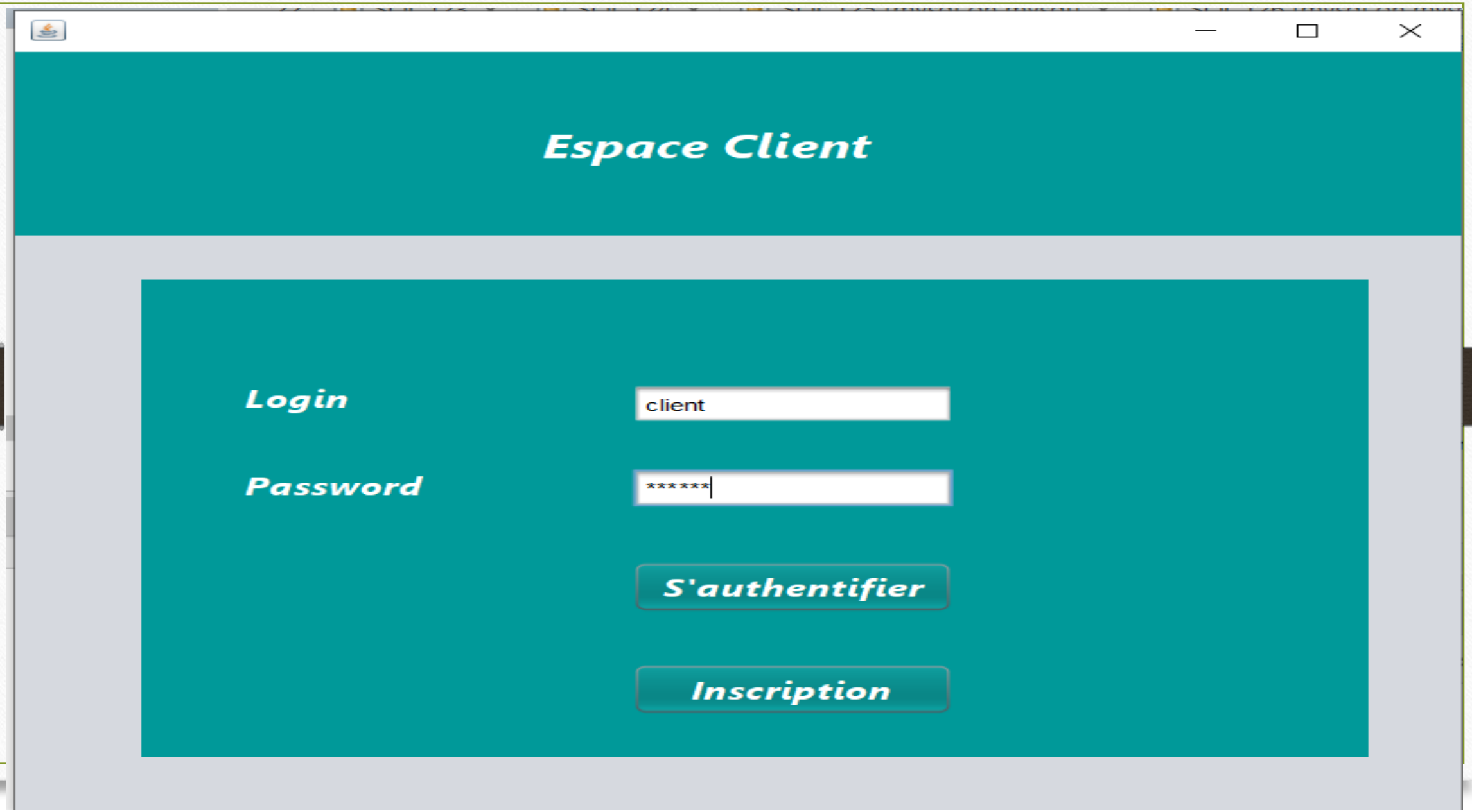


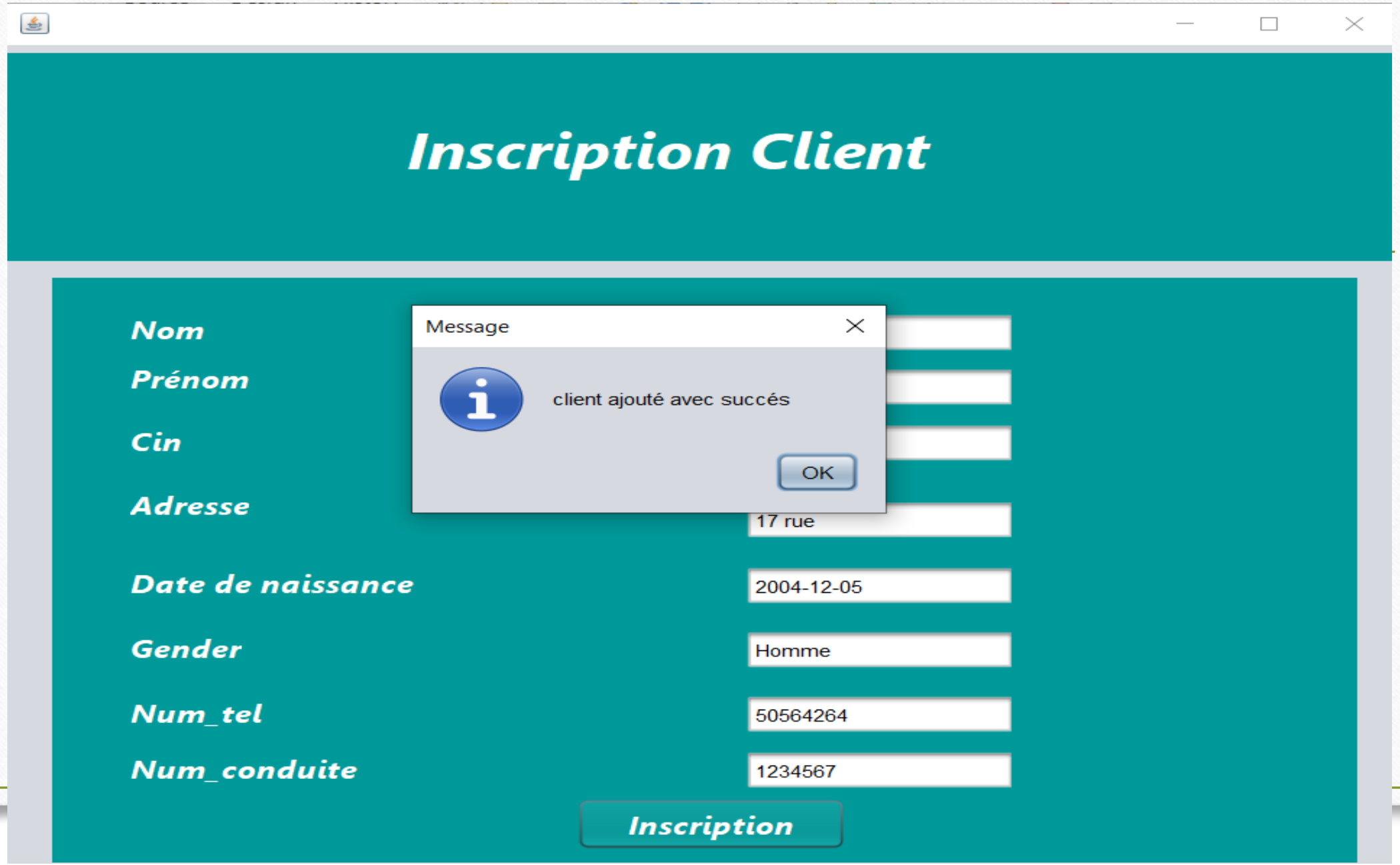
Diagramme cas d'utulisation



Page d'accueil

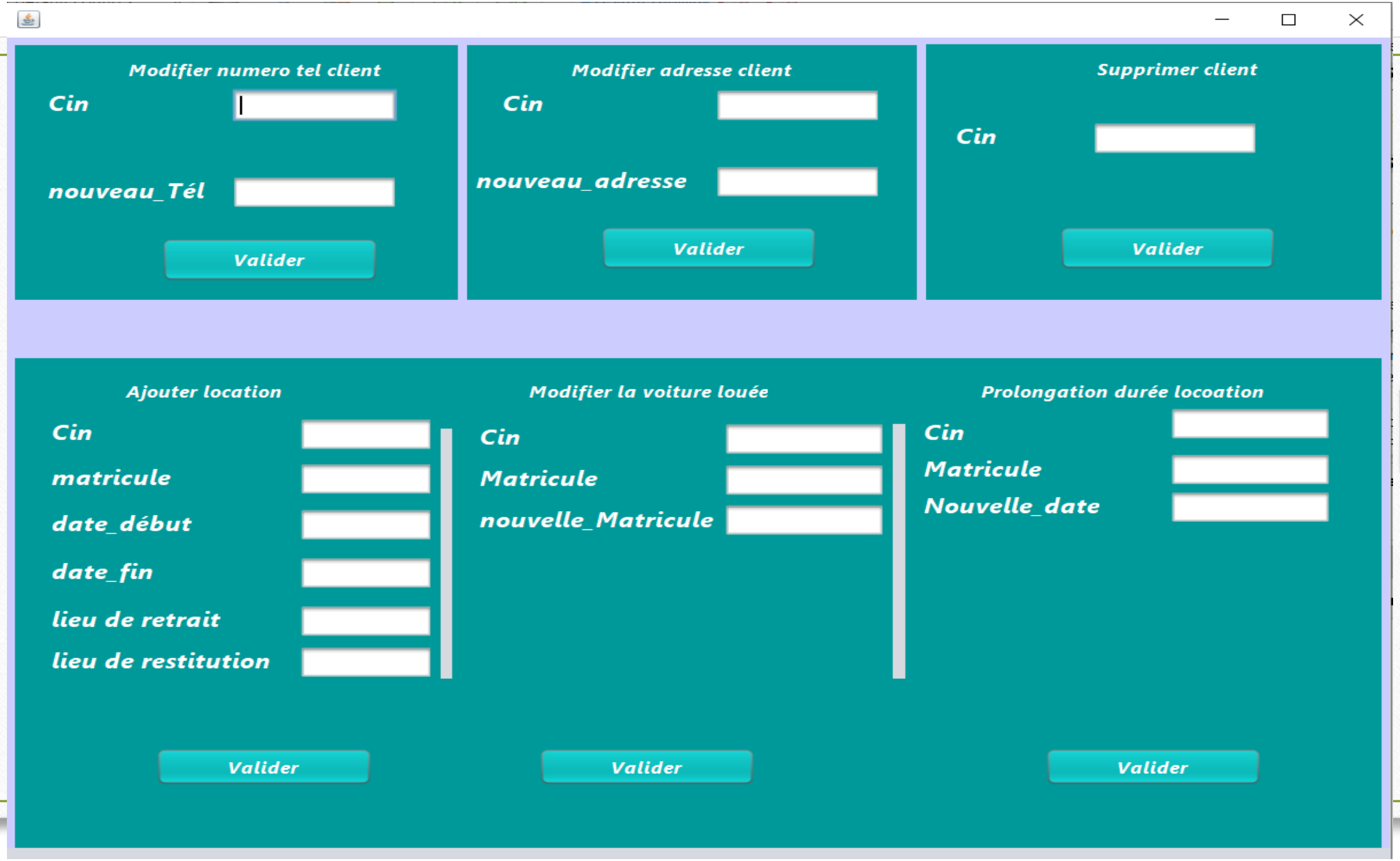



Espace Client
Login
client
Password
******
S'authentifier
Inscription

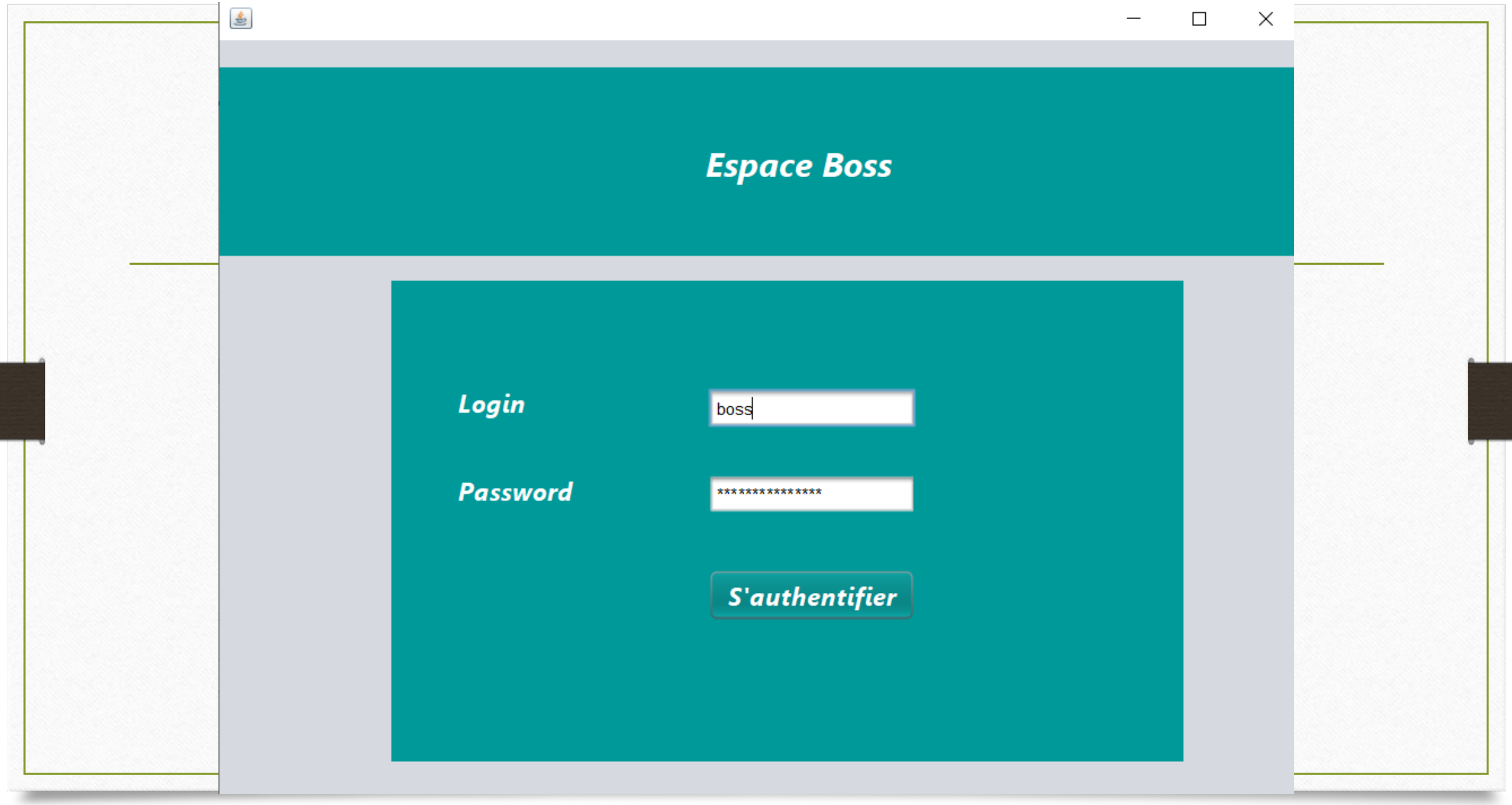

Inscription Client
Nom
Prénom
Cin
Adresse
Date de naissance
Gender
Num_tel
Num_conduite
17 rue
2004-12-05
Homme
50564264
1234567
Inscription
Message
client ajouté avec succés
OK

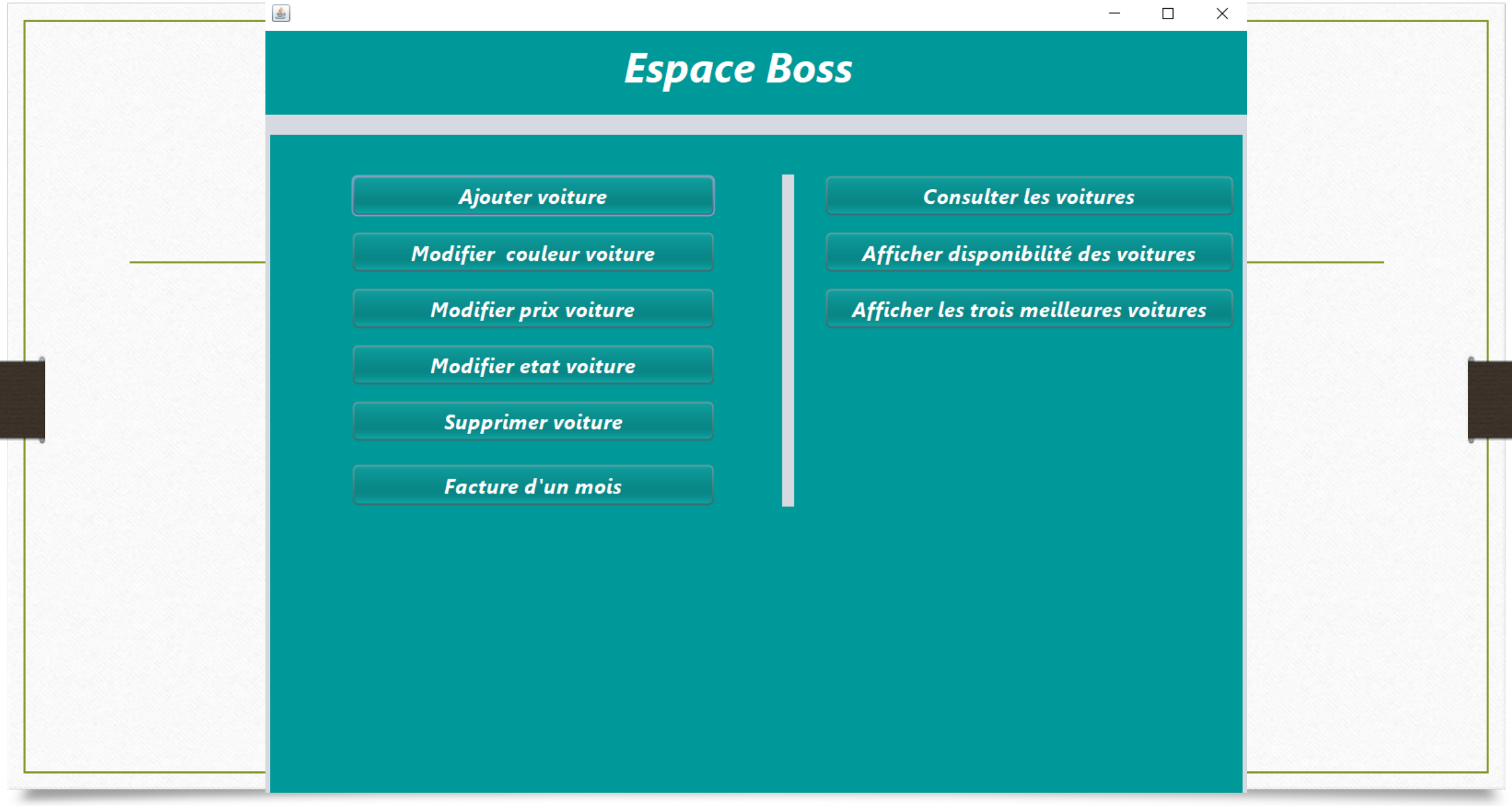
Espace Client

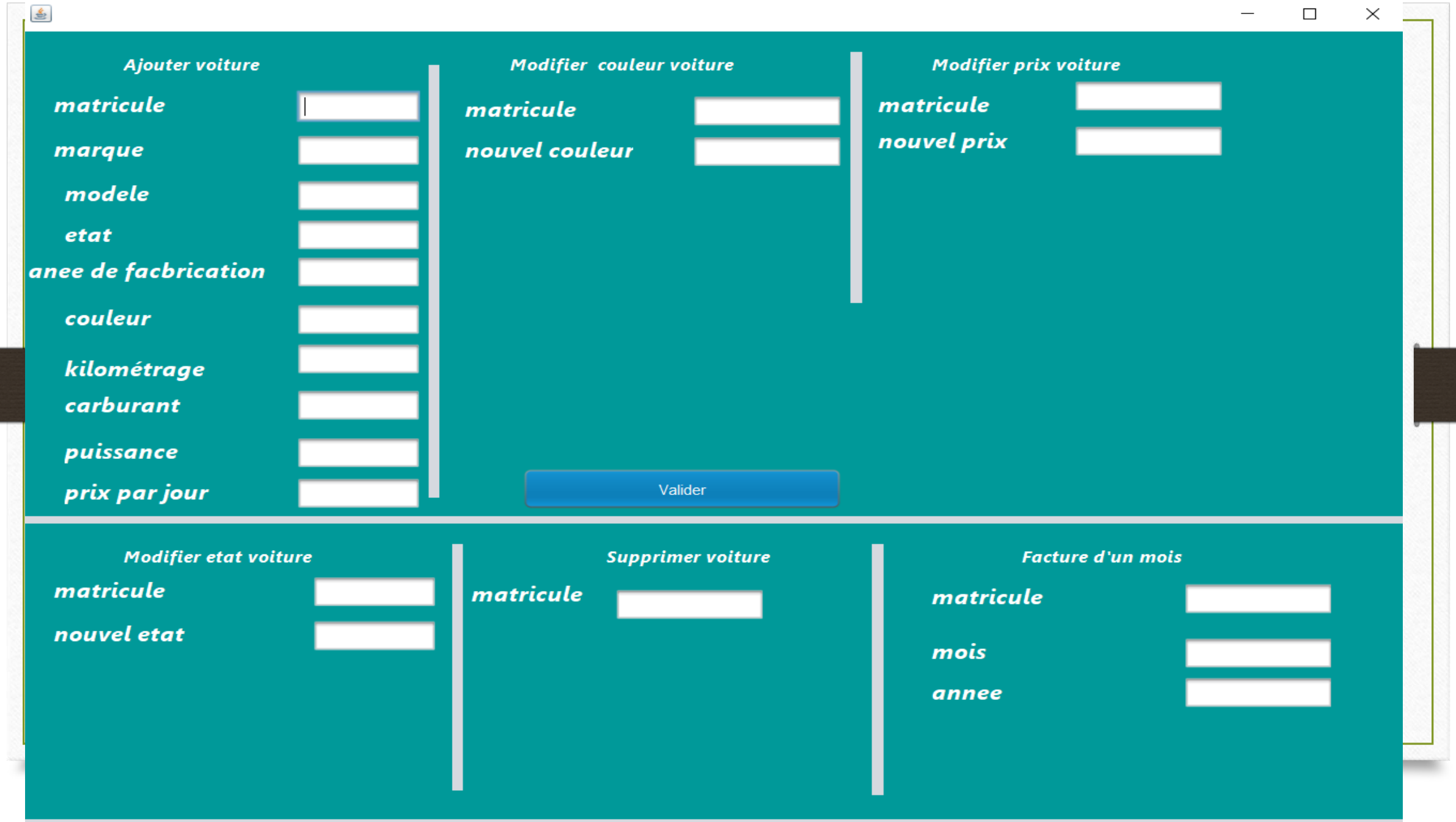
- Modifier numero tel client
- Modifier adresse client
- Supression Client
- Ajouter location
- Modifier la voiture louée
- Prolongation durée locoation
- Consulter les voitures
- Afficher disponibilité des voitures
- Afficher les trois meilleures voitures
- Info sur client
- Vérifier régularité
- Facture client
- Vérifier pénalité client
- Supprimer locoation


Modifier numero tel client
Cin
nouveau_Tél
Valider
Modifier adresse client
Cin
nouveau_adresse
Valider
Supprimer client
Cin
Valider
Ajouter location
Cin
matricule
date_début
date_fin
lieu de retrait
lieu de restitution
Valider
Modifier la voiture louée
Cin
Matricule
nouvelle_Matricule
Valider
Prolongation durée locoation
Cin
Matricule
Nouvelle_date
Valider


Espace Client
Modifier numero tel client
Modifier adresse client
Supression Client
Ajouter location
Modifier la voiture louée
Prolongation durée locoation
Consulter les voitures
Afficher disponibilité des voitures
Afficher les trois meilleures voitures
Info sur client
Vérifier régularité
Facture client
Vérifier pénalité client
Supprimer locoation
voiture numero 1 ; matricule: ABC123
voiture numero 2 ; matricule: BCD890
voiture numero 3 ; matricule: DEF456


Espace Boss
Login
boss
Password
****
S'authentifier


Espace Boss
Ajouter voiture
Modifier couleur voiture
Modifier prix voiture
Modifier etat voiture
Supprimer voiture
Facture d'un mois
Consulter les voitures
Afficher disponibilité des voitures
Afficher les trois meilleures voitures


Ajouter voiture
matricule
marque
modele
etat
anee de facbrication
couleur
kilométrage
carburant
puissance
prix par jour
Modifier couleur voiture
matricule
nouvel couleur
Modifier prix voiture
matricule
nouvel prix
Valider
Modifier etat voiture
matricule
nouvel etat
Supprimer voiture
matricule
Facture d'un mois
matricule
mois
annee

```
CREATE OR REPLACE PROCEDURE inserer_voiture (
    mat VARCHAR2,
    marque VARCHAR2,
    modele VARCHAR2,
    annee_de_fabrication INT,
    couleur VARCHAR2,
    kilometrage INT,
    carburant VARCHAR2,
    puissance INT,
    disponibilite NUMBER,
    Voiturecol VARCHAR2,
    Prix_par_jour INT,
    etat NUMBER
)
IS
    verif voiture.matricule%type;
BEGIN
    select matricule
    into verif
    from voiture
    where matricule =mat;
    IF (verif is not null) THEN
        DBMS_OUTPUT.PUT_LINE('La voiture ne peut pas être insérée, matricule déjà existant.');
    END IF;
    EXCEPTION
    when  no_data_found THEN

        INSERT INTO voiture (matricule, marque, modele, annee_de_fabrication, couleur, kilometrage, carburant, puissance, disponibilite, Voiturecol, Prix_par_jour, etat)
        VALUES (mat, marque, modele, annee_de_fabrication, couleur, kilometrage, carburant, puissance, disponibilite, Voiturecol, Prix_par_jour, etat);


        DBMS_OUTPUT.PUT_LINE('Insertion effectuée avec succès.');


   -- CLOSE c1;
END;
```

```sql
/*-------------------------modifier couleur voiture---------------------------*/
CREATE OR REPLACE PROCEDURE modifier_couleur_voiture (
    mat  voiture.matricule%type,
    coul voiture.couleur%type
)
IS
    CURSOR c1 IS
        SELECT *
        FROM voiture
        WHERE matricule = mat;
    rec c1%rowtype;

BEGIN
    OPEN c1;
    FETCH c1 INTO rec;

    IF c1%FOUND THEN
        UPDATE voiture
        SET couleur = coul
        WHERE matricule = mat;

        DBMS_OUTPUT.PUT_LINE('Modification effectuée avec succès.');
    ELSE
        DBMS_OUTPUT.PUT_LINE('La voiture n existe pas.');
    END IF;

    CLOSE c1;
END;
```

```
-----------------------------modif  prix voiture-------------------------------------
create or replace procedure modifier_prix_voiture
(
    mat  voiture.matricule%type,
    prix voiture.prix_par_jour%type
)
is

cursor c1 is
select *
from voiture
where matricule = mat;

rec c1%rowtype;
begin
    open c1;
    fetch c1 into rec;
    if(c1%found) then
        update voiture
        set voiture.prix_par_jour=prix
        where matricule =mat;

        dbms_output.put_line('modification effectuee avec succees');
    else
        dbms_output.put_line('voiture n exisite pas');
    end if;

    close c1;
end;
```

```
----------------------modif etat voiture----------------------------------------
create or replace procedure modifier_etat_voiture
(
    mat  voiture.matricule%type,
    et voiture.etat%type
)
is

cursor c1 is
select *
from voiture
where matricule = mat;

rec c1%rowtype;
begin
    open c1;
    fetch c1 into rec;
    if(c1%found) then
        update voiture
        set voiture.etat=et
        where matricule =mat;

        dbms_output.put_line('modification effectuee avec succees');
    else
        dbms_output.put_line('voiture n exisite pas');
    end if;

    close c1;
end;
```

```
--------------------delete voiture-----------------------------------------------

create or replace procedure delete_voiture
(
    mat voiture.matricule%type

)
is
cursor c1 is
select *
from voiture
where matricule = mat;

rec c1%rowtype;
begin
    open c1;
    fetch c1 into rec;
    if(c1%found) then
        delete from voiture
        where matricule=mat;

        dbms_output.put_line('delete effectuee avec succees');
    else
        dbms_output.put_line('voiture non trouvee');
    end if;

    close c1;
end;
```

```
create or replace procedure inseret_client
(
    cin_cl client.cin%type,
    p_nom client.nom%type,
    p_prenom client.prenom%type,
    p_date_naiss client.date_naiss%type,
    p_numero_tel client.numero_tel%type,
    p_num_conduite client.num_conduite%type,
    p_gender client.gender%type,
    p_adresse client.adresse%type

)

is
cursor c1 is
select *
from client
where cin = cin_cl;
rec c1%rowtype;
begin
    open c1;
    fetch c1 into rec;
    if(c1%found) then
        dbms_output.put_line('client ne peut pas etre insere');
    else

        insert into client (cin, nom, prenom, date_naiss, numero_tel, num_conduite, gender, adresse)
        values (cin_cl, p_nom, p_prenom, p_date_naiss, p_numero_tel, p_num_conduite, p_gender, p_adresse);

        dbms_output.put_line('insertion effectuee avec succees');
    end if;
    close c1;
end;
```

```
create or replace procedure inseret_client
(
    cin_cl client.cin%type,
    p_nom client.nom%type,
    p_prenom client.prenom%type,
    p_date_naiss client.date_naiss%type,
    p_numero_tel client.numero_tel%type,
    p_num_conduite client.num_conduite%type,
    p_gender client.gender%type,
    p_adresse client.adresse%type

)

is
cursor c1 is
select *
from client
where cin = cin_cl;
rec c1%rowtype;
begin
    open c1;
    fetch c1 into rec;
    if(c1%found) then
        dbms_output.put_line('client ne peut pas etre insere');
    else

        insert into client (cin, nom, prenom, date_naiss, numero_tel, num_conduite, gender, adresse)
        values (cin_cl, p_nom, p_prenom, p_date_naiss, p_numero_tel, p_num_conduite, p_gender, p_adresse);

        dbms_output.put_line('insertion effectuee avec succees');
    end if;
    close c1;
end;
```

```
------------------------------modif tel client-------------------------------------
create or replace procedure modif_tel_client
(
   cin_cl client.cin%type,
   tel client.numero_tel%type

)
is
c client.cin%type;

begin
   select cin
   into c
   from client
   where cin = cin_cl;

   update client
   set numero_tel=tel
   where cin = cin_cl;
   dbms_output.put_line('operation effectuee avec succes' );

   exception
    when no_data_found then
      dbms_output.put_line('operation impossible, client introuvable' );

end;
```

```
-----------------------modif adresse-----------------------------------------

create or replace procedure modif_adresse_client
(
   cin_cl client.cin%type,
   adr client.adresse%type

)
is
c client.cin%type;


begin
   select cin
   into c
   from client
   where cin = cin_cl;

   update client
   set adresse=adr
   where cin = cin_cl;
   dbms_output.put_line('operation effectuee avec succes' );

   exception
    when no_data_found then
      dbms_output.put_line('operation impossible, client introuvable' );

end;
```

```
------------------------------delete client-----------------------------------------
create or replace procedure delete_client
(  cin_cl client.cin%type

)


is
cursor c1 is
select *
from client
where cin = cin_cl;

rec c1%rowtype;
begin
    open c1;
    fetch c1 into rec;
    if(c1%found) then
        delete from client
        where cin =cin_cl;

        dbms_output.put_line('delete effectuee avec succees');
    else
        dbms_output.put_line('client non trouvee');
    end if;

    close c1;
end;
```

```
------------------fonction verif panne-----------------------------------
create or replace function etat_voiture
(       mat in Locations.matricule%type
)return number
is
     etat voiture.etat%type;
begin
   select etat
    into etat
    from voiture
    where matricule =mat;

    return(etat);

exception
      when no_data_found then
        return(2);
end;
```

```
--------------------fonction verif dispo-----------------------------------------------
create or replace function dispo_voiture
(
  mat in voiture.matricule%type
  )return number
  is
   cursor c1 is
   select *
   from locations
   where matricule=mat and sysdate<date_fin and sysdate>date_debut;

   cursor c2
   is select *
   from voiture
   where matricule =mat;


   rec c1%rowtype;
   rec2 c2%rowtype;
  begin
  open c1;
  open c2;
  fetch c2 into rec2;
  fetch c1 into rec;
  if(c1%found ) then
     return(0) ;
  else if (c2%notfound) then
     return (2);      ----voiture inexstant dans la table mere
  else
     return(1);
end if;
   close c1;
   close c2;
end;
```

```
----------------------------dispo client-----------------------------------------
create or replace function dispo_client
(
    cin_cl client.cin%type
)return number
is
 a number;

BEGIN
    select cin
    into a
    from client
    where cin = cin_cl;

    return(1);

exception
    when no_data_found then
        return(0);

end;
```

```
-----------------insert location---------------------------------
CREATE OR REPLACE PROCEDURE insert_location (
    mat IN locations.matricule%TYPE,
    cin IN locations.cin%TYPE,
    date_debut IN locations.date_debut%TYPE,
    date_fin IN locations.date_fin%TYPE,
    lieu_retrait IN locations.lieu_de_retrait%TYPE,
    lieu_restitution IN locations.lieu_de_restitution%TYPE
)
IS
BEGIN
    IF ( dispo_voiture(mat) = 1 and etat_voiture(mat) = 1 and  dispo_client(cin)=1) THEN
        INSERT INTO locations (matricule, cin, date_debut, date_fin, lieu_de_retrait, lieu_de_restitution)
        VALUES (mat, cin, date_debut, date_fin, lieu_retrait, lieu_restitution);
        DBMS_OUTPUT.PUT_LINE('Insertion effectuée avec succès.');
    ELSIF (dispo_voiture(mat) = 0) THEN
        DBMS_OUTPUT.PUT_LINE('Voiture déjà louée.');
    elsif(dispo_client(cin) = 0) then
            DBMS_OUTPUT.PUT_LINE('client inexistant' );
    elsif (ETAT_VOITURE(mat) = 2) then
                DBMS_OUTPUT.PUT_LINE('Voiture n existe pas.');

    ELSE
        DBMS_OUTPUT.PUT_LINE('Voiture ne peut pas être louée car elle est en panne.');
    END IF;
END;
```

```
----------------------modif location_nouvelle_voiture------------------------------

create or replace procedure modif_location_nouvelle_voiture
(
    mat1 voiture.matricule%type,
    mat2 voiture.matricule%type,
    cin1 client.cin%type
)
is

begin

    if(dispo_voiture(mat1) = 0 and dispo_voiture(mat2) =1    ) then
        update locations
        set matricule=mat2
        where matricule =mat1
        and cin=cin1;

        dbms_output.put_line('modification effectuee avec succees');
    else
        dbms_output.put_line('operation impossible');
    end if;

end;
```

```
----------------------modif location_corrdonnes----------------------------------
create or replace procedure modif_location_date_fin
(
        mat voiture.matricule%type,
        df locations.date_fin%type

)
is
   d locations.date_fin%type;

begin

    if(dispo_voiture(mat) = 0    ) then
        select date_fin
        into d
        from locations
        where  matricule =mat;

        if(d < df) then

        update locations
        set date_fin=df
        where matricule =mat
        and cin=cin_cl ;

        dbms_output.put_line('modification effectuee avec succees');

        else
                    dbms_output.put_line('nouvelle date erronee');
        end if;
    else
        dbms_output.put_line('voiture n exisite pas');
    end if;

end;
```

```
----------delete location-------------------------------------------------------
create or replace procedure delete_location
(
    mat voiture.matricule%type,
    cin_cl client.cin%type
)
is
    rec locations%rowtype;
begin
     select *
     into rec
     from locations
     where matricule =mat and cin = cin_cl;

     delete from locations
     where matricule =mat and cin = cin_cl;
      dbms_output.put_line('operation effectue avec succes');

exception
    when no_data_found then
        dbms_output.put_line('donne invalide, il n y a pas de locations avec ces parametres');
end;
```

```
------------------------function verif penalite-------------------------------------
create or replace function verif_penalite
(
    mat in voiture.matricule%type,
    cin in client.cin%type
)
return number
is
    duree number;

begin
    select (date_fin - date_debut)
    into duree
    from locations
    where matricule = mat and cin = cin;

    if (duree > 30) then
        return (1);
    else
        return (0);
    end if;

exception
    when no_data_found then
        return (2);
end;
```

```
----------------------------function prix totale--------------------------------
create or replace function prix_total(
    mat voiture.matricule%type,
    cin_cl client.cin%type
)return NUMBER

is
    duree number;
    p voiture.prix_par_jour%type;
begin
    select (date_fin - date_debut),prix_par_jour
    into duree,p
    from locations inner join Voiture
    on locations.matricule=voiture.matricule
    where locations.matricule = mat and locations.cin=cin_cl;
    if(duree <30) THEN
        return(duree*p);
    else
        return(duree*p*1.5);
    end if;




exception
    when no_data_found THEN
        return 0;
end;
```

```
----------------------afficher etat des voitures----------------------------
create or replace procedure  affiche_tout_voiture

is
cursor c1 is
select matricule ,marque ,modele  ,etat
from voiture;


begin

for ligne in c1 loop
  if(ligne.etat =0) then
     dbms_output.put_line('matricule:' || ligne.matricule || ' marque: '||ligne.marque || ' modele: ' || ligne.modele || ' etat: en panne ' );
  else
       dbms_output.put_line('matricule:' || ligne.matricule || ' marque: '||ligne.marque || ' modele: ' || ligne.modele || ' etat: en marche ' );
  end if;
end loop;
end;
```

```
-----------------------affiche_dispo_tot---------------------------------
create or replace procedure afficher_dispo_tot

is

cursor c1 is
select matricule ,marque ,modele  ,etat
from voiture
where voiture.matricule not in(select matricule from locations);

cursor c2 is
select matricule  ,marque ,modele  ,etat
from voiture
where voiture.matricule in(select matricule from locations);


begin
    dbms_output.put_line('Voitures disponibles:');
for ligne in c1 loop
    dbms_output.put_line('matricule:' || ligne.matricule || ' marque: ' || ligne.marque || ' modele: ' || ligne.modele );
end loop;

 dbms_output.put_line('Voitures non disponibles:');
for ligne in c2 loop
    dbms_output.put_line('matricule:' || ligne.matricule || ' marque:  '|| ligne.marque || ' modele: ' ||ligne.modele );
end loop;
end;
```

```
------------------------affiche meilleur-----------------------------
create or replace procedure affiche_meilleur_voiture

is



cursor c1 is
select matricule,count(*)
from locations
group by matricule
order by count(*) desc
fetch first 3 rows only;

i number :=1;
begin
for ligne in c1 loop

    dbms_output.put_line('voiture numero '||i||' ; matricule:' || ligne.matricule  );
    i:=i+1;
end loop;
end;
```

```
------------client regulier >3 fois--------------------
create or replace procedure client_regulier
(
   cin_cl  client.cin%type

)
is
b number;
begin
     select count(*)
     into b
     from locations
     where cin =cin_cl;
     if(b >= 3) then
        dbms_output.put_line('vous etes un client regulier');
    elsif(b> 0) then
        dbms_output.put_line('vous n etes pas un client regulier');

    else
       dbms_output.put_line('vous n avez pas effectue des locations');
     end if;
end;
```

```
------------question 5--------------------
create or replace procedure client_info
(
   cin_cl  client.cin%type

)
is
b number;

cursor c1
is select *
from LOCATIONS
where cin = cin_cl;
begin
     select count(*)
     into b
     from locations
     where cin =cin_cl;
    dbms_output.put_line('nombre de fois:' || b);
     for ligne in c1 loop
          dbms_output.put_line('Voiture:' || ligne.matricule || '       duree' || (ligne.date_debut-ligne.date_fin) );

     end loop;

end;
```

Conclusion

Le développement de cette application de gestion de location de voitures a été une expérience enrichissante et formatrice. Nous avons pu mettre en œuvre de manière concrète les notions théoriques acquises en programmation orientée objet, conception de bases de données relationnelles, interfaces graphiques et communication entre une application Java et une base de données Oracle via JDBC.

Ce projet nous a permis de :

- Maîtriser l'utilisation de **Java avec Swing** pour créer des interfaces adaptées à différents types d'utilisateurs (administrateur et client).
- Apprendre à **gérer des rôles et des droits d'accès**, en assurant la sécurité des opérations.
- Développer des **procédures stockées** et des **triggers PL/SQL** robustes pour garantir l'intégrité des données.
- Travailler en **collaboration**, en appliquant une approche structurée pour le développement logiciel.