Projet CORBA :

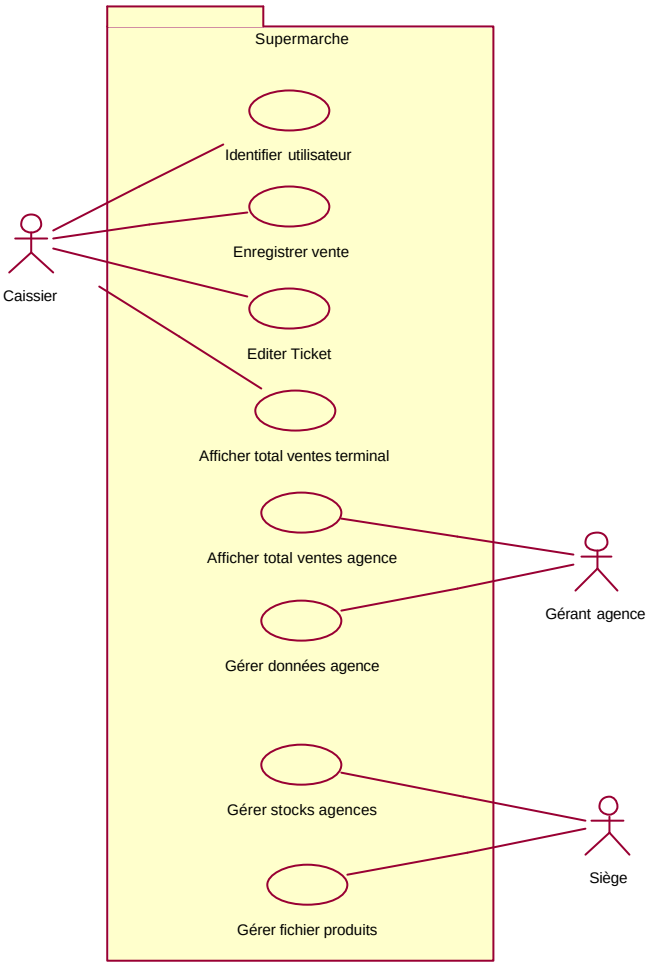
gestion de supermarchés

Catherine FRANCO, Laetitia SOULIÉ et David ROUSSE

SOMMAIRE

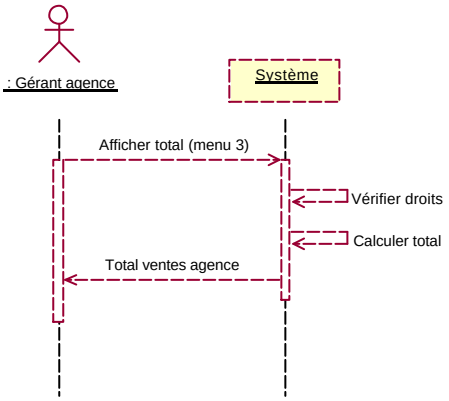
I - Spécifications

a) Cas d'utilisation

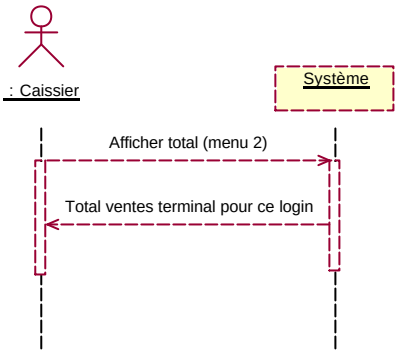


b) Scénarios

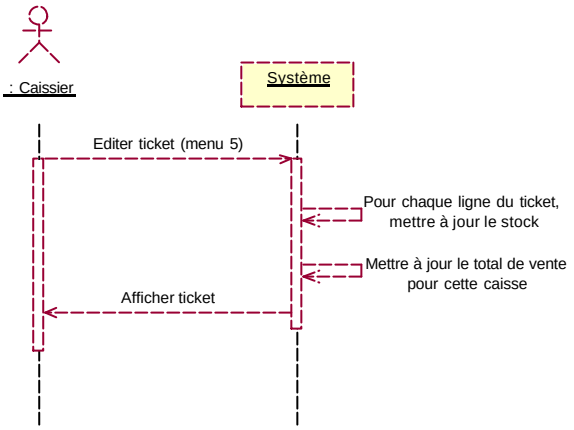
- *Afficher total ventes agences*



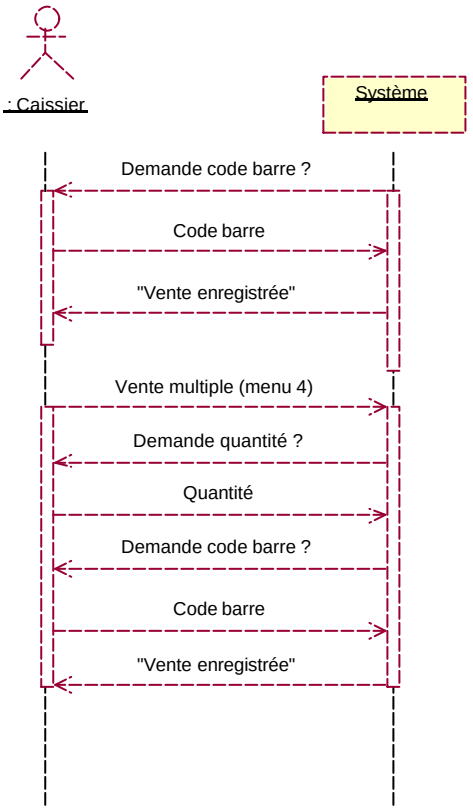
- *Afficher total ventes terminal*



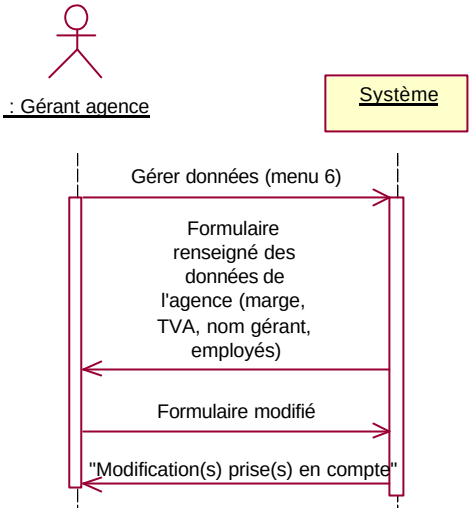
- *Editer ticket*



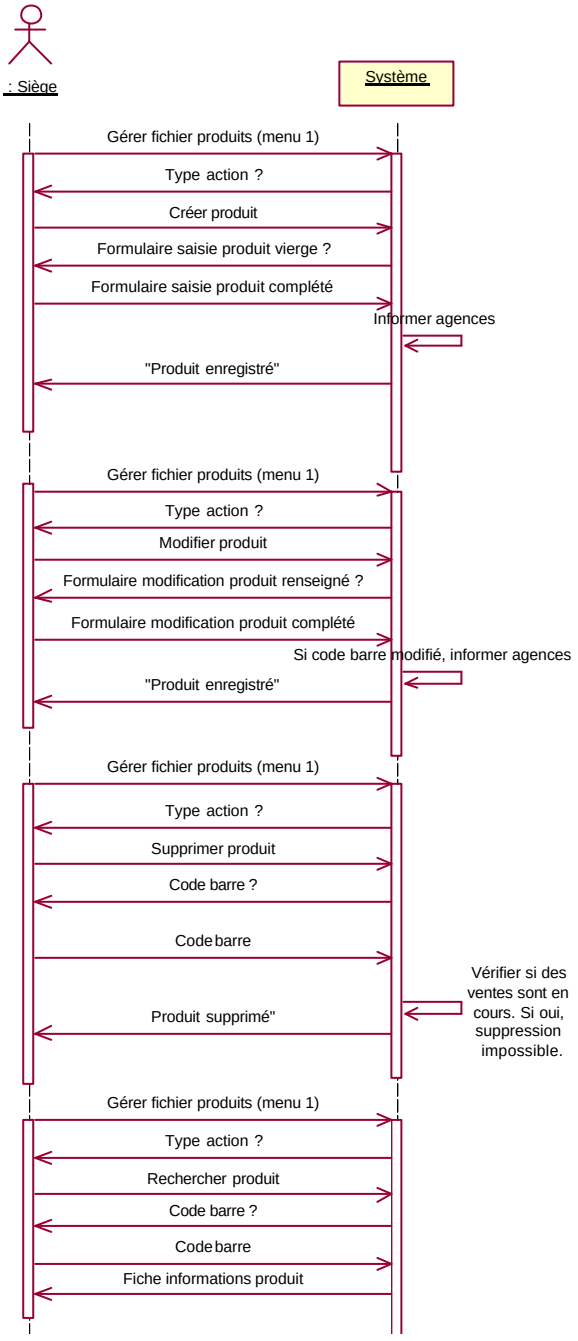
- *Enregistrer vente*



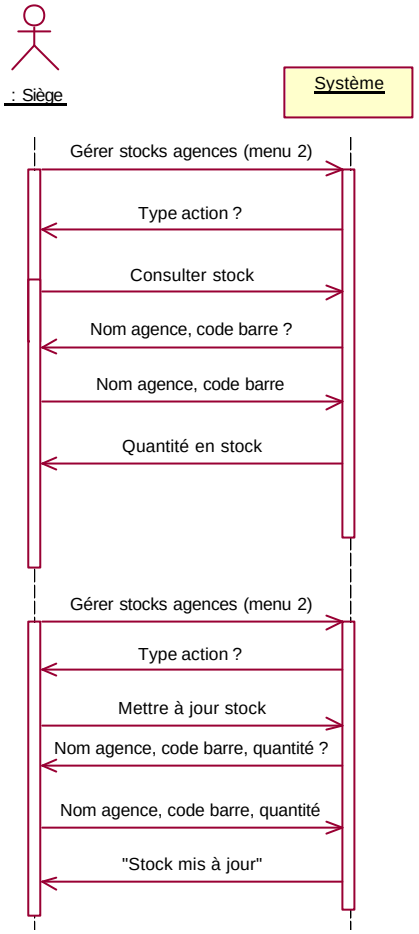
- *Gérer données agence*



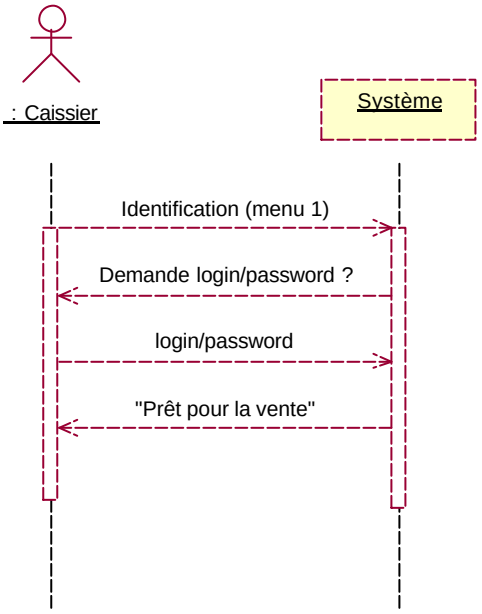
- *Gérer fichier produits*



- *Gérer stocks agence*

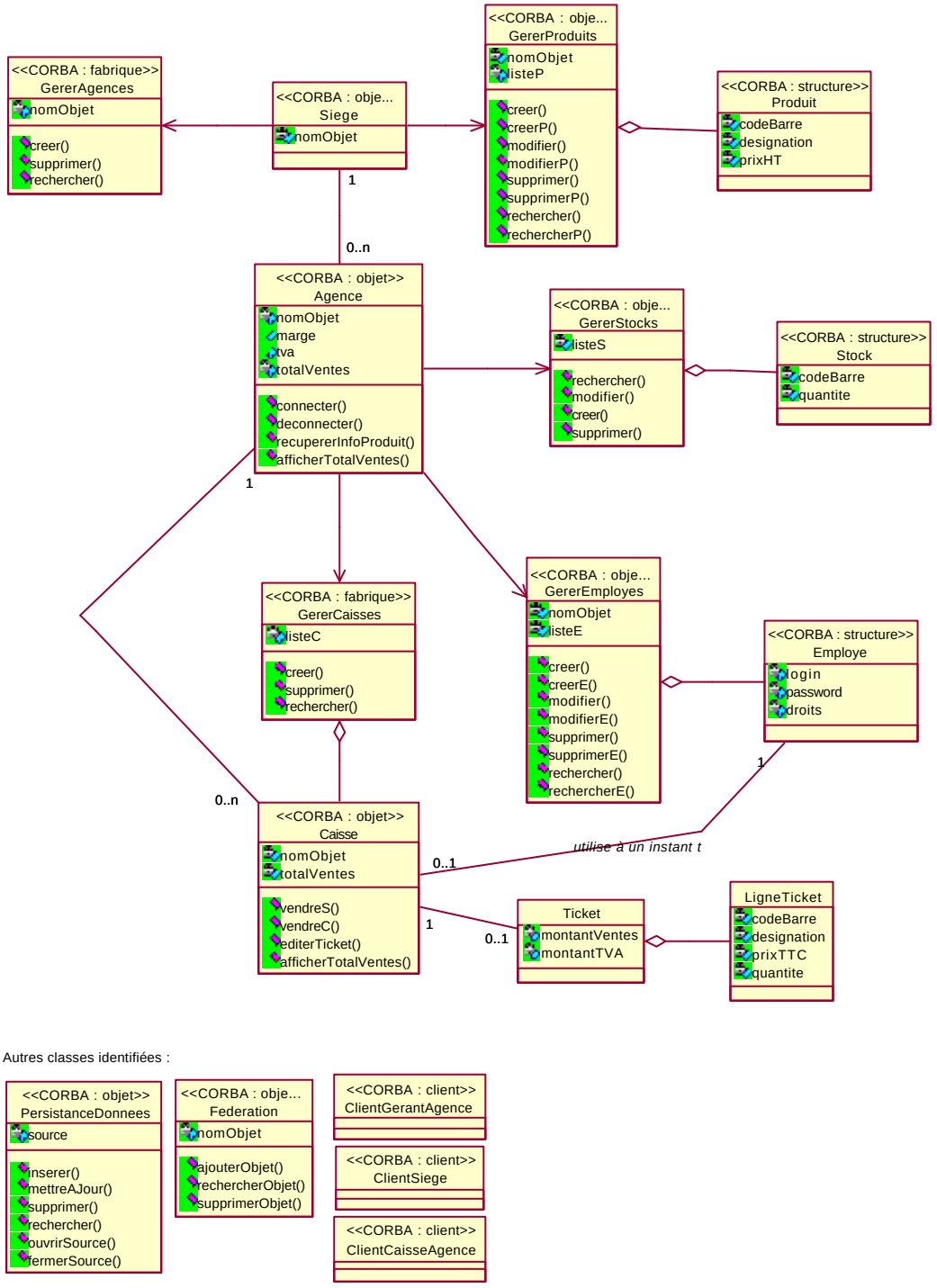


- *Identifier utilisateur*



II - Analyse

a) Diagramme des classes



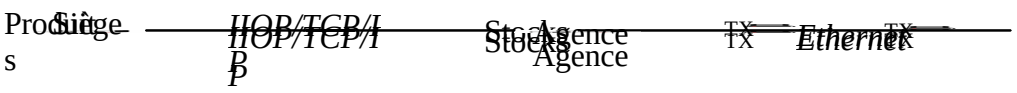
b) Règles de gestion

On considère que pour une agence donnée, le taux de TVA est le même pour tous les produits.
On mettra à jour le stock lorsque le ticket est édité.

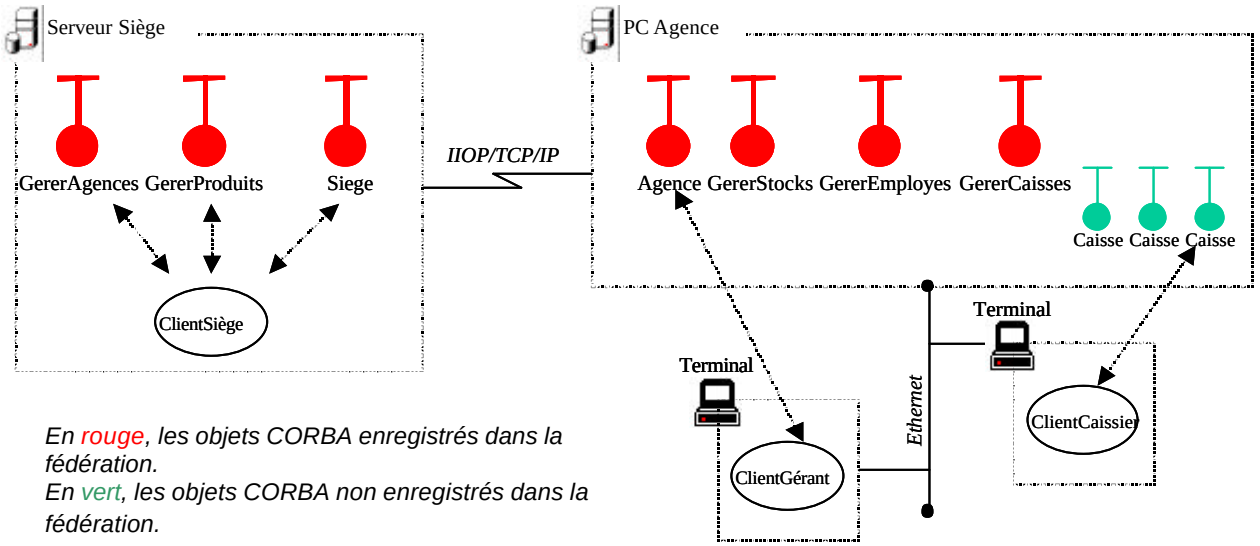
III - Conception

a) Architecture

- *Vue générale*

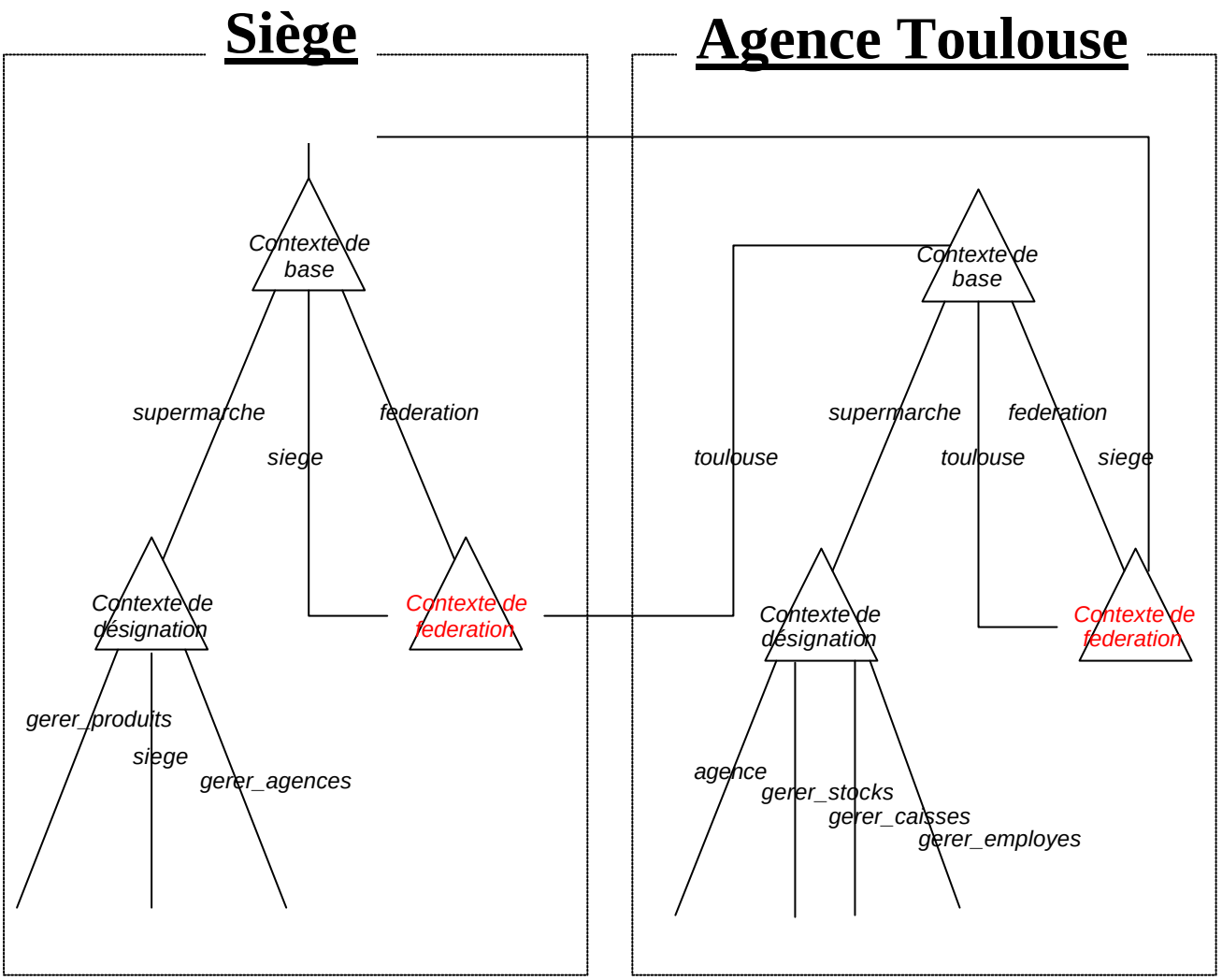


- *Vue CORBA*



En rouge, les objets CORBA enregistrés dans la fédération.
En vert, les objets CORBA non enregistrés dans la fédération.

- *Fédération*



Ainsi, par exemple, on pourra consulter les stocks des agences depuis le siège en nommant les objets ainsi :
federation.*toulouse*.supermarche.gerer_stocks
federation.*pau*.supermarche.gerer_stocks
federation.*marseille*.supermarche.gerer_stocks
ect …

b) *Contrat IDL*

```
module supermarche {

  exception CaisseException {
    string raison;
  };

  exception GererCaissesException {
    string raison;
  };

  exception AgenceException {
    string raison;
  };
```

```
exception GererAgencesException {
  string raison;
};

exception SiegeException {
  string raison;
};

exception GererStocksException {
  string raison;
};

exception GererProduitsException {
  string raison;
};

exception GererEmployesException {
  string raison;
};

struct Employe {
  string login;
  string password;
  string droit;
};

struct Stock {
  string codeBarre;
  short qte;
};

struct Produit {
  string codeBarre;
  string designation;
  double prixHT;
};

interface Base {
  readonly attribute string nomObjet;
};

interface Caisse : Base {
  attribute double totalVentes;
  readonly attribute string agence;
  readonly attribute string caissier;
  void vendreS(in string codeBarre) raises (CaisseException);
  void vendreC(in string codeBarre, in short qte) raises (CaisseException);
  void editerTicket() raises (CaisseException);
};

typedef sequence<Caisse> listeCaisses;
```

```
  interface GererCaisses : Base {
    readonly attribute listeCaisses listeC;
    Caisse creer(in string login, in string agence, in string loginCaissier) raises
(GererCaissesException);
    void supprimer(in string login) raises (GererCaissesException);
    Caisse rechercher(in string login) raises (GererCaissesException);
  };

  typedef sequence<Stock> listeStocks;

  interface GererStocks : Base {
    readonly attribute listeStocks listeS;
    void creer (in string codeBarre, in short qte) raises (GererStocksException);
    void creerS (in Stock s) raises (GererStocksException);
    void modifier (in string codeBarre, in short qte) raises (GererStocksException);
    void modifierS (in Stock s) raises (GererStocksException);
    void supprimer (in string codeBarre) raises (GererStocksException);
    void supprimerS (in Stock s) raises (GererStocksException);
    Stock rechercher (in string codeBarre) raises (GererStocksException);
    Stock rechercherS (in Stock s) raises (GererStocksException);
  };

  typedef sequence<Employe> listeEmployes;

  interface GererEmployes : Base {
    readonly attribute listeEmployes listeE;
    void creer (in string login, in string password, in string droit) raises
(GererEmployesException);
    void creerE (in Employe e) raises (GererEmployesException);
    void modifier (in string login, in string password, in string droit) raises
(GererEmployesException);
    void modifierE (in Employe e) raises (GererEmployesException);
    void supprimer (in string login) raises (GererEmployesException);
    void supprimerE (in Employe e) raises (GererEmployesException);
    Employe rechercher (in string login) raises (GererEmployesException);
    Employe rechercherE (in Employe e) raises (GererEmployesException);
  };

  interface Agence : Base {
    readonly attribute double totalVentes;
    attribute double marge;
    attribute double TVA;
    Caisse connecter(in string login, in string password) raises (AgenceException);
    void deconnecter(in string login) raises (AgenceException);
    Produit recupererInfoProduit(in string codeBarre) raises (AgenceException);
  };

  typedef sequence<Agence> listeAgences;
```

```
  interface GererAgences : Base {
    readonly attribute listeAgences listeA;
    Agence creer(in string nomObjet) raises (GererAgencesException);
    void supprimer(in string nomObjet) raises (GererAgencesException);
    Agence rechercher(in string nomObjet) raises (GererAgencesException);
  };

  interface Siege : Base {
    Produit recupererProduit (in string codeBarre) raises (SiegeException);
  };

  typedef sequence<Produit> listeProduits;

  interface GererProduits : Base {
    readonly attribute listeProduits listeP;
    void creer (in string codeBarre, in string designation, in double prixHT) raises
(GererProduitsException);
    void creerP (in Produit p) raises (GererProduitsException);
    void supprimer (in string codeBarre) raises (GererProduitsException);
    void supprimerP (in Produit p) raises (GererProduitsException);
    void modifier (in string codeBarre, in string designation, in double prixHT) raises
(GererProduitsException);
    void modifierP (in Produit p) raises (GererProduitsException);
    Produit rechercher (in string codeBarre) raises (GererProduitsException);
    Produit rechercherP (in Produit p) raises (GererProduitsException);
  };

};
```