

# **SAÉ 2.01**

---

# **Documentation technique**

## Sommaire :

<b>1) Présentation de l'application.....</b>	<b>3</b>
<b>2) Architecture.....</b>	<b>3</b>
<b>3) Fonctionnalités.....</b>	<b>5</b>
a) Version 1.....	5
i) Créer des employés (A. B.).....	5
ii) Voir les employés (A. B.).....	5
iii) Modifier les employés (H. B.).....	6
iv) Effacer les employés (H. B.).....	6
v) Créer un compte (B. S.).....	7
vi) Créditer / Débitier sur un compte (B. S.).....	7
vii) Effectuer un virement (J. M.).....	8
viii) Clôturer un compte (J. M.).....	9
b) Version 2 (en cours de développement).....	9
i) Débit exceptionnel.....	9
ii) Simuler emprunt.....	9

## 1) Présentation de l'application

La banque DailyBank souhaite développer une application de gestion des comptes clients en utilisant JAVA-Oracle pour remplacer plusieurs outils obsolètes. Cette initiative s'inscrit dans le cadre de la restructuration de ses services bancaires.

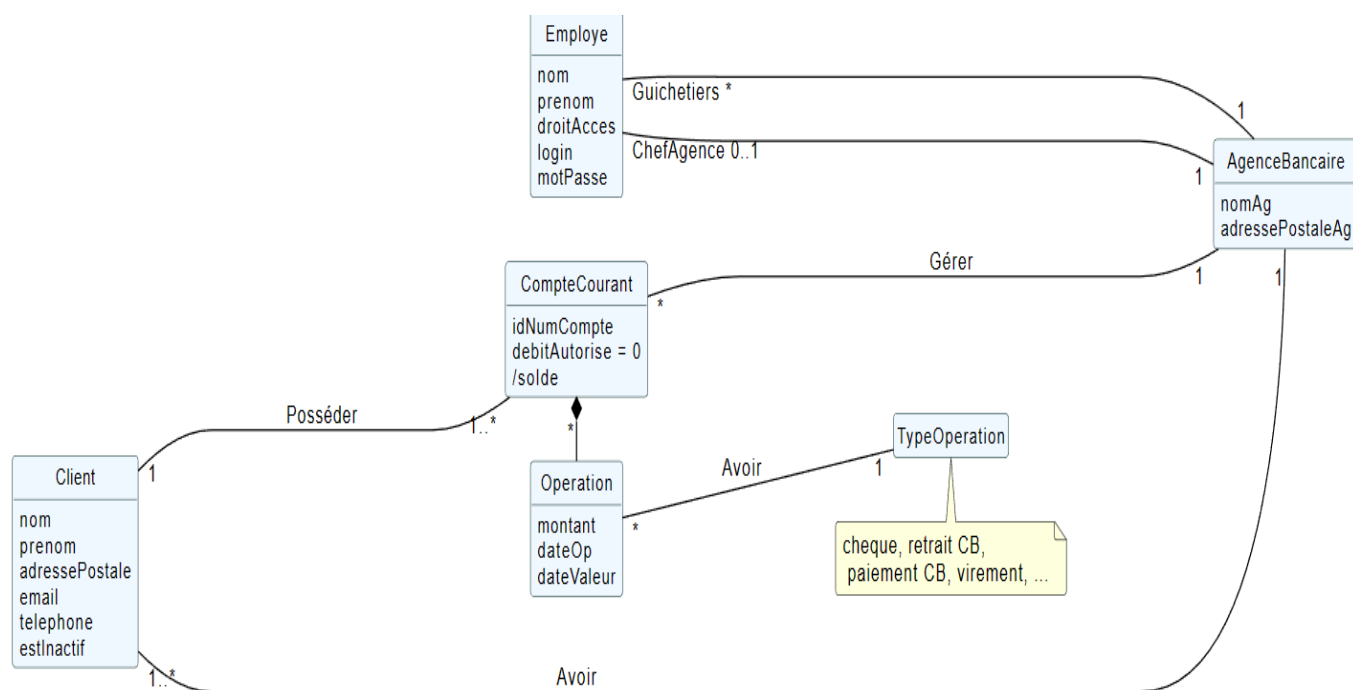
Le travail sera basé sur une application existante nommée "Daily Bank" à laquelle nous allons apporter de nouvelles fonctionnalités pour répondre aux nouveaux besoins de la banque.

Cette application sera déployée dans les 100 agences que compte son réseau.

Le rôle de l'application va être de gérer des comptes bancaires, de pouvoir débiter ainsi que créditer soit par virement bancaire soit par une personne physique auprès d'un guichet de la banque.

Elle va aussi permettre de gérer les employés : créer un employé, le modifier, le supprimer ou afficher son profil.

Le diagramme de classe de la version 0 de l'application est le suivant :

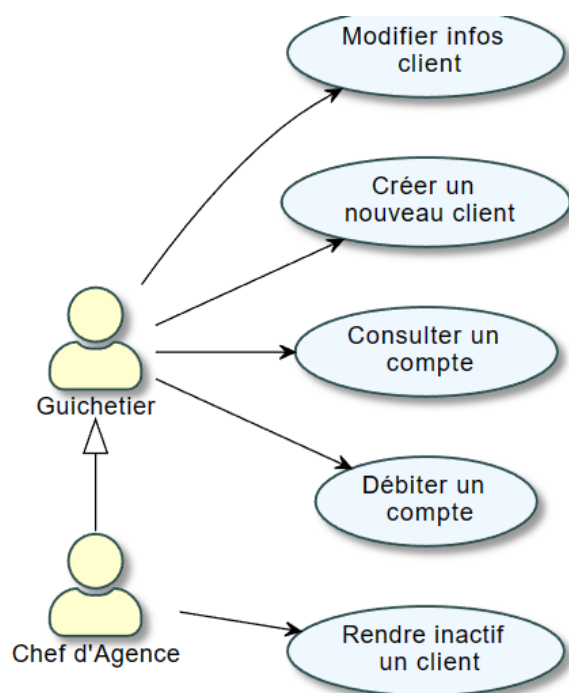


Il permet de mettre en place toutes les fonctionnalités dont l'application dispose dans sa version 0 :

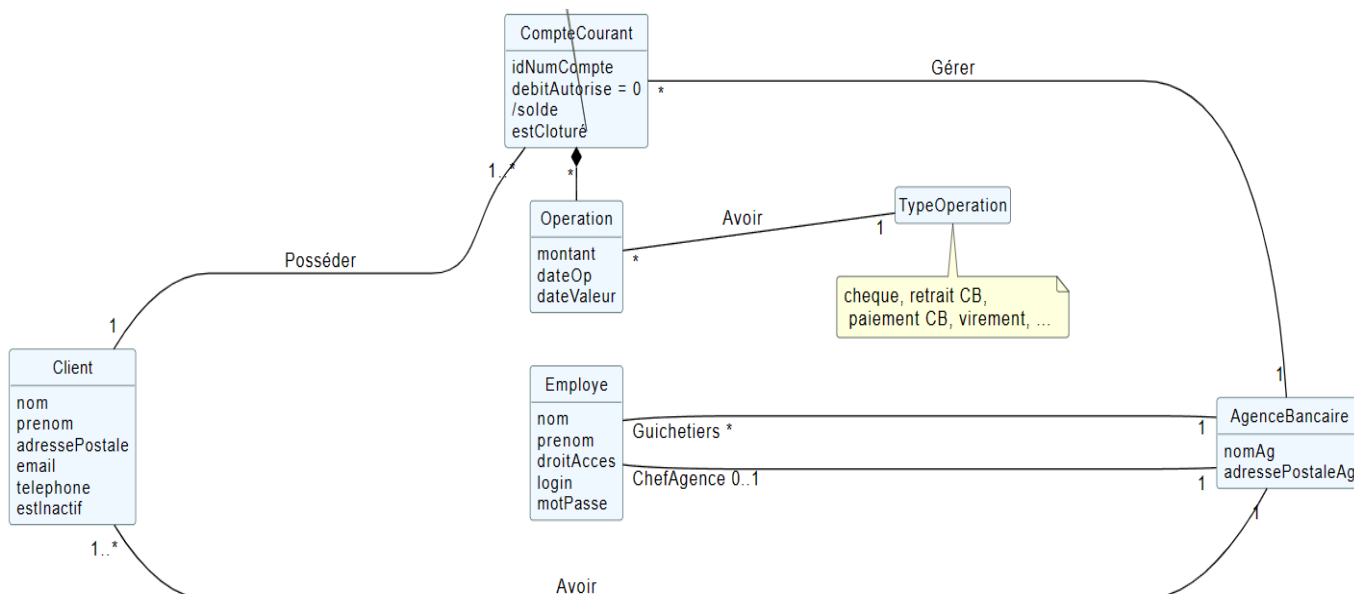
Pour un guichetier, créer un nouveau client, consulter un compte, modifier les infos d'un client et débiter un compte.

Pour un chef d'agence, rendre un client inactif ainsi que toutes les fonctionnalités d'un guichetier.

Ces fonctionnalités sont représentées par le use case suivant :



La version 0 de l'application a ensuite été améliorée, ce qui a engendré la version 1 de l'application. Dans cette version développée par notre équipe, de nouvelles fonctionnalités ont été ajoutées. L'application peut être représentée au mieux par ce diagramme de classe mis à jour :

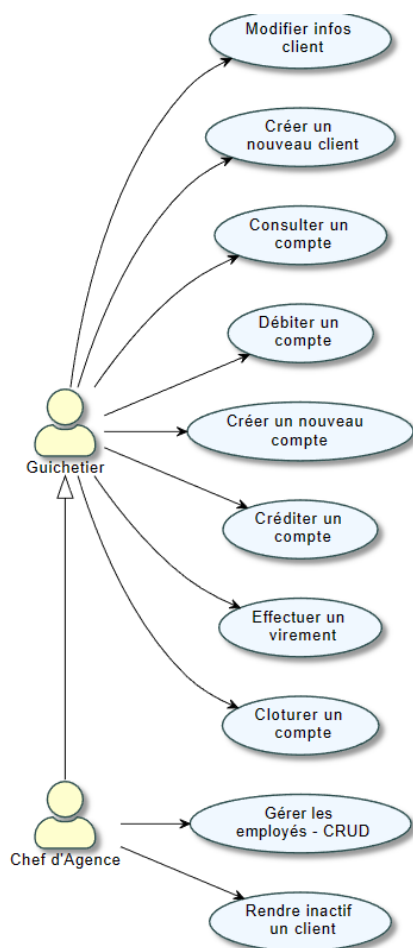


Les nouvelles fonctionnalités implémentées à ajouter aux anciennes sont les suivantes :

Pour un guichetier, créer un nouveau compte, clôturer un compte, créditer un compte et effectuer un virement entre comptes.






















































































Pour un chef d'agence, créer un profil d'employé, afficher un profil d'employé, modifier un profil d'employé et supprimer un profil d'employé.

Ces nouvelles fonctionnalités peuvent être illustrées par le use case suivant :



En résumé, l'application permet de gérer les employés et les clients de plusieurs agences bancaires. Chaque agence bancaire comporte deux types d'employés, les guichetiers et les chefs d'agence. Ces derniers ont chacun certaines fonctionnalités à dispositions présentées ci-dessus. Une agence peut avoir un seul chef d'agence ou aucun chef d'agence et au moins un guichetier. Le chef d'agence peut rendre un client inactif, cela signifie que ce client ne pourra plus effectuer d'opération sur son compte (débit, crédit, virement).

## 2) Architecture

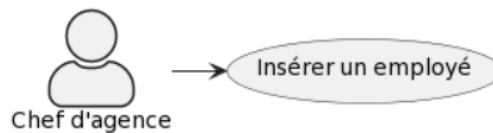
- ▼  DailyBank [Wspces\_S2\_01\_2022\_2023\_V0 dev]
  - ▼  src/main/java
    - ▼  application
      - >  DailyBankApp.java
      - >  DailyBankState.java
    - ▼  application.control
      - >  ClientEditorPane.java
      - >  ClientsManagement.java
      - >  CompteEditorPane.java
      - >  ComptesManagement.java
      - >  DailyBankMainFrame.java
      - >  EmployeEditorPane.java
      - >  EmployesManagement.java
      - >  ExceptionDialog.java
      - >  LoginDialog.java
      - >  OperationEditorPane.java
      - >  OperationsManagement.java
    - ▼  application.tools
      - >  AlertUtilities.java
      - >  CategorieOperation.java
      - >  ConstantesIHM.java
      - >  EditionMode.java
      - >  NoSelectionModel.java
      - >  PairsOfValue.java
      - >  StageManagement.java
    - ▼  application.view
      - >  ClientEditorPaneController.java
      - >  ClientsManagementController.java
      - >  CompteEditorPaneController.java
      - >  ComptesManagementController.java
      - >  DailyBankMainFrameController.java
      - >  EmployeEditorPaneController.java
      - >  EmployesManagementController.java
      - >  ExceptionDialogController.java
      - >  LoginDialogController.java
      - >  OperationEditorPaneController.java
      - >  OperationsManagementController.java
    - ▼  model.data
      - >  AgenceBancaire.java
      - >  Client.java
      - >  CompteCourant.java
      - >  Employe.java
      - >  Operation.java
      - >  TypeOperation.java
    - ▼  model.orm
      - >  Access\_BD\_AgenceBancaire.java
      - >  Access\_BD\_Client.java
      - >  Access\_BD\_CompteCourant.java
      - >  Access\_BD\_Employe.java
      - >  Access\_BD\_Operation.java
      - >  Access\_BD\_TypeOperation.java
      - >  LogToDatabase.java
    - ▼  model.orm.exception
      - >  ApplicationException.java
      - >  DataAccessException.java
      - >  DatabaseConnexionException.java
      - >  ManagementRuleViolation.java
      - >  Order.java
      - >  RowNotFoundOrTooManyRowsException.java
      - >  Table.java
    -  Readme.txt
  - ▼  src/main/resources
    - ▼  application
      - ▼  view
        - >  clienteditorpane.fxml
        - >  clientsmanagement.fxml
        - >  compteeditorpane.fxml
        - >  comptesmanagement.fxml
        - >  dailybankmainframe.fxml
        - >  employeeditorpane.fxml
        - >  employesmanagement.fxml
        - >  exceptiondialog.fxml
        - >  logindialog.fxml
        - >  operationeditorpane.fxml
        - >  operationsmanagement.fxml
      - >  application.css
  -  src/test/java
  -  src/test/resources
  - >  JRE System Library [JavaSE-17]
  - >  Maven Dependencies
  - >  basededonnee
  - >  src
  - >  target
  - >  dependency-reduced-pom.xml
  - >  pom.xml

### 3) Fonctionnalités

#### a) Version 1

##### i) Créer des employés (A. B.)

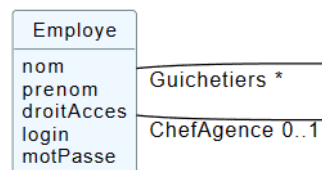
Use case de la fonctionnalité :



Classes concernées :

application.control.EmployesManagement ( nouveauEmploye)  
 application.view.EmployesManagementController  
 (doNouveauEmploye )  
 model.orm.Acces\_BD\_Employe ( insertEmploye)

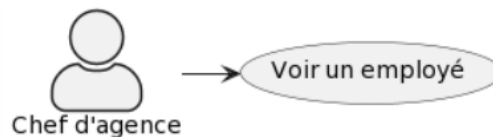
Partie du diagramme de classes concernée :



Lorsque le chef d'agence ajoute un nouvel employé, il est ajouté dans la base de données et dans la liste des employés. Pour plus de détails, voir doc utilisateur.

##### ii) Voir les employés (A. B.)

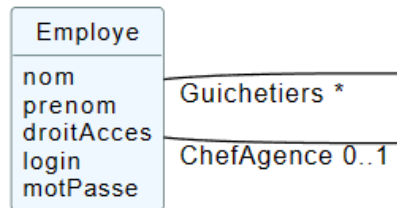
Use case de la fonctionnalité :



Classes concernées :

application.control.EmployesManagement (getlisteEmployes)  
 application.view.EmployesManagementController (doRechercher )  
 model.orm.Acces\_BD\_Employe ( getEmploye)

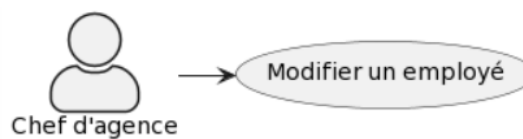
Partie du diagramme de classes concernée :



Lorsque le chef d'agence utilise la fonctionnalité rechercher, la liste des employés est affichée. Pour plus de détails, voir doc utilisateur.

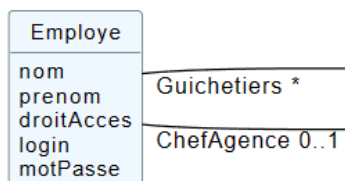
### iii) Modifier les employés (H. B.)

Use case de la fonctionnalité :



Classes concernées : `application.control.EmployeesManagement ( modifierEmploye )`  
`application.view.EmployeesManagementController`  
`(doModifierEmploye )`  
`model.orm.Acces_BD_Employe ( updateEmploye )`

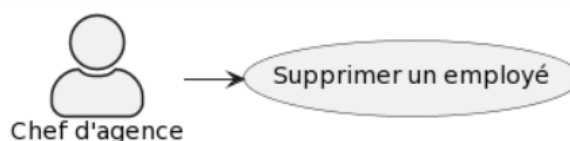
Partie du diagramme de classes concernée :



Lorsque le chef d'agence modifie un employé, les champs modifiés sont changés et les autres sont laissés tels quels. L'identifiant unique de l'employé ne peut cependant pas être modifié pour éviter d'avoir des employés en double. Pour plus de détails, voir doc utilisateur.

### iv) Effacer les employés (H. B.)

Use case de la fonctionnalité :



Classes concernées :

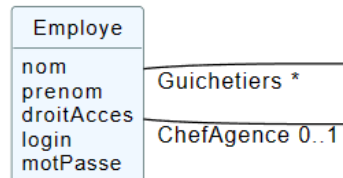


```

application.control.EmployeesManagement { supprimerEmploye }
application.view.EmployeesManagementController {
doSupprimerEmploye }
model.orm.Acces_BD_Employe { removeEmploye }

```

Partie du diagramme de classes concernée :



Lorsque le chef d'agence supprime un employé, celui-ci est supprimé de la base de données ainsi que de la liste des employés. Pour plus de détails, voir doc utilisateur.

### v) Créer un compte (B. S.)

Use case de la fonctionnalité :

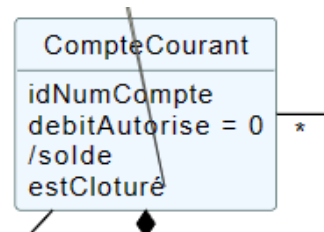


Classes concernées:

```

application.control.ComptesManagement(creeerNouveauCompte)
model.orm.Access_BD_CompteCourant(insertCompteCourrant)
application.view.ComptesManagementController(doNouveauCompte
)

```



Partie du diagramme de classes concernée :

Lorsque le guichetier crée un nouveau compte, il est ajouté dans la base de données et le débit autorisé est de 0 par défaut. Pour plus de détails, voir doc utilisateur.

### vi) Créditer / Débitier sur un compte (B. S.)

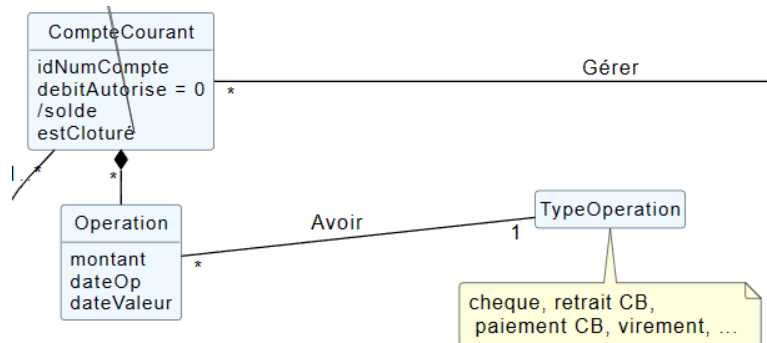
Use case de la fonctionnalité :



Classes concernées:

application.view.OperationsManagementController (doCredit )  
 model.orm.Access\_BD\_Operation (insertCredit)  
 application.control.OperationsManagement (enregistrerCredit)  
 application.view.OperationEditorPaneController (doAjouter -> case  
 crédit , displayDialog -> case crédit)

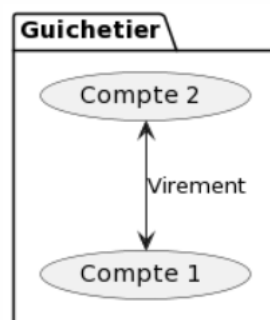
Partie du diagramme de classes concernée :



Lorsque un guichetier effectue un crédit ou un débit sur un compte, le solde de ce compte est modifié, la base de données est modifiée en conséquence. Pour plus de détails, voir doc utilisateur.

## vii) Effectuer un virement (J. M.)

Use case de la fonctionnalité :



Classes concernées :

### viii) Clôturer un compte (J. M.)

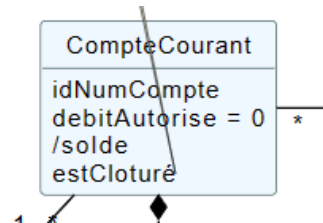
Use case de la fonctionnalité :



Classes concernées :

model.orm.Access\_BD\_CompteCourant (cloturerCompteCourant)  
 application.view.ComptesManagementController(doSupprimerCompte)  
 application.view.ComptesManagementController(validateComponentState)  
 application.view.OperationManagementController(validateComponentState)

Partie du diagramme de classes concernées :



Lorsque le guichetier clôture un compte, on ne peut plus effectuer d'opérations sur ce compte mais il reste dans la base de données. Pour plus de détails, voir doc utilisateur.

### b) Version 2 (en cours de développement)

- i) Débit exceptionnel
- ii) Simuler emprunt
- iii) ...