Corrigé des Ateliers Architecture distribuée en JEE

Mohammed Lahmer
m.lahmer@umi.ac.ma

Table de matière

e-Banking (RMI/JDBC) **3**

Solution e-Bancking **4**

Front en JavaFX 4

Backend en JavaFX 10

SQL Editor (Servlet/JSP/Bean) **15**

Solution SQL Editor **16**

E-contact (Servlet/JSP/Beans) **22**

Solution **23**

Gestion des Notes d'information (MVC-2) **28**

Solution **29**

Back-end 29

Front-end 44

XmlToUml JSF/XML **52**

Solution **53**

Atelier JSF /EJB /JPA **60**

Solution Projet Ejb 61

Solution Projet JSF 64

Jackarta/OpenLiberty **68**

Projet Authentification JSF 68

Projet Web Service REST/Persistance 71

e-Banking (RMI/JDBC)

Objectifs :

Maîtriser les principales méthodes de l'API JDBC et RMI et concrétiser le principe du modèle MVC(Modèle, Vue Contrôle) dans une architecture Client/Serveur

Partie Front-end

Dans ce TP vous serez ramené à réaliser une application graphique en javaFX de gestion des comptes d'une banque. Un compte bancaire est composé de numCompte, Nom, Prénom, Cin et le Solde.

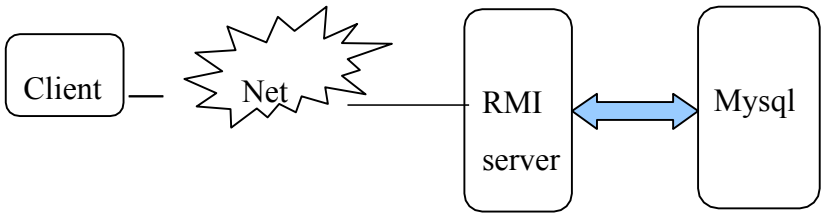
L'application doit permettre à l'utilisateur de :

S'authentifier (login, password)

- Dépôt d'argent
- Retrait d'argent
- Transfert vers un autre compte
- Releve

Partie Back-end

On vous propose maintenant de déployer l'application en mode client serveur en utilisant l'architecture RMI.



1- Identifie les services de l'interface distante et implémenter les dans l'objet distant adéquat.2- Réaliser le client RMI

3- Pour améliorer les performances de l'application, vous allez implémenter un pool de connexions de taille max 4. Chaque utilisateur connecté se verra attribué une connexion, une fois traité la connexion associée est libérée. La fermeture effective des connexions est faite à l'arrêt du serveur RMI.

Solution e-Bancking

Front en JavaFX

pom.xml

```
<dependencies>
        <dependency>
            <groupId>org.openjfx</groupId>
            <artifactId>javafx-fxml</artifactId>
            <version>11</version>
        </dependency>
        <dependency>
            <groupId>org.openjfx</groupId>
            <artifactId>javafx-media</artifactId>
            <version>11</version>
        </dependency>
    </dependencies>
```

src/ressources

HomeScreen.fxml

```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.geometry.Insets?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.TableColumn?>
<?import javafx.scene.control.TableView?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.effect.Glow?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.Pane?>
<?import javafx.scene.text.Font?>

<AnchorPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity"
minWidth="-Infinity" prefHeight="537.0" prefWidth="1040.0" style="-fx-
background-color: FFFFFF;" xmlns="http://javafx.com/javafx/8.0.171"
xmlns:fx="http://javafx.com/fxml/1"
fx:controller="estm.dsic.Controller.HomeController">
   <children>
      <TableView fx:id="tbv_Operations" layoutX="48.0" layoutY="279.0"
prefHeight="229.0" prefWidth="632.0">
        <columns>
          <TableColumn fx:id="ID" prefWidth="75.0" text="Id Transaction" />
            <TableColumn fx:id="typeOp" prefWidth="75.0" text="Operation's Type
" />
            <TableColumn fx:id="amount" prefWidth="75.0" text="Amount" />
            <TableColumn fx:id="Date" prefWidth="75.0" text="Date" />
        </columns>
         <columnResizePolicy>
            <TableView fx:constant="CONSTRAINED_RESIZE_POLICY" />
         </columnResizePolicy>
      </TableView>
      <Pane depthTest="DISABLE" layoutX="48.0" layoutY="43.0" prefHeight="200.0"
prefWidth="446.0" style="-fx-background-color: EAF5F5;">
         <children>
            <Label layoutX="24.0" layoutY="35.0" text="Account Owner" />
            <Label layoutX="24.0" layoutY="72.0" text="Account Number" />
            <Label layoutX="24.0" layoutY="111.0" text="Email" />
            <Label fx:id="lbl_owner" layoutX="198.0" layoutY="35.0" text="Label"
/>
            <Label fx:id="lbl_AccNum" layoutX="198.0" layoutY="72.0"
text="Label" />
```

```
            <Label fx:id="lbl_email" layoutX="198.0" layoutY="111.0"
text="Label" />
         </children>
         <effect>
            <Glow />
         </effect>
      </Pane>
      <Pane layoutX="589.0" layoutY="43.0" prefHeight="200.0" prefWidth="200.0"
style="-fx-background-color: EAF5F5;">
         <children>
            <Label layoutX="52.0" layoutY="46.0" text="Actual Balance" />
            <Label fx:id="lbl_balance" layoutX="66.0" layoutY="108.0" style="-
fx-background-color: #ffbf00;" text="Label">
               <font>
                  <Font name="System Bold" size="22.0" />
               </font>
               <padding>
                  <Insets bottom="5.0" left="5.0" right="5.0" top="5.0" />
               </padding></Label>
         </children>
      </Pane>
      <Button fx:id="btn_Deposit" layoutX="848.0" layoutY="368.0"
mnemonicParsing="false" onAction="#btn_DepositClicked" text="Deposit" />
      <TextField fx:id="txt_Amount" layoutX="797.0" layoutY="324.0" />
      <Button fx:id="btn_Withdraw" layoutX="843.0" layoutY="405.0"
mnemonicParsing="false" onAction="#btn_WithdrawClicked" text="Withdraw" />
      <Button fx:id="btn_logout" layoutX="946.0" layoutY="495.0"
mnemonicParsing="false" onAction="#btn_logoutClicked" text="Logout" />
   </children>
</AnchorPane>
```

LoginScreen.fxml

```
<?xml version="1.0" encoding="UTF-8"?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<?import javafx.scene.control.PasswordField?>
<?import javafx.scene.control.TextField?>
<?import javafx.scene.layout.AnchorPane?>
<?import javafx.scene.layout.Pane?>
<?import javafx.scene.text.Font?>

<AnchorPane id="AnchorPane" prefHeight="450.0" prefWidth="630.0" style="-fx-
background-color: #3333;" xmlns="http://javafx.com/javafx/8.0.171"
xmlns:fx="http://javafx.com/fxml/1"
fx:controller="estm.dsic.Controller.LoginController">
   <children>
      <Pane layoutX="172.0" layoutY="55.0" prefHeight="339.0" prefWidth="283.0"
style="-fx-background-radius: 12px; -fx-background-color: #ffff;">
         <children>
            <TextField fx:id="txtEmail" layoutX="56.0" layoutY="130.0" />
            <PasswordField fx:id="txtPass" layoutX="56.0" layoutY="195.0" />
            <Button fx:id="btnLogin" layoutX="89.0" layoutY="244.0"
mnemonicParsing="false" onAction="#btnLoginClick" style="-fx-background-color:
lightgreen;" text="Se Connecter" />
            <Label layoutX="58.0" layoutY="176.0" text="Password" />
            <Label layoutX="58.0" layoutY="106.0" text="Email" />
            <Label layoutX="64.0" layoutY="44.0" text="Authentification"
underline="true">
               <font>
                  <Font name="System Bold" size="17.0" />
               </font>
            </Label>
         </children>
      </Pane>
   </children>
</AnchorPane>
```

src/java : package estm.dsic.Models

Account.java

```
package estm.dsic.Models;

import java.io.Serializable;
import java.util.ArrayList;

public class Account implements Serializable {
    private int accountNumber;
    private int accountOwner;
    private double balance;
    private Client client;
    private ArrayList<Operation> lstOps;

    public ArrayList<Operation> getLstOps() {
        return lstOps;
    }

    public void setLstOps(ArrayList<Operation> lstOps) {
        this.lstOps = lstOps;
    }

    public Client getClient() {
        return client;
    }

    public void setClient(Client client) {
        this.client = client;
    }

    public Account(int accountNumber, int accountOwner, double balance, Client
Clt) {
        this.accountNumber = accountNumber;
        this.accountOwner = accountOwner;
        this.balance = balance;
        this.client = Clt;
    }

    public Account(int accountNumber, int accountOwner, double balance,
ArrayList<Operation> lstOps) {
        this.accountNumber = accountNumber;
        this.accountOwner = accountOwner;
        this.balance = balance;
        this.lstOps = lstOps;
    }

    public Account(int accountNumber, int accountOwner, double balance) {
        this.accountNumber = accountNumber;
        this.accountOwner = accountOwner;
        this.balance = balance;
    }

    public int getAccountNumber() {
        return accountNumber;
    }

    public void setAccountNumber(int accountNumber) {
        this.accountNumber = accountNumber;
    }

    public int getAccountOwner() {
        return accountOwner;
    }

    public void setAccountOwner(int accountOwner) {
```

```
        this.accountOwner = accountOwner;
    }

    public double getBalance() {
        return balance;
    }

    public void setBalance(double balance) {
        this.balance = balance;
    }

}
```

src/java : package estm.dsic.Controllers

HomeController.java

```
package estm.dsic.Controllers;
import java.net.URL;
import java.rmi.RemoteException;
import java.util.ArrayList;
import java.util.ResourceBundle;
import estm.dsic.MainApp;
import estm.dsic.Models.Account;
import estm.dsic.Models.Client;
import estm.dsic.Models.Operation;
import javafx.application.Platform;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Alert;
import javafx.scene.control.Button;
import javafx.scene.control.ButtonType;
import javafx.scene.control.Label;
import javafx.scene.control.TableColumn;
import javafx.scene.control.TableView;
import javafx.scene.control.TextField;
import javafx.scene.control.Alert.AlertType;
import javafx.scene.control.cell.PropertyValueFactory;

public class HomeController implements Initializable {
    @FXML
    Label lbl_email;
    @FXML
    Label lbl_owner;
    @FXML
    Label lbl_AccNum;
    @FXML
    Label lbl_balance;
    @FXML
    Button btn_Deposit;
    @FXML
    Button btn_logout;
    @FXML
    Button btn_Withdraw;
    @FXML
    TextField txt_Amount;
    @FXML
    TableView<Operation> tbv_Operations;

    @FXML
    private TableColumn<Operation, Integer> ID;

    @FXML
    private TableColumn<Operation, String> typeOp;
```

```
    @FXML
    private TableColumn<Operation, Double> amount;

    @FXML
    private TableColumn<Operation, String> Date;

    Client currentClient = NavigationController.currentClient;

    @Override
    public void initialize(URL location, ResourceBundle resources) {


lbl_AccNum.setText(String.valueOf(currentClient.getAccount().getAccountNumber())
);

lbl_balance.setText(String.valueOf(currentClient.getAccount().getBalance()));
        lbl_email.setText(currentClient.getEmail());
        lbl_owner.setText(currentClient.getFirstName().toUpperCase() + " " +
                currentClient.getLastName());

        ID.setCellValueFactory(new PropertyValueFactory<Operation,
Integer>("id"));
        typeOp.setCellValueFactory(new PropertyValueFactory<Operation,
String>("typeOp"));
        amount.setCellValueFactory(new PropertyValueFactory<Operation,
Double>("amount"));
        Date.setCellValueFactory(new PropertyValueFactory<Operation,
String>("dateOpe"));

        RefreshTableView(currentClient.getAccount().getLstOps());

    }

    public void btn_DepositClicked(ActionEvent event) {
        try {
            double amount = Double.parseDouble(txt_Amount.getText());
            Account acc =
MainApp.objAccount.Deposit(currentClient.getAccount().getAccountNumber(),
amount);
            if (acc != null) {
                showSuccessAlert(AlertType.INFORMATION, "Success");
                currentClient.setAccount(acc);
                RefreshTableView(acc.getLstOps());

lbl_balance.setText(String.valueOf(currentClient.getAccount().getBalance()));
            } else {
                showSuccessAlert(AlertType.WARNING, "Something went wroong");

            }

        } catch (RemoteException e) {
            e.printStackTrace();
        }
    }

    public void btn_WithdrawClicked(ActionEvent event) {
        try {
            double amount = Double.parseDouble(txt_Amount.getText());
            Account acc =
MainApp.objAccount.Withdrawal(currentClient.getAccount().getAccountNumber(),
amount);
            if (acc != null) {
                showSuccessAlert(AlertType.INFORMATION, "Success");
                currentClient.setAccount(acc);
                RefreshTableView(acc.getLstOps());
```

```
lbl_balance.setText(String.valueOf(currentClient.getAccount().getBalance()));
            } else {
                showSuccessAlert(AlertType.WARNING, "Something went wroong");
            }

        } catch (RemoteException e) {
            e.printStackTrace();
        }
    }

    public void btn_logoutClicked(ActionEvent event) {
        try {
            currentClient = null;
            NavigationController.currentClient = null;
            NavigationController.navigateTo("LoginScreen", btn_Deposit, 630,
450);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

    private void RefreshTableView(ArrayList<Operation> lst) {
        if (lst != null) {
            ObservableList data = FXCollections.observableArrayList(lst);
            tbv_Operations.setItems(data);
        }
    }

    public void showSuccessAlert(AlertType alertType, String content) {
        Platform.runLater(() -> {
            Alert a = new Alert(alertType, "", ButtonType.OK);
            a.setTitle("");
            a.setHeaderText(content);
            a.show();

        });
    }
}
```

src/java : package estm.dsic.Controllers

LoginController.java

```
package estm.dsic.Controller;

import java.net.URL;
import java.util.ResourceBundle;

import estm.dsic.MainApp;
import estm.dsic.Models.Client;
import javafx.event.ActionEvent;
import javafx.fxml.FXML;
import javafx.fxml.Initializable;
import javafx.scene.control.Button;
import javafx.scene.control.PasswordField;
import javafx.scene.control.TextField;

public class LoginController implements Initializable {
    @FXML
    TextField txtEmail;
    @FXML
    PasswordField txtPass;
    @FXML
    Button btnLogin;
```

```
    public void btnLoginClick(ActionEvent event) {
        try {

            Client cl = MainApp.objClient.login(txtEmail.getText(),
txtPass.getText());
            if (cl != null) {
                NavigationController.currentClient = cl;
                NavigationController.navigateTo("HomeScreen",
btnLogin,1050,560);
            } else {
                System.out.println("bad email/password");
            }
        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }

    @Override
    public void initialize(URL arg0, ResourceBundle arg1) {

    }
}
```

src/java : package estm.dsic.Controllers

IAccount.java

```
package estm.dsic.Business;
import java.rmi.Remote;
import java.rmi.RemoteException;
import estm.dsic.Models.Account;
public interface IAccount extends Remote {
    public Account Deposit(int accNum, double amount) throws RemoteException;
    public Account Withdrawal(int accNum, double amount) throws RemoteException;

}
```

IClient.java

```
import java.rmi.Remote;
import java.rmi.RemoteException;
import estm.dsic.Models.Client;
public interface IClient extends Remote {

    public Client login(String email, String password) throws RemoteException;
}
```

Backend en JavaFX

pom.xml

```
      <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>8.0.32</version>
        </dependency>
```

src/java : package estm.dsic.Models

Operation.java

```
package estm.dsic.Models;

import java.io.Serializable;

public class Operation implements Serializable {
    private int id;
    private String typeOp;
    private double amount;
    private String dateOpe;
    private int accNumber;
    public String getDateOpe() {
        return dateOpe;
    }
    public void setDateOpe(String dateOpe) {
        this.dateOpe = dateOpe;
    }
    public double getAmount() {
        return amount;
    }
    public void setAmount(double amount) {
        this.amount = amount;
    }
    public Operation(int id, String typeOp, double amount, String dateOpe, int
accNumber) {
        this.id = id;
        this.typeOp = typeOp;
        this.amount = amount;
        this.accNumber = accNumber;
        this.dateOpe = dateOpe;
    }public Operation(String typeOp, double amount, String dateOpe, int
accNumber) {
        this.typeOp = typeOp;
        this.amount = amount;
        this.accNumber = accNumber;
        this.dateOpe = dateOpe;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public String getTypeOp() {
        return typeOp;
    }
    public void setTypeOp(String typeOp) {
        this.typeOp = typeOp;
    }
    public int getAccNumber() {
        return accNumber;
    }
    public void setAccNumber(int accNumber) {
        this.accNumber = accNumber;
    }
}
```

src/java : package estm.dsic.Dal

ClientDAO.java

```
package estm.dsic.dal;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
```

```
import java.sql.Statement;
import estm.dsic.Models.Client;
public class ClientDAO {

    public Client getClient(String email) {
        Connection con = Database.getConnection();
        Client client = null;
        try {
            String qry = "Select * from Holder where email='" + email + "'";
            Statement stmt = con.createStatement();
            ResultSet res = stmt.executeQuery(qry);
            if (res.isBeforeFirst() || res.getRow() != 0) {
                while (res.next()) {
                    client = new Client(
                            res.getInt("ID"),
                            res.getString("NOM"),
                            res.getString("PRENOM"),
                            res.getString("EMAIL"),
                            res.getString("MDP"),
                            new AccountDao().getAccount(res.getInt("ID"), -1));
                }
                return client;
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return null;
    }
}
```

ClientDAO.java

```
package estm.dsic.DataAccessLayer;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;

import estm.dsic.Models.Operation;

public class OperationDAO {
    public ArrayList<Operation> getOperations(int accNum) {
        Connection con = Database.getConnection();
        ArrayList<Operation> lstOps = new ArrayList<>();
        ;
        try {
            String qry = "Select * from Operation where Account_Num=" + accNum;
            Statement stmt = con.createStatement();
            ResultSet res = stmt.executeQuery(qry);
            if (res.isBeforeFirst() || res.getRow() != 0) {
                while (res.next()) {
                    lstOps.add(new Operation(
                            res.getInt("ID"),
                            res.getString("Type_Op"),
                            res.getDouble("Amount"),
                            res.getString("DateOp"),
                            res.getInt("Account_Num")));
                }
                return lstOps;
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return null;
```

```
    }

    public int addOperation(Operation op) {
        Connection con = Database.getConnection();
        try {
            String qry = "INSERT INTO Operation( Type_Op, Amount, DateOp,
Account_Num) VALUES ('" + op.getTypeOp()
                    + "','" + op.getAmount() + "','" + op.getDateOpe() + "','" +
op.getAccNumber() + "')";
            Statement stmt = con.createStatement();
            return stmt.executeUpdate(qry);

        } catch (SQLException e) {
            e.printStackTrace();
        }
        return 0;
    }
}
```

src/java : package estm.dsic.Business

ClientBusiness.java

```
package estm.dsic.Business;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;

import estm.dsic.DataAccessLayer.ClientDAO;
import estm.dsic.Models.Client;

public class ClientBusiness extends UnicastRemoteObject implements IClient {

    public ClientBusiness() throws RemoteException {
    }

    @Override
    public Client login(String email, String password) throws RemoteException {
        ClientDAO clDao = new ClientDAO();
        Client cl = clDao.getClient(email);
        if (cl != null && cl.getPassword().equals(password)) {
            return cl;
        }
        return null;
    }

}
```

ClientBusiness.java

```
package estm.dsic.Business;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.text.SimpleDateFormat;
import java.util.Date;
import estm.dsic.DataAccessLayer.AccountDao;
import estm.dsic.DataAccessLayer.OperationDAO;
import estm.dsic.Models.Account;
import estm.dsic.Models.Operation;

public class AccountBusiness extends UnicastRemoteObject implements IAccount {
    public AccountBusiness() throws RemoteException {
    }
    SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd hh:mm:ss");
    @Override
    public Account Deposit(int accNum, double amount) throws RemoteException {
        Operation op = new Operation("DEPOSIT", amount, sdf.format(new Date()),
accNum);
        OperationDAO opDao = new OperationDAO();
        if (opDao.addOperation(op) != 0) {
```

```
            AccountDao accDao = new AccountDao();
            accDao.updateAccount(accNum, amount);
            return accDao.getAccount(-1, accNum);
        }
        return null;
    }
    @Override
    public Account Withdrawal(int accNum, double amount) throws RemoteException
{
        Operation op = new Operation("WITHDRAWAL", amount, sdf.format(new
Date()), accNum);
        OperationDAO opDao = new OperationDAO();
        if (opDao.addOperation(op) != 0) {
            AccountDao accDao = new AccountDao();
            accDao.updateAccount(accNum, (-amount));
            return accDao.getAccount(-1, accNum);
        }
        return null;
    }
}
```

src/java : package estm.dsic.Controllers

MainController.java

```
package estm.dsic.Controllers;
import java.rmi.Naming;
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import estm.dsic.Business.AccountBusiness;
import estm.dsic.Business.ClientBusiness;
public class MainController {
    public static void main(String[] args) {
        int port = 2023;
        try {
            Registry registry = LocateRegistry.createRegistry(port);
            ClientBusiness clBs = new ClientBusiness();
            String url = "rmi://localhost:" + port + "/login";
            Naming.rebind(url, clBs);

            AccountBusiness accBs = new AccountBusiness();
            String urlOperation = "rmi://localhost:" + port + "/operation";
            Naming.rebind(urlOperation, accBs);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

SQL Editor (Servlet/JSP/Bean)

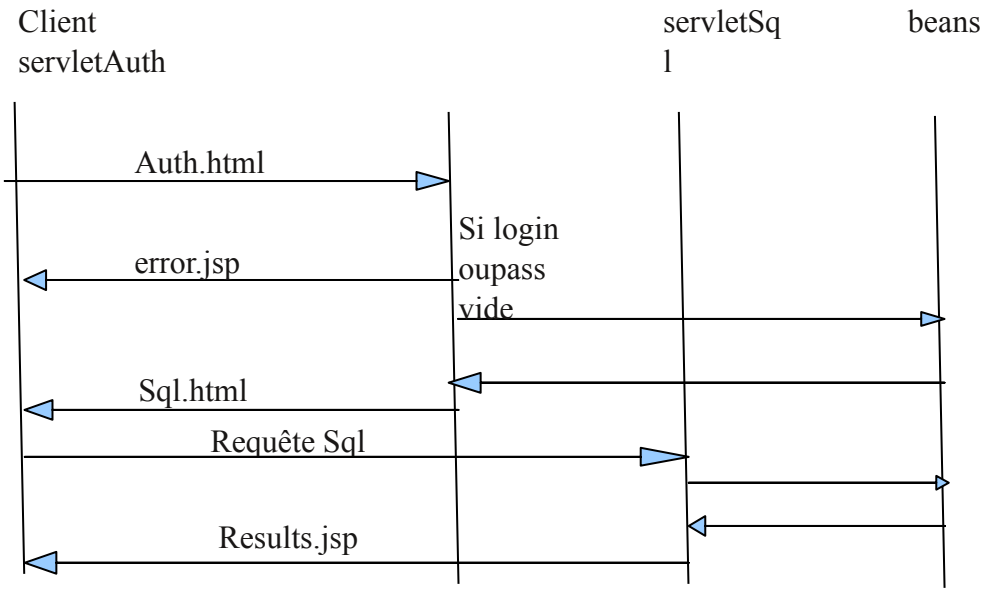
Objectifs

Maîtriser l'interaction jsp, servlet et bean dans une application web JEE respectant le modèle MVC-1.

Dans cet atelier vous serez ramené à réaliser une application web pour l'exploitation d'une base dedonnées distante. La vue est composée de

- une page auth.html d'authentification
- une page sql.html qui contient
  - une zone de texte pour la saisie des ordre SQL
  - deux boutons exécuter et Annuler
- une page results.jsp qui affiche le résultat sous forme de table html

Le scénario d'exécution de l'application peut être résumer dans le diagramme de séquence UML suivant :



La page error.jsp est affichée dans trois cas :

- login ou mot de passe vide (vous pouvez utiliser javascript c'est plus rapide)
- login ou mot de passe incorrecte par rapport à la base de données
- erreur dans la requête sql par rapport à la base

Pour réaliser une page réutilisable, je vous suggère d'utiliser un bean error.java avec deux propriétés :code_erreur (1 pour le cas 2 et 2 pour le cas 3) et message_erreur. Dans tous les cas d'erreurs la page doit donnée la possibilité de ressaisir le login et le mot de passe dans le cas 1 et 2et de retaper la requête dans le cas 3.

Afin d'éviter le va et viens vers la base de données on vous suggère de réaliser un petit analyseur syntaxique en java script qui permettra d'analyser les erreurs dans la requête sql avant de l'envoyer au serveur.

Solution SQL Editor

src/java : package estm.dsic.sqleditor.controller

EditorServlet.java

```
package estm.dsic.sqleditor.controller;
import estm.dsic.sqleditor.dao.LoggedOutUserException;
import estm.dsic.sqleditor.model.Error;
import estm.dsic.sqleditor.service.EditorService;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.sql.SQLException;

@WebServlet(name = "SqlEditorServlet", urlPatterns = {"/editor", "/editor.jsp"})
public class EditorServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        if (req.getSession().getAttribute("user") == null)
resp.sendRedirect("auth");
        req.getRequestDispatcher("WEB-INF/sql.page.jsp").forward(req, resp);
    }

    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        try {
            String query = req.getParameter("query");
            if (query == null || req.getAttribute("result") != null) doGet(req,
resp);
            else {
                String result = EditorService.executeQuery(query);

                req.setAttribute("result", result);
                req.getRequestDispatcher("editor").forward(req, resp);
            }
        } catch (SQLException ex) {
            Error.ErrorCode code = Error.ErrorCode.SQL_ERROR;
            if (ex instanceof LoggedOutUserException) {
                req.getSession().removeAttribute("user");
                code = Error.ErrorCode.AUTH_ERROR;
            }
            req.setAttribute("error", Error.fromException(code, ex));
            req.getRequestDispatcher("error.page.jsp").forward(req, resp);
        }
    }
}
```

AuthServlet.java

```
package estm.dsic.sqleditor.controller;

@WebServlet(urlPatterns = {"/auth", "/auth.jsp"})
public class AuthServlet extends HttpServlet {
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        if (req.getSession().getAttribute("user") != null)
resp.sendRedirect("editor");
        else req.getRequestDispatcher("WEB-INF/auth.page.jsp").forward(req,
resp);
    }
```

```
    @Override
    protected void doPost(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        String username = req.getParameter("username");
        String password = req.getParameter("password");
        User user = new User(username, password);
        try {
            AuthService.authenticate(user);

            req.getSession().setAttribute("user", user);
            resp.sendRedirect("editor");
        } catch (SQLException ex) {
            req.setAttribute("error",
Error.fromException(Error.ErrorCode.AUTH_ERROR, ex));
            req.getRequestDispatcher("error.page.jsp").forward(req, resp);
        }
    }
}
```

src/java : package estm.dsic.sqleditor.dao

Connection.java

```
package estm.dsic.sqleditor.dao;

import estm.dsic.sqleditor.model.User;

import javax.naming.InitialContext;
import javax.naming.NamingException;
import javax.sql.DataSource;
import java.sql.SQLException;

public class Connection {
    protected static DataSource dataSource;
    protected JBossConnection() {
        if (dataSource == null) {
            try { dataSource = InitialContext.doLookup("java:/
jspOracleAdmin"); }
            catch (NamingException ignored) { }
        }
    }
    @Override
    protected boolean open(User user) throws SQLException {
        if (connection != null && !connection.isClosed()) {
            try {
                connection.close();
            } catch (SQLException e) {
                connection = null;
            }
        }
        try {
            connection = dataSource.getConnection(user.getUsername(),
user.getPassword());
        } catch (Exception ex) {
            throw new SQLException("Nom d'utilisateur ou mot de passe
incorrect");
        }

        return connection != null;
    }

    @Override
    protected void close() {
        try { if (connection != null && !connection.isClosed())
connection.close(); }
```

```
        catch (SQLException ignored) { }
        finally { connection = null; }
    }
}
```

QueryDao.java

```
package estm.dsic.sqleditor.dao;
import estm.dsic.sqleditor.model.Query;
import java.sql.ResultSet;
import java.sql.ResultSetMetaData;
import java.sql.SQLException;
import java.sql.Statement;
import static estm.dsic.sqleditor.model.Query.QueryType.LDD;
import static estm.dsic.sqleditor.model.Query.QueryType.LMD_WITH_RESULT;

public final class QueryDao implements IQueryDao {
    @Override
    public String execute(Query query) throws SQLException {
        Statement stm = Connection.getConnection().createStatement();
        StringBuffer result = new StringBuffer();

        if (query.getType() == LMD_WITH_RESULT) {
            ResultSet resultSet = stm.executeQuery(query.getQuery());

            if (!resultSet.next()) {
                result.append("<div class='w3-container pale-blue'><h1><i
class='ic-success'></i>No result founded</h1></div>");
            } else {
                ResultSetMetaData metaData = resultSet.getMetaData();
                int colCount = metaData.getColumnCount();
                int i;

                // table header
                result.append("<thead><tr>");
                for (i = 1; i <= colCount; i++)

result.append("<th>").append(metaData.getColumnLabel(i)).append("</th>");
                result.append("</tr></thead>");

                // table data
                result.append("<tbody>");
                do {
                    result.append("<tr>");
                    for (i = 1; i <= colCount; i++)

result.append("<td>").append(resultSet.getObject(i)).append("</td>");
                    result.append("</tr>");
                } while (resultSet.next());
                result.append("</tbody>");

                // wrap in table
                result.insert(0, "<table class='w3-table-all'>");
                result.append("</table>");
            }
            resultSet.close(); // prevent open cursors limit exception
        } else {
            int count = stm.executeUpdate(query.getQuery());
            String cls = "pale-";

            if (query.getType() == LDD) {
                result.append("L'opération a réussi");
                cls += "green";
            } else {
                result.append(count).append(" ligne(s) affectée(s)");
                cls += "blue";
```

```
            }

            result.insert(0, "<div class='w3-container " + cls + "'><h1><i
class='ic-success'></i>");
            result.append("</h1></div>");
        }

        stm.close();
        return result.toString();
    }
}
```

src/java : package estm.dsic.sqleditor.service

EditorService.java

```
package estm.dsic.sqleditor.service;
import estm.dsic.sqleditor.dao.IQueryDao;
import estm.dsic.sqleditor.dao.QueryDao;
import estm.dsic.sqleditor.model.Query;
import java.sql.SQLException;

public final class EditorService {
    private static final IQueryDao queryDao = new QueryDao();

    public static String executeQuery(String query) throws SQLException {
        return queryDao.execute(new Query(query));
    }
} }
```

AuthService.java

```
package estm.dsic.sqleditor.service;
import estm.dsic.sqleditor.dao.IUserDao;
import estm.dsic.sqleditor.dao.UserDao;
import estm.dsic.sqleditor.model.User;

import java.sql.SQLException;

public final class AuthService {
    private final static IUserDao userDao = new UserDao();

    public static void authenticate(User user) throws SQLException {
        userDao.login(user);
    }

    public static void logout() throws SQLException {
        userDao.logout();
    }
}
```

Web: sql.jsp

```
<%@ page import="estm.dsic.sqleditor.service.AuthService" %>
<%@ page import="java.sql.SQLException" %>

<%
    if (request.getParameter("logout") != null) {
        out.println("Ok");
        try {
            AuthService.logout();
        } catch (SQLException ignored) {
        }
        request.getSession().removeAttribute("user");
        response.sendRedirect(request.getContextPath() + "/auth");
        return;
    }
```

```
%>
<%@ page contentType="text/html;charset=UTF-8" %>
<html>
<head>
    <link rel="stylesheet" href="${pageContext.request.contextPath}/assets/
w3.css">
    <link rel="stylesheet" href="${pageContext.request.contextPath}/assets/
atelier.css">
    <link rel="stylesheet" href="${pageContext.request.contextPath}/assets/my-
icons/myicons.css">
    <style>
        body {
            background-image: url(${pageContext.request.contextPath}/assets/
Pattern-Randomized.svg);
            background-attachment: fixed;
            background-size: cover;
            min-height: 100vh;
        }

        header form {
            height: 100%;
            display: flex !important;
            align-items: center !important;
            justify-content: center !important;
        }
    </style>
    <title>SQL Editor</title>
</head>
<body>

<header class="w3-container blue w3-display-container w3-border-bottom">
    <h1 class="w3-center">SQL Editor</h1>
    <form class="w3-display-topright w3-btn" onclick="this.submit()">
        <i class="ic-disconnect"></i>
        <span>Déconnecter</span>
        <input name="logout" value="a" hidden>
    </form>
</header>
<main class="w3-panel">
    <form onsubmit="checkQuery(event)"
          class="w3-container white w3-content w3-padding w3-padding-16 w3-
card-4 w3-border"
          action="editor" method="POST">
        <label><i class="ic-edit"></i>Requete</label>
        <textarea class="w3-input w3-border w3-margin-bottom" name="query"
                  placeholder="Entrer votre requete SQL ici." style="resize:
none"></textarea>
        <div class="w3-right w3-container w3-btn teal">
            <i class="ic-run"></i><input type="submit" class="w3-btn w3-hover-
none" value="Exécuter">
        </div>
    </form>
</main>

<%
    String result;
    if ((result = (String) request.getAttribute("result")) != null) {
%>
<div class="w3-content w3-panel w3-card-4 w3-animate-top" style="padding: 0;
width: 96vw">
    <header class="w3-container blue">
        <h1 class="w3-center">Execution Resultat</h1>
    </header>
    <%=result%>
</div>
<% } %>
```

```
<script>
    const tx = document.querySelector('textarea');
    tx.style.height = tx.scrollHeight + 'px';
    tx.style.overflowY = 'hidden';
    tx.addEventListener("input", () => {
        tx.style.height = 'auto';
        tx.style.height = (tx.scrollHeight) + 'px';
    }, false);

    const regExCreate = /^CREATE\s+TABLE\s+('?)[^'\d\s]+\w*\1\s*\((\s*[^'\d\s]+
\w*\s+(((VARCHAR2?|INTEGER|CHAR)(\(\d+\))?)|(DATE|DATETIME)|(NUMBER(\(\d+(,\d+)?
\))?))(\s+DEFAULT\s+('?)\w+\12)?(\s+(NOT)?\s+NULL)?(\s+PRIMARY KEY)?\s*(,(?!$)|
\);?$))+/gim;
    const regExSelectDelete = /^(SELECT\s+(\*|([A-Z_]+\w*\s*(,|\s+)?)+)|DELETE)
\s+FROM\s+[A-Z_]+\w*(\s*;?$|\s+WHERE(\s+[A-Z_]+\w*\s*(=|<[>=]?|>=?|\s+LIKE\s+)
\s*[A-Z_]+\w*(\s+(AND|OR)(?!$)|\s*;?$))+)/gim;
    const regExUpdate = /^UPDATE\s+[A-Z_]\w*\s+SET\s+(\s*[A-Z_]\w*\s*=\s*\w+
\s*(,(?!$)|\s*(?=WHERE|.$)))+(WHERE(\s+[A-Z_]+\w*\s*(=|<[>=]?|>=?|\s+LIKE\s+)
\s*\w+(\s+(AND|OR)(?!$)|\s*;?$))+)?/gim;

    function checkQuery(event) {
        const v = tx.value;
        if (v.match(regExCreate) || v.match(regExSelectDelete) ||
v.match(regExUpdate)) {
            tx.value = tx.value.replaceAll(";", "");
            return true;
        } else {
            const p = confirm("Cette requete peut generer une erreur. Voulez-
vous executer cette requete ?");
            if (!p) event.preventDefault();
            return p;
        }
    }
</script>
</body>
</html>
```

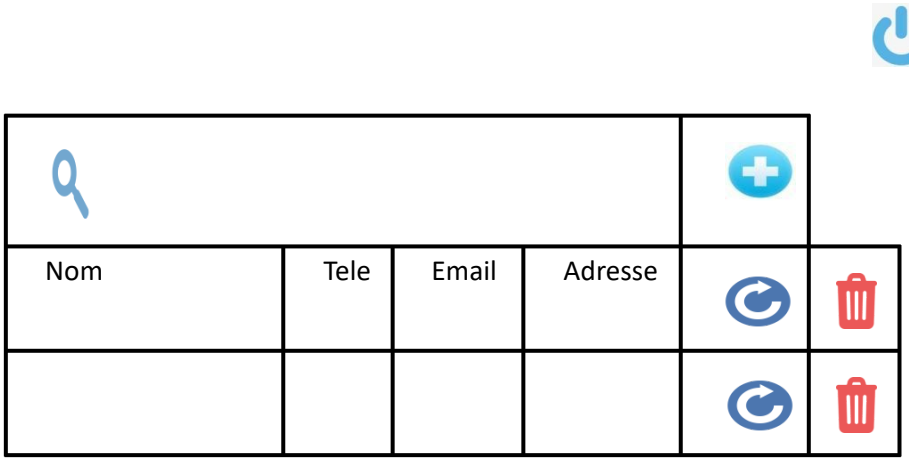
E-contact (Servlet/JSP/Beans)

L'application consiste en un petit portail en JEE qui permettra à une entreprise de gérer ses contacts. Pour simplifier votre base de données sera composée d'une table User (id auto-increment, login, password) et d'une table contact est défini par (#Nom,Tele, email, Adresse, id_User)

L'inscription de l'entreprise dans le portail se fera à travers la page d'indexe qui n'est rien qu'une page d'authentification pour les entreprises déjà inscrite. Une fois authentifiée l'entreprise doit pouvoir gérer ses contacts par :

- L'Ajout
- La suppression
- La modification et
- La recherche des contacts

A travers une page gestionContacts dont le format est le suivant :



Service	Description
	Lors de l'ajout d'un contact, il faut vérifier le format Email et Tele et qu'il n'existe pas déjà dans la base de donnée
	Il est recommandé que la modification d'un contact se fait sur la même page
	Pour la suppression du contact sélectionné de la table
	La cherche multicritère peut être réalisée par ajax
	Pour se déconnecté il suffit d'invalider la session et de retourner à la page d'authentification (session.invalidate())

Solution

src/java : package dal

Connexion.java

```
package dal;
import java.sql.Connection;
import java.sql.DriverManager;
public class Connexion{
    private static Connection connection;
    static {
        try {
            Class.forName("com.mysql.cj.jdbc.Driver");
            connection = DriverManager.getConnection
                    ("jdbc:mysql://localhost:8889/contact","root","root");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public static Connection getConnection() {
        return connection;
    }
}
```

IContactDao.java

```
package dal;
import model.Contact;
import model.User;
import java.util.List;
public interface IContactDao {
    Contact update(Contact contact);
    Contact add(Contact contact);
    void delete(String nom);
    Contact serach(Integer id);
    List<Contact> getAllConatct();
}
```

ContactDaoImpl.java

```
package dal;
import model.Contact;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;
import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.ArrayList;
import java.util.List;
public class ContactDaoImpl implements IContactDao {
    @Override
    public Contact add(Contact contact) {
        Connection conn = ConnexionSingleton.getConnection ();
        try {
            PreparedStatement ps = conn.prepareStatement ("INSERT INTO CONTACT(NOM, TELE,
EMAIL, ADRESSE) VALUES (?, ?, ? ,?)");
            ps.setString(1, contact.getNom ());
            ps.setString (2, contact.getTelephone ());
            ps.setString (3, contact.getEmail ());
```

```
            ps.setString (4, contact.getAdresse ());
            ps.executeUpdate();
            ps.close ();
        } catch (SQLException e) {
            e.printStackTrace ();
        }
        return contact;
    }
    @Override
    public Contact update(Contact contact) {
        Connection conn = ConnexionSingleton.getConnection();
        try {
PreparedStatement ps = conn.prepareStatement("UPDATE CONTACT SET NOM=?, TELE=?, EMAIL=?,
ADRESSE=? WHERE NOM = ?");
            ps.setString (1, contact.getTelephone ());
            ps.setString (2, contact.getNom ());
            ps.setString (3, contact.getEmail ());
            ps.setString (4, contact.getAdresse ());
            ps.executeUpdate();
            ps.close();
        } catch (SQLException e){
            e.printStackTrace ();
        }
        return contact;
    }
    @Override
    public void delete(String nom) {
        Connection conn = ConnexionSingleton.getConnection();
        try {
            PreparedStatement ps= conn.prepareStatement("DELETE FROM CONTACT WHERE NOM
= ?");
            ps.setString (1, nom);
            ps.executeUpdate();
            ps.close();
        } catch (SQLException e) {
            e.printStackTrace();
        }
    }
    @Override
    public Contact serach(Integer id) {
        Connection con = ConnexionSingleton.getConnection ();
        Contact contact = new Contact ();
        try {
PreparedStatement ps = con.prepareStatement ("SELECT * FROM CONTACT WHERE ID=?");
            ps.setLong(1, id);
            ResultSet rs = ps.executeQuery();
            if (rs.next ()) {
                contact.setNom (rs.getString ("nom"));
                contact.setEmail (rs.getString ("email"));
                contact.setTelephone (rs.getString ("tele"));
            }
        } catch (SQLException e) {
            e.printStackTrace ();
        }
        return contact;
    }
    @Override
    public List<Contact> getAllConatct() {
        List<Contact> contacts = new ArrayList<> ();
        Connection con = ConnexionSingleton.getConnection ();
```

```
        try {
            PreparedStatement ps = con.prepareStatement ("SELECT * FROM CONTACT");
            // ps.setLong(1, id);
            ResultSet rs = ps.executeQuery();
            while (rs.next ()) {
                String nom = rs.getString ("nom");
                String email = rs.getString ("email");
                String adresse = rs.getString ("adresse");
                String tele = rs.getString ("tele");
                contacts.add (new Contact ( nom, tele, email, adresse));
            }
        } catch (SQLException e) {
            e.printStackTrace ();
        }
        return contacts;
    }
}
```

src/java : package model

Contact.java

```
package model;
public class Contact {
    private String nom;
    private String telephone;
    private String email;
    private String adresse;
    public Contact() {}
    public Contact(String nom, String telephone, String email, String adresse) {
        this.nom = nom;
        this.telephone = telephone;
        this.email = email;
        this.adresse = adresse;
    }
//getters setters omis
}
```

Partie Web

Index.jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<style>input[type=text], select {
  width: 100%;
  padding: 12px 20px;
  margin: 8px 0;
  display: inline-block;
  border: 1px solid #ccc;
  border-radius: 4px;
  box-sizing: border-box;
}
input[type=submit] {
  width: 100%;
  background-color: #0D9DD1;
  color: white;
  padding: 14px 20px;
  margin: 8px 0;
  border: none;
  border-radius: 4px;
  cursor: pointer;
}
```

```
input[type=submit]:hover {
  background-color: #45a049;
}

div {
  border-radius: 5px;
  background-color: #f2f2f2;
  padding: 20px;
}
</style>
  <head>
    <title>$TITLE$</title>
  </head>
  <body>
  <div>
    <form action="<%= request.getContextPath() %>/login" method="post">
      <label for="login">LOGIN</label>
      <input type="text" id="login" name="login" placeholder="login..">

      <label for="password">PASSWORD</label>
      <input type="text" id="password" name="password" placeholder="Password">
            <br>${message}
      <input type="submit" value="Se connecter">
    </form>
  </div>
  </body>
</html>
```

AddContact.jsp

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
<head>
    <title>CONTACT FORM</title>
</head>
<h3>CONTACT FORM</h3>
<div>
    <form action="<%= request.getContextPath() %>/ajouter" method="post">
        <label for="nom">NAME</label>
        <input type="text" id="nom" name="nom" placeholder="Nom">
        <label for="telephone">PHONE</label>
        <input type="text" id="telephone" name="telephone" placeholder="Télephone">
        <label for="email">EMAIL</label>
        <input type="text" id="email" name="email" placeholder="Email">
        <label for="adresse">ADRESS</label>
        <input type="text" id="adresse" name="adresse" placeholder="Adresse">
        <input type="submit" value="submit">
    </form>
</div>
</body>
</html>
```

Contact.jsp

```
<%@ page import="model.Contact" %>
<%@ page import="java.util.List" %>
<html>
<head>
<title>CONTACTS</title>
</head>
<body>
```

```
<h1 style="text-align: center">CONTACTS </h1>
<a href="index.jsp"><i style='font-size:44px; color:dodgerblue' class='fas'>&#xf011;</
i></a>
<table id="myTable">
    <tr >
        <th colspan="4" class="amine"><div class="searchContainer">
            <i class="fa fa-search searchIcon" onclick="myFunction()" style="cursor:
pointer" ></i>
            <input class="searchBox" type="text" id="myInput"  name="search"
placeholder=" Cherchez ici..." >
        <th><a href="<%=request.getContextPath()%>/new"/><i class='fas fa-plus-circle'
style='font-size:40px;color:#4BC0DA'></i></th>
        <th style="border:0px"></th>
    </tr>
    <tr>
        <th>NAME</th>
        <th>PHONE</th>
        <th>EMAIL</th>
        <th>ADRESS</th>
        <th><a><i class='fas fa-redo-alt' style='font-size:32px;color:#076687'></i></a></
th>
        <th><a><i class='fas fa-trash-alt' style='font-size:32px;color:#FF7E5B'></i></
a></th>
    </tr>
    <a href="<%=request.getContextPath()%>/list"><i class="fa fa-address-book"
style="font-size:42px;color:darkgreen"></i></a>
    <%
        List<Contact> contacts =(List<Contact>) request.getAttribute("contacts");
        String name = request.getParameter("val");
        if (contacts != null) {
            for (Contact c: contacts) {
    %>
    <tr>
        <td><%=c.getNom ()%></td>
        <td><%=c.getTelephone ()%></td>
        <td><%=c.getEmail()%></td>
        <td><%=c.getAdresse ()%></td>
                <td><a href="edit?nom=<%= c.getNom ()%> "><i class='fas fa-redo-alt'
style='font-size:32px;color:#076687'></i></a></td>
        <td><a href="delete?nom=<%= c.getNom ()%>"><i class='fas fa-trash-alt'
style='font-size:32px;color:#FF7E5B'></i></a></td>
    </tr>
    <% }
    }%>
</table>
</body>
</html>
```

Gestion des Notes d'information (MVC-2)

L'objectif est de mettre en place une application JEE basée sur Servlet/Jsp/Beans implémentant le modèle MVC-2 donc une seule Servlet contrôleur. L'accès à un service se fera par un path unique dans l'URL.

L'application permettra à un utilisateur inscrit de gérer ses notes d'information (Ajouter, modifier supprimer, lister et chercher). Le scripte sql de la base de données (db_note.sql) est illustrée ci-dessous.

```
DROP TABLE IF EXISTS `note`;
CREATE TABLE IF NOT EXISTS `note`
(
  `id` int NOT NULL AUTO_INCREMENT,
  `title` varchar(50) NOT NULL,
  `content` text NOT NULL,
  `date` varchar(50) NOT NULL,
  `email` varchar(50) NOT NULL,
  PRIMARY KEY (`id`),
  KEY `fk_email` (`email`)
)
```

```
DROP TABLE IF EXISTS `user`;
CREATE TABLE IF NOT EXISTS `user` (
  `email` varchar(50) NOT NULL,
  `name` varchar(50) NOT NULL,
  `password` varchar(50) NOT NULL,
  `isAdmin` tinyint(1) NOT NULL,
  PRIMARY KEY (`email`)
)
```

Un Admin est un utilisateur une fois connecté à l'application verra un dashboard spécifique. Il aura la possibilité de :

- Lister les utilisateurs
- Chercher un utilisateur
- Modifier un utilisateur
- Supprimer un utilisateur

Un utilisateur de l'application, une fois authentifié doit pouvoir gérer ces notes d'information :

- Lister
- Ajouter
- Chercher
- Modifier
- Supprimer

Si l'utilisateur n'existe pas il faut lui proposer de s'inscrire .

NB:

Il faut gérer toutes les erreurs de l'application

Solution

Back-end

pom.xml

```
 <dependeny>
             <groupId>jakarta.servlet</groupId>
             <artifactId>jakarta.servlet-api</artifactId>
             <version>5.0.0</version>
             <scope>provided</scope>
  </dependency>
```

src/java : package estm.dsic.models

Note.java

```
 package estm.dsic.models;
 public class Note {
     private int id;
     private String title;
     private String content;
     private String date;
     private String email;
 //getters servers omis
 }
```

```
package estm.dsic.models;
public class User {
      private String name;
      private String email;
      private String pwd;
      private boolean isAdmin = false;
//getters servers omis
}
```

src/java : package estm.dsic.dals

Database.java

```
 package estm.dsic.dal;
 import java.sql.Connection;
 import java.sql.SQLException;
 import javax.naming.Context;
 import javax.naming.InitialContext;
 import javax.sql.DataSource;
 public class Database {
      private static Connection cnx;
      public static Connection getConnection() {
            if (cnx==null)
            try {
                  Context ctx=new InitialContext();
      DataSource ds=(DataSource)ctx.lookup("java:comp/env/jdbc/db_notes");
                  cnx=ds.getConnection();
                  if(cnx == null) {
                  System.out.println("error connecction");
                  }
                  return cnx;

            } catch (Exception e) {
                  // TODO Auto-generated catch block
                  e.printStackTrace();
            }
            return cnx;
      }
      private void closeConnection() {
            if (cnx != null) {
```

```
                try {
                      cnx.close();
                      System.out.println("Connection closed successfully");
                } catch (SQLException ex) {
                System.out.println("Failed to close JDBC Connection !");
                      ex.printStackTrace();
                }
          } else {
                System.out.println("No JDBC connection to close !");
          }
     }
}
```

NoteDao.java

```
package estm.dsic.dal;
import estm.dsic.models.Note;
import estm.dsic.models.User;
import java.sql.*;
import java.util.Vector;
public class NoteDao {
     public User login(User u) {
           User us = null;
           try {
                Connection cnx = Database.getConnection();
                PreparedStatement stm = cnx.prepareStatement("SELECT
email,name,   password, isAdmin FROM user WHERE email like ? AND password
like ?");
                stm.setString(1, u.getEmail());
                stm.setString(2, u.getPwd());
                ResultSet rs=stm.executeQuery();
                while(rs.next()){
                      us = new User();
                      us.setName(rs.getString(1));
                      us.setEmail(rs.getString(2));
                      us.setPwd(rs.getString(3));
                      us.setAdmin(rs.getBoolean(4));
                }
           } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
           }
           return us;
     }
     public boolean add(Note u) {
           int rs = 0;
           try {
                Connection cnx = Database.getConnection();
                PreparedStatement stm = cnx.prepareStatement("INSERT INTO
note(title, content, date, email) values(?, ?, ?, ?)",
Statement.RETURN_GENERATED_KEYS);
                stm.setString(1, u.getTitle());
                stm.setString(2, u.getContent());
```

```
                stm.setString(3, u.getDate());
                stm.setString(4, u.getEmail());
                rs=stm.executeUpdate();
                if (rs != 0){
                      System.out.println("updated");
                }
          } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
          }
          return rs!=0;
     }
     public boolean update(Note u) {
          int rs = 0;
          try {
                System.out.println("updating in db");
                Connection cnx = Database.getConnection();
                PreparedStatement stm = cnx.prepareStatement("UPDATE note set
title= ?, content = ?, date = ? where id= ?",
Statement.RETURN_GENERATED_KEYS);
                stm.setString(1, u.getTitle());
                stm.setString(2, u.getContent());
                stm.setString(3, u.getDate());
                stm.setInt(4, u.getId());
                rs=stm.executeUpdate();
                if (rs != 0){
                      System.out.println("updated");
                }
          } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
          }
          return rs!=0;

     }
     public boolean delete(Integer id) {
          int rs = 0;
          try {
                Connection cnx = Database.getConnection();
                PreparedStatement stm = cnx.prepareStatement("DELETE FROM
note where id= ?", Statement.RETURN_GENERATED_KEYS);
                stm.setInt(1, id);
                rs=stm.executeUpdate();
                if (rs != 0){
                      System.out.println("deleted");
                }
          } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
          }
          return rs!=0;
     }
```

```
	public Vector<Note> getNotes(String email) {
		ResultSet rs = null;
		Connection conn = null;
		Note note = new Note();
		Vector<Note> notes = new Vector<>();
		System.out.println("passed email is: " + email);
		try {
			conn = Database.getConnection();
			if (conn != null) {
				PreparedStatement ps = conn.prepareStatement("SELECT *
FROM note where email = ?");
				ps.setString(1, email);
				rs = ps.executeQuery();

				if (rs != null) {
					while (rs.next()) {
						note = new Note();
						note.setId(rs.getInt(1));
						note.setTitle(rs.getString(2));
						note.setContent(rs.getString(3));
						note.setDate(rs.getString(4));
						note.setEmail(rs.getString(5));
						notes.add(note);
					}
					System.out.println(notes);
				}
			}
		} catch (Exception ex) {
			ex.printStackTrace();
		} finally {
			try {
				if(rs!=null)
				{
					rs.close();
				}
			} catch (SQLException ex) {
				ex.printStackTrace();
			}
		}
		return notes;
	}
public Note getNote(String id) {
		ResultSet rs = null;
		Connection conn = null;
		Note us = new Note();
		try {
			conn = Database.getConnection();
			if (conn != null) {
				PreparedStatement ps = conn.prepareStatement(
						"select * from note where id = ?");
				ps.setString(1, id);
				rs = ps.executeQuery();
```

```
                              if (rs.next()) {
                                    us = new Note();
                                    us.setId(rs.getInt(1));
                                    us.setTitle(rs.getString(2));
                                    us.setContent(rs.getString(3));
                                    us.setDate(rs.getString(4));
                                    us.setEmail(rs.getString(5));
                              }
                        }
                  } catch (Exception ex) {
                        ex.printStackTrace();
                  } finally {
                        try {
                              if(rs!=null)
                              {
                                    rs.close();
                              }
                        } catch (SQLException ex) {
                              ex.printStackTrace();
                        }
                  }
                  return us;
            }
      }
```

src/java : package estm.dsic.services

NoteService.java

```
 package estm.dsic.services;
 import estm.dsic.dal.NoteDao;
 import estm.dsic.dal.UserDao;
 import estm.dsic.models.Note;
 import estm.dsic.models.User;
 import java.util.Vector;
 public class NoteService {
      private UserDao userdao=new UserDao();
      private NoteDao noteDao=new NoteDao();
      public Vector<Note> getNotes(String email) {
            return noteDao.getNotes(email);
      }
      public Note getNote(String id) {
            return noteDao.getNote(id);
      }
      public boolean update(Note note) {
            return noteDao.update(note);
      }
      public boolean add(Note note) {
            return noteDao.add(note);
      }
      public boolean remove(Integer note) {
            return noteDao.delete(note);
      }
 }
```

src/java : package estm.dsic.controllers

MainControler.java

```
package estm.dsic.controllers;
import com.example.docapp.util.DateFormatter;
import estm.dsic.models.Note;
import estm.dsic.models.User;
import estm.dsic.services.NoteService;
import estm.dsic.services.UserService;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.http.HttpSession;
import java.io.IOException;
import java.time.LocalDateTime;
import java.util.Vector;
import java.util.regex.Matcher;
import java.util.regex.Pattern;
@WebServlet(urlPatterns = "/")
public class MainController extends HttpServlet {
    private static final long serialVersionUID = 1L;
    private final UserService userService;
    private NoteService noteService;

    public MainController() {
        super();
        // TODO Auto-generated constructor stub
        userService = new UserService();
        noteService = new NoteService();
    }
protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {

        HttpSession session = request.getSession(true);
        User user = (User) session.getAttribute("user");
if  (request.getRequestURI().equals(getServletContext().getContextPath()  +  "/
auth")) {
            login(request, response, session);
        }
if  (request.getRequestURI().equals(getServletContext().getContextPath()  +  "/
register")) {
            register(request, response);
        }
if(request.getRequestURI().equals(getServletContext().getContextPath()  +  "/
logout")) {
            logout(request, response, session);
        }
```

```
if (request.getRequestURI().equals(getServletContext().getContextPath() + "/
edit")) {
            showEdit(request, response, session);
        }
if (request.getRequestURI().equals(getServletContext().getContextPath() + "/
editUser")) {
            edit(request, response, session);
        }
if (request.getRequestURI().equals(getServletContext().getContextPath() + "/
delete")) {
            showDelete(request, response, session);
        }
if (request.getRequestURI().equals(getServletContext().getContextPath() + "/
deleteUser")) {
            deleteUser(request, response, session);
        }
if (request.getRequestURI().equals(getServletContext().getContextPath() + "/
note")){
            showNote(request, response, session);
        }
if (request.getRequestURI().equals(getServletContext().getContextPath() + "/
editNote")){
            editNote(request, response, session);
        }
if (request.getRequestURI().equals(getServletContext().getContextPath() + "/
addNote")){
            newNote(request, response, session);
        }
if (request.getRequestURI().equals(getServletContext().getContextPath() + "/
newNote")){
            showNewNote(request, response, session);
        }
if (request.getRequestURI().equals(getServletContext().getContextPath() + "/
delNote")){
            showNoteDelete(request, response, session);
        }
if (request.getRequestURI().equals(getServletContext().getContextPath() + "/
deleteNote")){
            deleteNote(request, response, session);
        }
    }
       protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        doGet(request, response);
    }
      private void deleteNote(HttpServletRequest request, HttpServletResponse
response, HttpSession session) {
        System.out.println("deleting...");
        if (request.getParameter("id") != null) {
            String id = request.getParameter("id");
            boolean done = noteService.remove(Integer.valueOf(id));
            if (done) {
```

```
                session.setAttribute("message", "Success!");
                showHome(request, response,session);
                try {
                    response.sendRedirect(request.getContextPath() + "/auth");
                } catch (IOException e) {
                    throw new RuntimeException(e);
                }
            } else {
                session.setAttribute("message", "Erreur veuillez réessayer!");
                showHome(request, response,session);
                try {
                    response.sendRedirect(request.getContextPath() + "/auth");
                } catch (IOException e) {
                    throw new RuntimeException(e);
                }
            }
        }else{
            session.setAttribute("message", "Erreur veuillez réessayer!");
            showHome(request, response,session);
            try {
                response.sendRedirect(request.getContextPath() + "/auth");
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
        }
    }
      private void showNewNote(HttpServletRequest request, HttpServletResponse
response, HttpSession session) {
        try {
                    String id = request.getParameter("noteBtn");
                    System.out.println("ntdje" + id);
                                        request.getRequestDispatcher("User/
newNote.jsp").forward(request, response);
        } catch (ServletException | IOException e) {
            throw new RuntimeException(e);
        }
    }
          private void showNoteDelete(HttpServletRequest request,
HttpServletResponse response, HttpSession session) {
        try {
            String id = request.getParameter("deleteBtn");
            System.out.println(id);
            if (id == null) {
                response.sendRedirect(request.getContextPath() + "/auth");
            } else {
                Note note = noteService.getNote(id);
                System.out.println(note);
                if (note != null) {
                    request.setAttribute("currentNote", note);
                                        request.getRequestDispatcher("User/
delete.jsp").forward(request, response);
                }
```

```
            }
        } catch (ServletException | IOException e) {
            throw new RuntimeException(e);
        }
    }
        private void newNote(HttpServletRequest request, HttpServletResponse
response, HttpSession session) {
        String title = request.getParameter("title");
        String content = request.getParameter("content");
        Integer nb = (Integer) session.getAttribute("nbNotes");
        if (title != null && content != null) {
                String user = request.getParameter("email");
                Note note = new Note();
                note.setTitle(title);
                note.setContent(content);
                note.setEmail(user);
            note.setDate(LocalDateTime.now().format(DateFormatter.formatter));
                boolean a = noteService.add(note);
                if (a) {
                    nb++;
                    System.out.println("this is nb notes in new" + nb);
                    session.setAttribute("nbNotes", nb);
                    session.setAttribute("message", "Success!");
                    try {
                        showHome(request, response, session);
                           response.sendRedirect(request.getContextPath() + "/
auth");
                    } catch (IOException e) {
                        throw new RuntimeException(e);
                    }
                }else{
session.setAttribute("message", "Error please try again.");
                    try {
                        showHome(request, response, session);
response.sendRedirect(request.getContextPath() + "/auth");
                    } catch (IOException e) {
                        throw new RuntimeException(e);
                    }
                }

        } else {
            try {
                response.sendRedirect(request.getContextPath() + "/auth");
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
        }
    }
```

```
private void showNote(HttpServletRequest request, HttpServletResponse
response, HttpSession session) {
        try {
            String id = request.getParameter("editBtn");
            System.out.println(id);
            if (id == null) {
                response.sendRedirect(request.getContextPath() + "/auth");
            } else {
                Note n = noteService.getNote(id);
                System.out.println(n);
                if (n!= null) {
                    request.setAttribute("currentNote", n);
                                            request.getRequestDispatcher("User/
edit.jsp").forward(request, response);
                }
            }

        } catch (ServletException | IOException e) {
            throw new RuntimeException(e);
        }
    }
private void editNote(HttpServletRequest request, HttpServletResponse
response, HttpSession session) {
        if (request.getParameter("id") != null) {
            System.out.println("updating...");
            int id = Integer.parseInt(request.getParameter("id"));
            String title = request.getParameter("title");
            String content = request.getParameter("content");
            String date = LocalDateTime.now().format(DateFormatter.formatter);
            Note note = new Note();
            note.setId(id);
            note.setTitle(title);
            note.setContent(content);
            note.setDate(date);
            boolean done = noteService.update(note);
            if (done) {
                System.out.println("setting msg status");
                session.setAttribute("message", "Success!");
                try {
                    response.sendRedirect(request.getContextPath() + "/auth");
                } catch (IOException e) {
                    throw new RuntimeException(e);
                }
            } else {
                session.setAttribute("message", "Erreur veuillez réessayer!");
                try {
                    response.sendRedirect(request.getContextPath() + "/auth");
                } catch (IOException e) {
                    throw new RuntimeException(e);
                }
            }
        }else{
```

```
            request.setAttribute("message", "Erreur veuillez réessayer!");
            showHome(request, response,session);
        }
    }
private void showDelete(HttpServletRequest request, HttpServletResponse
response, HttpSession session) {
        try {
            String email = request.getParameter("deleteBtn");
            System.out.println(email);
            if (email == null) {
                response.sendRedirect(request.getContextPath() + "/auth");
            } else {
                User u = userService.getUser(email);
                System.out.println(u);
                if (u != null) {
                    request.setAttribute("currentUser", u);
                                        request.getRequestDispatcher("Admin/
delete.jsp").forward(request, response);
                }
            }
        } catch (ServletException | IOException e) {
            throw new RuntimeException(e);
        }
    }
private void deleteUser(HttpServletRequest request, HttpServletResponse
response, HttpSession session) {
        if (request.getParameter("email") != null) {
            String email = request.getParameter("email");
            User user = new User();
            user.setEmail(email);
            boolean done = userService.deleteUser(user);
            if (done) {
                request.setAttribute("message", "Success!");
                showHome(request, response,session);
            } else {
                request.setAttribute("message", "Erreur veuillez réessayer!");
                showHome(request, response,session);
            }
        }else{
            request.setAttribute("message", "Erreur veuillez réessayer!");
            showHome(request, response,session);
        }
    }

       private void showEdit(HttpServletRequest request, HttpServletResponse
response, HttpSession session) {
        try {
            String email = request.getParameter("editBtn");
            System.out.println(email);
            if (email == null) {
                response.sendRedirect(request.getContextPath() + "/auth");
            } else {
```

```
                User u = userService.getUser(email);
                System.out.println(u);
                if (u != null) {
                    request.setAttribute("currentUser", u);
                                        request.getRequestDispatcher("Admin/
edit.jsp").forward(request, response);
                }
            }

        } catch (ServletException | IOException e) {
            throw new RuntimeException(e);
        }
    }

        public void register(HttpServletRequest request, HttpServletResponse
response) {
        String email = request.getParameter("email");
        if (email != null) {
            String name = request.getParameter("name");
            String password = request.getParameter("pwd");
            String passwordconf = request.getParameter("pwdconf");
            String PASSWORD_PATTERN =
                    "^(?=.*[0-9])(?=.*[a-z])(?=.*[A-Z])(?=.*[!@#&()–[{}]:;',?/
*~$^+=<>]).{8,20}$";
                String emailPattern = "^(?=.{1,64}@)[A-Za-z0-9_-]+(\\.[A-Za-
z0-9_-]+)*@"
                    + "[^-][A-Za-z0-9-]+(\\.[A-Za-z0-9-]+)*(\\.[A-Za-z]{2,})
$";
            Pattern passPattern = Pattern.compile(PASSWORD_PATTERN);
            Matcher matcher = passPattern.matcher(password);
            Pattern pattern = Pattern.compile(emailPattern);
            Matcher Ematcher = pattern.matcher(email);
            if (!passwordconf.equals(password)) {
                try {
request.setAttribute("message", "Passwords do not match.");
                request.getRequestDispatcher("register.jsp").forward(request,
response);
                } catch (ServletException | IOException e) {
                    throw new RuntimeException(e);
                }
            } else if (!matcher.matches()) {
                try {
request.setAttribute("message", "Password must be at least 8 characters long
and contain at least one lowercase letter, one uppercase letter, and one
n u m b e r " ) ;
request.getRequestDispatcher("register.jsp").forward(request, response);
                } catch (ServletException | IOException e) {
                    throw new RuntimeException(e);
                }
            } else if (!Ematcher.matches()) {
request.setAttribute("message", "Please enter a valid email.");
                try {
```

```
 request.getRequestDispatcher("register.jsp").forward(request, response);
                } catch (ServletException | IOException e) {
                    throw new RuntimeException(e);
                }
            } else {
                User u = new User();
                u.setEmail(email);
                u.setName(name);
                u.setPwd(password);
                u.setAdmin(false);
                boolean a = userService.register(u);
                if (a) {
                    try {

request.setAttribute("message", "Succesfully registered, please login.");

request.getRequestDispatcher("index.jsp").forward(request, response);
                    } catch (ServletException | IOException e) {
                        throw new RuntimeException(e);
                    }
                }
            }
        } else {
            try {
request.getRequestDispatcher("register.jsp").forward(request, response);
            } catch (ServletException | IOException e) {
                throw new RuntimeException(e);
            }
        }
    }
public void login(HttpServletRequest request, HttpServletResponse response,
HttpSession session) {
        Object logged = session.getAttribute("loggedIn");
        System.out.println("Logged in:" + logged);
        if (logged == null) {
            String email = request.getParameter("email");
            String password = request.getParameter("pwd");
            System.out.println("Passed " + email + " " + password);
            if(email==null || password==null){
                System.out.println("going to index");
                try {
                    request.setAttribute("message", "Please login.");
                     request.getRequestDispatcher("login.jsp").forward(request,
response);
                } catch (ServletException | IOException e) {
                    throw new RuntimeException(e);
                }
            }else{
                User user = new User();
                user.setEmail(email);
                user.setPwd(password);
                user = userService.Login(user);
```

```
                System.out.println(user);
                if( user != null ){
                    session.setAttribute("loggedIn", true);
                    session.setAttribute("user", user);
                    showHome(request,response,session);
                }else{
                    System.out.println("going to index");
                    try {
request.setAttribute("message", "Invalid credentials.");

request.getRequestDispatcher("login.jsp").forward(request, response);
                    } catch (ServletException | IOException e) {
                        throw new RuntimeException(e);
                    }
                }
               System.out.println("user set into sesssion");
            }
        }else{
            showHome(request,response,session);
        }
    }
        public void showHome(HttpServletRequest request, HttpServletResponse
response, HttpSession session) {
        User user = (User) session.getAttribute("user");
        Integer nb = (Integer) session.getAttribute("nbNotes");
        Vector<User> users = null;
        Vector<Note> notes = null;
        if(user == null) {
            try {
                response.sendRedirect(request.getContextPath() + "/auth");
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
        }else{
            if (user.isAdmin()) {
                users = userService.getUsers();
                request.setAttribute("users", users);
                try {
request.getRequestDispatcher("Admin/admin.jsp").forward(request, response);
                } catch (ServletException | IOException e) {
                    throw new RuntimeException(e);
                }
            } else {
                try {
                    if(nb == null){
                        session.setAttribute("nbNotes", 0);
                    }
                    System.out.println("this is nb notes" + nb);
                    notes = noteService.getNotes(user.getEmail());
                    request.setAttribute("notes", notes);
                                        request.getRequestDispatcher("User/
user.jsp").forward(request, response);
```

```
                } catch (ServletException | IOException e) {
                    throw new RuntimeException(e);
                }
            }
        }

    }
     public void logout(HttpServletRequest request, HttpServletResponse
response, HttpSession session) {
        response.setContentType("text/html");
        response.setHeader("Cache-Control", "no-cache,no-store");
        response.setDateHeader("Expires", -1);
        response.setHeader("Pragma", "no-cache");
        session.invalidate();
        try {
            response.sendRedirect(request.getContextPath());
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
    public void edit(HttpServletRequest request, HttpServletResponse response,
HttpSession session) {
        if (request.getParameter("email") != null) {
            String email = request.getParameter("email");
            String password = request.getParameter("pwd");
            String name = request.getParameter("name");
            String isAdmin = request.getParameter("isAdmin");
            boolean admin = isAdmin.equals("admin");
            User user = new User();
            user.setEmail(email);
            user.setPwd(password);
            user.setName(name);
            user.setAdmin(admin);
            boolean done = userService.editUser(user);
            if (done) {
                session.setAttribute("message", "Success!");
                try {
                    response.sendRedirect(request.getContextPath() + "/auth");
                } catch (IOException e) {
                    throw new RuntimeException(e);
                }
            } else {
                session.setAttribute("message", "Erreur veuillez réessayer!");
                try {
                    response.sendRedirect(request.getContextPath() + "/auth");
                } catch (IOException e) {
                    throw new RuntimeException(e);
                }
            }
        }else{
            session.setAttribute("message", "Erreur veuillez réessayer!");
            showHome(request, response,session);
```

```
        }
    }

}
```

Front-end

Partie Admin : **admin.jsp**

```
<jsp:useBean id="message" scope="session" class="java.lang.String"/>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
         pageEncoding="ISO-8859-1" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html>
<html>
<head>
  <link  rel="stylesheet"  href="${pageContext.request.contextPath}/css/
bootstrap.min.css">
  <script  src="${pageContext.request.contextPath}/js/bootstrap.min.js"></
script>
  <script  src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/
jquery.min.js"></script>
    <meta charset="ISO-8859-1">
    <title>Home</title>
    <style>
        body {
            height: 100%;
            margin: 0;
            background: #F5F5F5;
        }
    </style>
</head>
<body>
<c:if test="${!empty message}">
      <div class="alert alert-info alert-dismissible fade show" role="alert"
id="successAlert">
            ${sessionScope.message}
          <button type="button" class="btn-close" data-bs-dismiss="alert" aria-
label="Close"></button>
    </div>
    <%
        System.out.println(session.getAttribute("message"));
        session.removeAttribute("message");
    %>
</c:if>
<div  class="modal  fade"  id="logoutModal"  tabindex="-1"  aria-
labelledby="logoutModalLabel" aria-hidden="true">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
                <h1 class="modal-title fs-5" id="logoutModalLabel">Logout</h1>
<button  type="button"  class="btn-close"  data-bs-dismiss="modal"  aria-
label="Close"></button>
```

```
            </div>
            <div class="modal-body">
                Are you sure you want to log out?
            </div>
<div class="modal-footer d-flex align-content-center justify-content-center">
<button type="button" class="btn btn-secondary" data-bs-dismiss="modal">No</
button>
<form action="${pageContext.request.contextPath}/logout" method="post">
<input class="btn btn-danger" type="submit" name="action" value="Yes"/>
                </form>
            </div>
        </div>
    </div>
</div>

<div class="card  mx-5 my-5">
    <div class="card-body d-flex justify-content-between">
        <p class="lead">
            Welcome back, ${user.getName()}
        </p>
          <button type="button" class="btn btn-primary" data-bs-toggle="modal"
data-bs-target="#logoutModal">
            Logout
        </button>
    </div>
</div>
<div class="card mx-5 my-5">
    <div class="card-header">
        <nav class="navbar bg-body-tertiary">
            <div class="container-fluid">
                <a class="navbar-brand">Users</a>
               <%-- <form class="d-flex" role="search">
                                <input class="form-control me-2" type="search"
placeholder="Search" aria-label="Search">
                                        <button class="btn btn-outline-success"
type="submit">Search</button>
                </form>--%>
            </div>
        </nav>
    </div>
    <div class="card-body">
        <table class="table">
            <thead>
            <tr>
                <th scope="col">Email</th>
                <th scope="col">Name</th>
                <th scope="col">isAdmin</th>
                <th scope="col">Action</th>
            </tr>
            </thead>
            <tbody>
            <jsp:useBean id="users" scope="request" class="java.util.Vector"/>
```

```
            <c:forEach items="${users}" var="us">
                <tr>
                    <td>
                        <p>${us.getName()}</p>
                    </td>
                    <td>
                        <p>${us.getEmail()}</p>
                    </td>
                    <td>
                        <p>${us.isAdmin()}</p>
                    </td>
                    <td>
  <div class="d-flex">
<form method="post" action="${pageContext.request.contextPath}/edit">

<button type="submit" class="btn btn-primary mx-4" name="editBtn"
                                        value="${us.getEmail()}">Edit
</button>
</form>
<form method="post" action="${pageContext.request.contextPath}/delete">
<button type="submit" class="btn btn-danger mx-4" name="deleteBtn"
                                        value="${us.getEmail()}">Delete
                                    </button>
                                </form>
                            </div>
                        </td>
                    </tr>
                </c:forEach>
                </tbody>
            </table>
        </div>
</div>
</body>
</html>
```

delete.jsp

```
<jsp:useBean id="currentUser" scope="request" type="estm.dsic.models.User"/>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html>
<html>
<head>
<link  rel="stylesheet"  href="${pageContext.request.contextPath}/css/
bootstrap.min.css">
       <script  src="${pageContext.request.contextPath}/js/bootstrap.min.js"></
script>
          <script  src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/
jquery.min.js"></script>
    <style>
        body {
            height: 100%;
```

```
            margin: 0;
            background: #F5F5F5;
        }
    </style>

<meta charset="ISO-8859-1">
<title>Edit user</title>
</head>
<body>
<!-- Modal -->
<div class="modal fade" id="EditModal" tabindex="-1" aria-
labelledby="EditModalLabel" aria-hidden="true">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
                <h1 class="modal-title fs-5" id="EditModalLabel">User</h1>
<button type="button" class="btn-close" data-bs-dismiss="modal" aria-
label="Close"></button>
            </div>
            <div class="modal-body">
                <input type="text" id="currentEmail">
            </div>
            <div class="modal-footer">
                    <button type="button" class="btn btn-secondary" data-bs-
dismiss="modal">Close</button>
<button type="button" class="btn btn-primary">Save changes</button>
            </div>
        </div>
    </div>
</div>
<br>
<div class="card  mx-5 my-5">
    <div class="card-body d-flex justify-content-between">
        <p class="lead">
            Welcome back,  ${user.getName()}
        </p>
        <div class="d-flex">
<form action="${pageContext.request.contextPath}/auth" method="post">
<input class="btn btn-success" type="submit" name="action" value="Back"/>
            </form>
        </div>
    </div>
</div>
<div class="card mx-5 my-5">
    <div class="card-header">
        <nav class="navbar bg-body-tertiary">
            <div class="container-fluid">
                <a class="navbar-brand">Delete user</a>
            </div>
        </nav>
    </div>
    <div class="card-body">
```

```
<form  action="${pageContext.request.contextPath}/deleteUser"  method="post"
id="form">
<div class="mb-3 text-center « >

<p class="lead">Are you sure you want to delete <b>${currentUser.email}</b> ?
</p>
<input  type="email"  class="form-control"  id="email"  name="email"  value="$
{currentUser.email}" hidden="hidden">
                      <input class="btn btn-danger fw-bold mb-2" name="action"
value="Delete" type="submit">
            </div>
        </form>
    </div>
</div>
</body>
</html>
```

Partie User

addNote.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE html>
<html>
<head>
         <link  rel="stylesheet"  href="${pageContext.request.contextPath}/css/
bootstrap.min.css">
       <script  src="${pageContext.request.contextPath}/js/bootstrap.min.js"></
script>
          <script  src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/
jquery.min.js"></script>
    <style>
        body {
            height: 100%;
            margin: 0;
            background: #F5F5F5;
        }
    </style>
<meta charset="ISO-8859-1">
<title>Edit note</title>
</head>
<body>
<div class="modal fade" id="EditModal" tabindex="-1" aria-
labelledby="EditModalLabel" aria-hidden="true">
    <div class="modal-dialog">
        <div class="modal-content">
            <div class="modal-header">
<h1 class="modal-title fs-5" id="EditModalLabel">User</h1>
<button  type="button"  class="btn-close"  data-bs-dismiss="modal"  aria-
label="Close"></button>
            </div>
            <div class="modal-body">
```

```
                <input type="text" id="currentEmail">
            </div>
            <div class="modal-footer">
                      <button type="button" class="btn btn-secondary" data-bs-
dismiss="modal">Close</button>
<button type="button" class="btn btn-primary">Save changes</button>
            </div>
        </div>
    </div>
</div>
<br>
<div class="card  mx-5 my-5">
    <div class="card-body d-flex justify-content-between">
        <p class="lead">
            Welcome back,  ${sessionScope.user.getName()}
        </p>
        <div class="d-flex">
                         <form  action="${pageContext.request.contextPath}/auth"
method="post">
                    <input class="btn btn-success" type="submit" name="action"
value="Back"/>
            </form>
        </div>
    </div>
</div>
<div class="card mx-5 my-5">
    <div class="card-header">
        <nav class="navbar bg-body-tertiary">
            <div class="container-fluid">
                <a class="navbar-brand">Edit note</a>
            </div>
        </nav>
    </div>
    <div class="card-body">
                     <form  action="${pageContext.request.contextPath}/addNote"
method="post" id="form">
                <input type="text" class="form-control" id="email" name="email"
value="${sessionScope.user.getEmail()}" placeholder="email" hidden="hidden">
            <div class="mb-3">
                <label for="title" class="form-label">Title</label>
                            <input type="text" class="form-control" id="title"
name="title"  placeholder="Enter title" required>
            </div>
            <div class="mb-3">
                <div class="form-floating">
                             <textarea class="form-control" placeholder="Write
something here" id="content" name="content" style="height: 200px" required></
textarea>
                    <label for="content">Content</label>
                </div>
            </div>
            <div class="d-grid">
```

```
                <input class="btn btn-lg btn-primary btn-login text-uppercase
fw-bold mb-2" name="action" value="Save" type="submit">
            </div>
        </form>
    </div>
</div>
</body>
</html>
```

Register.jsp

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
         pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<jsp:useBean id="message" scope="request" class="java.lang.String"/>
<html>
<head>
   <meta charset="ISO-8859-1">
    <title>Login</title>
         <link  rel="stylesheet"  href="${pageContext.request.contextPath}/css/
bootstrap.min.css">
</head>
<body>
<section class="vh-100" style="background-color: #508bfc;">
    <div class="container py-5 h-100">
<div class="row d-flex justify-content-center align-items-center h-100">
            <div class="col-12 col-md-8 col-lg-6 col-xl-5">
<div class="card shadow-2-strong" style="border-radius: 1rem;">
                    <div class="card-body p-5 text-center">
                       <h3 class="mb-5">Sign up</h3>
<form  action="${pageContext.request.contextPath}/register"  method="post"
id="form">
                            <div class="mb-3">
<label for="name" class="form-label">Nom complet </label>
<input  type="text"  class="form-control"  id="name"  name="name"
placeholder="Entrer votre nom complet" required>
                            </div>
                            <div class="mb-3">
<label for="email" class="form-label">Email </label>
<input  type="email"  class="form-control"  id="email"  name="email"
placeholder="Entrer votre email" required>
                            </div>
                            <div class="mb-3">
<label for="pwd" class="form-label" >Password </label>
<input  type="password"  class="form-control"  name="pwd"  id="pwd"
placeholder="Entrer votre mot de passe" required>
                            </div>
                            <div class="mb-3">
                                      <label for="pwdconf" class="form-label"
>Password </label>
<input  type="password"  class="form-control"  name="pwdconf"  id="pwdconf"
placeholder="Entrer votre mot de passe" required>
```

```
                            </div>
                            <c:if test="${!empty message}">
<div class="alert alert-info" role="alert" id="successAlert">
                                    ${message}
                                </div>
                            </c:if>

<div class="d-grid">
<input class="btn btn-primary btn-login fw-bold mb-2" name="action"
value="Register" type="submit">
<a class="btn btn-outline-success btn-login fw-bold mb-2" href="$
{pageContext.request.contextPath}/auth">Sign in</a>
                        </div>
                       </form>
                    </div>
                </div>
            </div>
        </div>
    </div>
</section>
</body>
</html>
```

XmlToUml JSF/XML

Partie I :

Modéliser un diagramme de classe UML en schéma XML, on rappel que une classe est définie par :

- un nom

- des attributs (chaque attribut a un nom et un type

- des méthodes (chaque méthode a un nom des arguments ou chaque argument a un type)

Les relations entre classes :

- Association (rôle et cardinalités)

- Héritage, Agrégation et composition

Partie II :

Créer une application JEE/JSF simple dans laquelle un client peut uploader sont fichier xml représentant un diagramme UML

- L'application vérifie si le fichier est valide par rapport au schéma établi dans I
- Si oui, on affiche au client le diagramme de classe associé
- Si non, on retourne le fichier avec les erreurs associées

Pour la réalisation du diagramme graphique correspondant à votre fichier XML vous pouvez vous baser sur javascript (api gojs)

Solution

src/java : package estm.lpdsic.xml2uml.models

Diagram.java

```
package estm.lpdsic.xml2uml.models;
import java.io.Serializable;
import java.util.Vector;
public class Diagram implements Serializable {
    private String name;
    private Vector<Class> classes = new Vector<>(1);
    private Vector<Association> associations = new Vector<>(0);
    //return json format
    @Override
    public String toString() {
        String jsonDiagram = "{" +
            "name: \"" + name + "\", " +
            (classes.size() > 0 ? "classes: " + classes + ", " : "") +
            (associations.size() > 0 ? "associations: " + associations : "") +
        "}";
        return jsonDiagram.replaceAll(", *}", "}");
    }
}
```

Class.java

```
package estm.lpdsic.xml2uml.dao;
import java.io.Serializable;
import java.util.Vector;
public class Class implements Serializable {
    private String name;
    private Vector<Attribute> attributes = new Vector<>(1);
    private Vector<Method> methods = new Vector<>(1);
    /**
     * @return JSON string of Class
     */
    @Override
    public String toString() {
        return "{" +
            "name: \"" + name + "\", " +
            "key: \"" + name + "\", " +
            (attributes != null ? "properties: " + attributes + ", " : "") +
            (methods != null ? "methods: " + methods : "") +
        "}";
    }
}
```

Association.java

```
public class Association implements Serializable {
    private String type;
    private AssociationItem from, to;
```

```
    @Override
    public String toString() {
        return "{" +
            "relationship: \"" + type + "\", " +
            "from: \"" + from.getClassName() + "\", " +
            (from.getRole() != null ? "fromRole: \"" + from.getRole() + "\", "
: "") +
                    (from.getCardinality() != null ? "fromCardinality: \"" +
from.getCardinality() + "\", " : "") +
            "to: \"" + to.getClassName() + "\", " +
            (to.getRole() != null ? "toRole: \"" + to.getRole() + "\", " : "")
+
                        (to.getCardinality()!= null ? "toCardinality: \"" +
to.getCardinality() + "\"" : "") +
        "}";
    }
}
```

XMLDiagramHandler.java

```
package estm.lpdsic.xml2uml.dao;
import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.SAXParseException;
import org.xml.sax.helpers.DefaultHandler;
public class XMLDiagramHandler extends DefaultHandler {
    private Diagram diagram;
    private Class aClass;
    private Attribute attribute;
    private Method method;
    private Argument argument;
    private Association association;
    private AssociationItem associationItem;
    private StringBuffer errors = new StringBuffer();

    @Override
    public void startDocument() throws SAXException {
        diagram = new Diagram();
    }
    @Override
       public void startElement(String uri, String localName, String qName,
Attributes attributes) throws SAXException {
        switch (qName) {
            case "diagram" -> {
if (attributes.getLength() > 0) diagram.setName(attributes.getValue(0));
            }
            case "class" -> {
                aClass = new Class();
                aClass.setName(attributes.getValue(0));
            }
            case "attribute" -> {
```

```
                attribute = new Attribute();
                for (int i = 0; i < attributes.getLength(); i++) {
                    switch (attributes.getQName(i)) {
ase "name" -> attribute.setName(attributes.getValue(i));
case "type" -> attribute.setType(attributes.getValue(i));
                        attribute.setDefaultValue(attributes.getValue(i));
case "visibility" -> attribute.setVisibility(attributes.getValue(i));
                    }
                }
            }
            case "method" -> {
                method = new Method();
                for (int i = 0; i < attributes.getLength(); i++) {
                    switch (attributes.getQName(i)) {
                        case "name" -> method.setName(attributes.getValue(i));
                        case "type" -> method.setType(attributes.getValue(i));
case "visibility" -> method.setVisibility(attributes.getValue(i));
                    }
                }
            }
            case "argument" -> {
                argument = new Argument();
                for (int i = 0; i < attributes.getLength(); i++) {
                    switch (attributes.getQName(i)) {
case "name" -> argument.setName(attributes.getValue(i));
case "type" -> argument.setType(attributes.getValue(i));
                    }
                }
            }
            case "association" -> {
                association = new Association();
                association.setType(attributes.getValue(0));
            }
            case "from", "to" -> {
                associationItem = new AssociationItem();
                for (int i = 0; i < attributes.getLength(); i++) {
                    switch (attributes.getQName(i)) {
case "class" -> associationItem.setClassName(attributes.getValue(i));
case "cardinality" -> associationItem.setCardinality(attributes.getValue(i));
case "role" -> associationItem.setRole(attributes.getValue(i));
                    }
                }
            }
        }
    }
    @Override
     public void endElement(String uri, String localName, String qName) throws
SAXException {
        switch (qName) {
            case "class" -> diagram.getClasses().add(aClass);
            case "attribute" -> aClass.getAttributes().add(attribute);
            case "method" -> aClass.getMethods().add(method);
```

```
            case "argument" -> method.getArguments().add(argument);
            case "association" -> diagram.getAssociations().add(association);
            case "from" -> association.setFrom(associationItem);
            case "to" -> association.setTo(associationItem);
        }
    }
    @Override
    public void warning(SAXParseException e) throws SAXException {
        errors.append("<div class='warning'>");
                errors.append("<b>at Line" + e.getLineNumber() + "<b>: " +
 e.getMessage());
        errors.append("</div>");
    }
    @Override
    public void error(SAXParseException e) throws SAXException {
        errors.append("<li class='error'>");
               errors.append("<b>at Line" + e.getLineNumber() + "</b>: " +
 e.getMessage());
        errors.append("</li>");
    }
    @Override
    public void fatalError(SAXParseException e) throws SAXException {
        error(e);
    }
    public Diagram getDiagram() {
        return diagram;
    }
    public String getErrors() {
        String allErrors = errors.toString();
          return allErrors.length() > 0 ? "<ul class='errors'>" + allErrors +
 "</ul>" : "";
    }
 }
```

src/java : package estm.lpdsic.xml2uml.beans

Util.java

```
 package estm.lpdsic.xml2uml.beans;
 import estm.lpdsic.xml2uml.models.XMLDiagramHandler;
 import org.xml.sax.SAXException;
 import javax.enterprise.context.SessionScoped;
 import javax.inject.Named;
 import javax.servlet.http.Part;
 import javax.xml.XMLConstants;
 import javax.xml.parsers.ParserConfigurationException;
 import javax.xml.parsers.SAXParser;
 import javax.xml.parsers.SAXParserFactory;
 import javax.xml.validation.SchemaFactory;
 import java.io.File;
 import java.io.InputStream;
 import java.io.Serializable;
```

```
import java.nio.file.Paths;

@Named(value = "xmlUtil")
@SessionScoped
public class Util implements Serializable {
    private final SAXParser saxParser;
    private Part uploadedFile;
    private String diagramJsObjectString;
    private String selectedFileName;
    private String errorsHTML;
    public Util() throws ParserConfigurationException, SAXException {
              SAXParserFactory parserFactory = SAXParserFactory.newInstance();
parserFactory.setSchema(SchemaFactory.newInstance(XMLConstants.W3C_XML_SCHEMA_
NS_URI).newSchema(getClass().getResource("/test.xsd")));
        saxParser = parserFactory.newSAXParser();
    }

    public void upload() {
        try {
File selectedFile = Paths.get(uploadedFile.getSubmittedFileName()).toFile();
            selectedFileName = selectedFile.getName();
            InputStream input = uploadedFile.getInputStream();
            XMLDiagramHandler diagramHandler = new XMLDiagramHandler();
            saxParser.parse(input, diagramHandler);
            diagramJsObjectString = diagramHandler.getDiagram().toString();
            errorsHTML = diagramHandler.getErrors();
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
    public String getDiagramJsObjectString() {
        return diagramJsObjectString;
    }
    public String getErrorsHTML() {
        return errorsHTML;
    }
    public Part getUploadedFile() {
        return uploadedFile;
    }
    public void setUploadedFile(Part uploadedFile) {
        this.uploadedFile = uploadedFile;
        upload();
    }
    public void setDiagramJsObjectString(String diagramJsObjectString) {
        this.diagramJsObjectString = diagramJsObjectString;
    }
    public void setSelectedFileName(String selectedFileName) {
        this.selectedFileName = selectedFileName;
    }
    public String getSelectedFileName() {
        return selectedFileName;
    }
```

```
}
```

Web:index.xhtml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html
        PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
        "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml"
      xmlns:h="http://xmlns.jcp.org/jsf/html"
      xmlns:c="http://java.sun.com/jsp/jstl/core"
      xmlns:f="http://java.sun.com/jsf/core">
<h:head>
    <title>UML Class Nodes</title>
        <meta name="description" content="UML Class-like nodes showing two
collapsible lists of items."/>
    <meta name="viewport" content="width=device-width, initial-scale=1"/>
    <meta name="charset" content="utf-8"/>
    <link rel="stylesheet" href="assets/index.css"/>
    <link rel="stylesheet" href="assets/icons.css"/>
    <script type="text/javascript" src="assets/go-debug.js"></script>
    <script type="text/javascript" src="assets/content.js"></script>
</h:head>

<h:body>
    <c:choose>
            <c:when test="#{!(xmlUtil.diagramJsObjectString eq null) and !
(xmlUtil.diagramJsObjectString eq '')}">
            <div class="header">
                <h:form>
                    <div class="input-file">
                                                <h:commandButton value="Return"
action="#{xmlUtil.setDiagramJsObjectString(null)}" />
                        <div class="button flex">
                            <i class="ic-return" />
                            <span>Retour</span>
                        </div>
                    </div>
                </h:form>
                        <h3 class="padding centered-text">UML: <h:outputLabel
value="#{xmlUtil.selectedFileName}" /></h3>
            </div>

            <div id="diagramArea">
                <div id="diagramName" class="padding centered-text"></div>
                <hr/>
                <c:choose>
                    <c:when test='#{xmlUtil.errorsHTML eq ""}'>
                        <div id="myDiagramDiv"></div>

                        <h:outputScript>
```

```
                            let diagram;
                            diagram =#{xmlUtil.diagramJsObjectString}

                            document.getElementById('diagramName').innerHTML =
diagram.name;
                                        init("myDiagramDiv", diagram.classes,
diagram.associations);
                        </h:outputScript>
                    </c:when>
                    <c:otherwise>
                        <div class="padding">
                                    <h:outputText value="#{xmlUtil.errorsHTML}"
escape="false" />
                        </div>
                    </c:otherwise>
                </c:choose>
            </div>
        </c:when>
        <c:otherwise>
            <div class="header padding" style="flex-direction: column">
                <h1 class="centered-text">XML 2 UML Converter</h1>
                <div class="centered-text">Simple to use</div>
            </div>

            <div class="main padding centered-text">
                  <div>Veillez Selectionner le fichier XML pour Vusualiser le
diagramme UML.</div>
                <hr style="margin: 18px -18px"/>
                <h:form id="f" enctype="multipart/form-data">
                    <div class="input-file">
                                <h:inputFile value="#{xmlUtil.uploadedFile}"
onchange="document.forms['f'].submit()">
                            <!--<f:ajax listener="#{xmlUtil.upload}"/>-->
                        </h:inputFile>
                        <!-- Custom upload button -->
                        <div class="button" style="padding: 24px 0">
                            <div class="submit">
                                <span>Upload</span>
                                <i class="ic-upload"/>
                            </div>
                        </div>
                    </div>
                </h:form>
            </div>
        </c:otherwise>
    </c:choose>
</h:body>

</html>
```

Atelier JSF /EJB /JPA

L'application consiste en un portail en JEE qui permettra à une entreprise de gérer ses contacts.
Un contact est défini par (#Nom,Tele,email,Adresse)
L'inscription de l'entreprise dans le portail se fera à travers la page d'indexe qui n'est rien qu'une page d'authentification pour les entreprises déjà inscrite.
Une fois authentifiée l'entreprise doit pouvoir gérer ses contacts par :

- L'Ajout
- La suppression
- La modification et
- La recherche des contacts

A travers une page gestionContacts dont le format est le suivant :

Déconnexion

Recherche	Recherche	Recherche	Recherche	Add	
Nom	Tele	Email	Adresse	Update	Delete

- Lors de l'ajout d'un contact, il faut vérifier le format Email et Tele et qu'il n'existe pas déjà dans la base de donnée
- L'application doit supporter les deux langues à savoir le Français et l'Anglais aussi bien dans les labels que les messages d'erreurs
- La recherche doit être faite en AJAX
- Module EJB avec deux packages entities (Deux entities User et Contact)et et business (Deux Interfaces Remote : IUser (Auticate et Inscription) et IContact (contenant les services : Add, Update, Delete, Search et List)) et les EJBs nécessaires
- Module JSF contenant les managedBean et les pages xhtml nécessaires

Remarque :
Vous allez créer un projet Entreprise Archive dans encastrant deux projets :

- Projet EJB/JPA
- Projet WEB en JSF

Solution Projet Ejb

pom.xml

```
<dependency>
            <groupId>javax.ejb</groupId>
            <artifactId>javax.ejb-api</artifactId>
            <version>3.2.2</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>javax.persistence</groupId>
            <artifactId>javax.persistence-api</artifactId>
            <version>2.2</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>javax.transaction</groupId>
            <artifactId>javax.transaction-api</artifactId>
            <version>1.3</version>
            <scope>provided</scope>
        </dependency>
```

src/java :package estm.dsic.entities

Contact.java

```
package estm.dsic.entities;
import javax.persistence.*;
import java.io.Serializable;
@NamedQuery(
        name = "USER_CONTACTS",
        query = "SELECT telephone, nom, email, adresse FROM Contact"
)
@NamedQuery(
        name = "FIND_CONTACT",
        query = "SELECT telephone, nom, email, adresse FROM Contact WHERE
telephone = :telephone"
)
@Entity
@Table(name = "contactsTable")
public class Contact implements Serializable {
    @Id
    protected String telephone;
    protected String nom;
    protected String email;
    protected String adresse;
//Getters et setters omis
}
```

User.java

```
package estm.dsic.entities;
import javax.persistence.*;
import java.io.Serializable;
import java.util.List;

@NamedQuery(
        name = "CHECK_AUTH",
        query = "SELECT username, password FROM User WHERE username = :username
AND password = :password"
)
@Entity
@Table(name = "usersTable")
public class User implements Serializable {
    @Id
    private String username;
    private String password;
    @OneToMany
    @JoinColumn(name = "username")
    private List<Contact> contacts;
//getters setters omis
   }
```

src/java :package estm.dsic.business

IContact.java

```
package estm.dsic.business;
import estm.dsic.entities.Contact;
import estm.dsic.entities.User;
import javax.ejb.Remote;
import java.util.List;
@Remote
public interface IContact {
    boolean add(String telephone, String email, String nom, String adresse);
    boolean modify(String tele, Contact contact);
    boolean delete(String telephone);
    List<Contact> getAll();
    List<Contact> findByPhone(String phone);
    List<Contact> findByName(String nom);
    List<Contact> findByEmail(String email);
    List<Contact> findByAddress(String address);
}
```

IUser.java

```
package estm.dsic.business;
import estm.dsic.entities.User;
import javax.ejb.Remote;
@Remote
public interface IUser {
    boolean authenticate(User user);
    boolean inscription(User user);
}
```

src/java :package estm.dsic.business

ContactManager.java

```
package estm.dsic.business;
import estm.dsic.entities.Contact;
import estm.dsic.entities.User;
import javax.ejb.Stateful;
import javax.persistence.EntityManager;
import javax.persistence.EntityTransaction;
import javax.persistence.NoResultException;
import javax.persistence.PersistenceContext;
import java.util.List;
@Stateful(mappedName = "contactManager")
public class ContactManager implements IContact {
    @PersistenceContext(unitName = "EJBannuaire")
    EntityManager entityManager;
    @Override
    public boolean add(String telephone, String email, String nom, String
adresse) {
        Contact contact = new Contact();
        contact.setTelephone(telephone);
        contact.setEmail(email);
        contact.setNom(nom);
        contact.setAdresse(adresse);
        System.out.println(contact);
        System.out.println(telephone);
        if (findByPhone(contact.getTelephone()).size() <= 0) {
            entityManager.persist(contact);
            return true;
        }
        return false;
    }
   @Override
    public boolean modify(String tele, Contact contact) {
        return false;
    }
    @Override
    public boolean delete(String telephone) {
        try
        {
            Contact contact = entityManager.find(Contact.class, telephone);
            entityManager.remove(contact);
            entityManager.flush();
        }
        catch(Exception e)
        {
System.out.println("Unable to delete : there are references to it.");
            return false;
        }
        return true;
    }
    @Override
```

```
    public List<Contact> getAll() {
        try {
return entityManager.createNamedQuery("USER_CONTACTS").getResultList();
        } catch (NoResultException exception) {
            return null;
        }
    }
```

Solution Projet JSF

pom.xml

```
 <dependency>
            <groupId>javax.enterprise</groupId>
            <artifactId>cdi-api</artifactId>
            <version>2.0</version>
        </dependency>
        <dependency>
            <groupId>javax.ejb</groupId>
            <artifactId>javax.ejb-api</artifactId>
            <version>3.2.2</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>javax.faces</groupId>
            <artifactId>javax.faces-api</artifactId>
            <version>2.3</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>javax.servlet</groupId>
            <artifactId>javax.servlet-api</artifactId>
            <version>4.0.1</version>
            <scope>provided</scope>
</dependency>
```

src/java :package estm.dsic.beans

ContactBean.java

```
package estm.dsic.beans;
import estm.dsic.business.IContact;
import estm.dsic.entities.Contact;
import javax.ejb.EJB;
import javax.enterprise.context.SessionScoped;
import javax.faces.application.FacesMessage;
import javax.faces.context.FacesContext;
import javax.inject.Named;
import java.io.Serializable;
import java.util.List;
@Named
@SessionScoped
public class ContactBean implements Serializable {
    private String telephone, email, nom, adresse;
    @EJB(lookup = "java:global/Ateliers/AtelierEJB/ContactManager")
```

```
private IContact contactManager;

    public boolean addContact(){
        if(!contactManager.add(telephone, email, nom, adresse)){
            FacesMessage msg = new FacesMessage("Vous ne pouvez pas ajouter ce
contact !");
            FacesContext.getCurrentInstance().addMessage("addForm:addContact",
msg);
            return false;
        }
        return true;
    }
public boolean remove(String telephone){
        System.out.println(telephone);
        if (!contactManager.delete(telephone)) {
            FacesMessage msg = new FacesMessage("Vous ne pouvez pas supprimer ce
contact !");

FacesContext.getCurrentInstance().addMessage("ManageAllContacts:deleteContact",
msg);
            return false;
        }
        return true;
    }
//getters et setters omis
}
```

UserBean.java

```
package estm.dsic.beans;
import estm.dsic.business.IContact;
import estm.dsic.business.IUser;
import estm.dsic.entities.Contact;
import estm.dsic.entities.User;
import javax.ejb.EJB;
import javax.enterprise.context.SessionScoped;
import javax.faces.application.FacesMessage;
import javax.faces.context.FacesContext;
import javax.inject.Named;
import java.io.Serializable;
@Named(value = "user")
@SessionScoped
public class UserBean extends User implements Serializable {
    private final User user = new User();
    @EJB(lookup = "java:global/Atelier3/Atelier3EJB/UserManager")
    private IUser userManager;
    @EJB(lookup = "java:global/Atelier3/Atelier3EJB/ContactManager")
    private IContact contactManager;
    public boolean login() {
        user.setUsername(username);
        user.setPassword(password);
```

```
        if (!userManager.authenticate(user)) {
FacesMessage msg = new FacesMessage("Username ou Password est incorrecte");
FacesContext.getCurrentInstance().addMessage("Auth:checkLogin", msg);
            return false;
        }
        return true;
    }
    public boolean inscrire() {
        user.setUsername(username);
        user.setPassword(password);

        if (!userManager.inscription(user)) {
            FacesMessage msg = new FacesMessage("Username est déjà existe !
Créez un nouveau");
            FacesContext.getCurrentInstance().addMessage("Auth:checkInscrire",
msg);
            return false;
        }
        return true;
    }
}
```

Faces-config.xml

```
<?xml version='1.0' encoding='UTF-8'?>
<faces-config version="2.2" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
              xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
              xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
    http://xmlns.jcp.org/xml/ns/javaee/web-facesconfig_2_2.xsd">
    <navigation-rule>
        <from-view-id>/index.xhtml</from-view-id>
        <navigation-case>
            <from-action>#{user.login}</from-action>
            <from-outcome>true</from-outcome>
            <to-view-id>/contacts.xhtml</to-view-id>
            <redirect/>
        </navigation-case>
        <navigation-case>
            <from-action>#{user.login}</from-action>
            <from-outcome>false</from-outcome>
            <to-view-id>/index.xhtml</to-view-id>
        </navigation-case>
    </navigation-rule>

    <navigation-rule>
        <from-view-id>/inscription.xhtml</from-view-id>
        <navigation-case>
            <from-action>#{user.inscrire}</from-action>
            <from-outcome>true</from-outcome>
            <to-view-id>/index.xhtml</to-view-id>
        </navigation-case>
        <navigation-case>
            <from-action>#{user.inscrire}</from-action>
```

```
            <from-outcome>false</from-outcome>
            <to-view-id>/inscription.xhtml</to-view-id>
        </navigation-case>
    </navigation-rule>
    <navigation-rule>
    <from-view-id>/addcontact.xhtml</from-view-id>
        <navigation-case>
            <from-action>#{contactBean.addContact}</from-action>
            <from-outcome>true</from-outcome>
            <to-view-id>/contacts.xhtml</to-view-id>
        </navigation-case>
        <navigation-case>
            <from-action>#{contactBean.addContact}</from-action>
            <from-outcome>false</from-outcome>
            <to-view-id>/addcontact.xhtml</to-view-id>
        </navigation-case>
    </navigation-rule>
</faces-config>
```

Jackarta/OpenLiberty

Projet Authentification JSF

Pom.xml

```
<parent>
        <groupId>io.openliberty.tools</groupId>
        <artifactId>liberty-maven-app-parent</artifactId>
        <version>3.7</version>
    </parent>
<dependency>
            <groupId>jakarta.platform</groupId>
            <artifactId>jakarta.jakartaee-web-api</artifactId>
            <version>10.0.0</version>
 </dependency>
    <plugin>
                <groupId>io.openliberty.tools</groupId>
                <artifactId>liberty-maven-plugin</artifactId>
                <version>3.7</version>
                <configuration>
                    <copyDependencies>
                      <dependencyGroup>
                           <location>jdbc</location>
                       <dependency>
                         <groupId>mysql</groupId>
                         <artifactId>mysql-connector-java</artifactId>
                          <version>8.0.30</version>
                          </dependency>
                      </dependencyGroup>
                     </copyDependencies>
                    <assemblyArtifact>
                        <groupId>io.openliberty</groupId>
                        <artifactId>openliberty-runtime</artifactId>
                        <version>[21.0.0.3,)</version>
                        <type>zip</type>
                    </assemblyArtifact>
                    <serverName>${project.artifactId}Server</serverName>
                    <include>${packaging.type}</include>
                </configuration>
            </plugin>
```

Liberty : Server.xml

```
<server description="Sample Servlet server">
    <featureManager>
        <feature>cdi-3.0</feature>
        <feature>faces-3.0</feature>
        <feature>jdbc-4.2</feature>
    </featureManager>
    <library id="jdbcLib">
        <fileset dir="jdbc" includes="*.jar"/>
    </library>
```

```
    <dataSource id="DefaultDataSource" jndiName="jdbc/mydb">
        <jdbcDriver libraryRef="jdbcLib"/>
        <connectionManager maxPoolSize="20" minPoolSize="5"
        connectionTimeout="10s" agedTimeout="30m"/>
        <properties serverName="localhost" portNumber="3306"
                    databaseName="db_users"
                    user="root"
                    password="123456789"/>
    </dataSource>
    <httpEndpoint httpPort="9080" httpsPort="9443" id="defaultHttpEndpoint" />
    <webApplication id="jsf001" location="jsf001.war" name="jsf001"/>
</server>
```

Src/java : package estm.dsic.jee.controllers

FacesApp.java

```
package estm.dsic.jee.controllers;
import jakarta.enterprise.context.ApplicationScoped;
import jakarta.faces.annotation.FacesConfig;
@FacesConfig
@ApplicationScoped
public class FacesApp {

}
```

Login.java

```
package estm.dsic.jee.controllers;
import java.io.Serializable;
import jakarta.enterprise.context.SessionScoped;
import jakarta.inject.Named;
@Named
@SessionScoped
public class Login implements Serializable{
    private String email;
    private String passwd;
//getters setters omis
}
```

Logout.java

```
package estm.dsic.jee.controllers;
import java.io.Serializable;
import jakarta.enterprise.context.SessionScoped;
import jakarta.faces.context.ExternalContext;
import jakarta.faces.context.ExternalContextFactory;
import jakarta.inject.Inject;
import jakarta.inject.Named;
@Named
@SessionScoped
public class Logout implements Serializable{
    @Inject ExternalContext ec;
    public String logout(){
        ec.invalidateSession();
        return "login";
```

```
    } }
```

Src/java : package estm.dsic.jee.dal

UserDao.java

```
package estm.dsic.jee.dal;
import java.sql.Statement;
import java.sql.Connection;
import java.sql.ResultSet;
import java.sql.SQLException;
import javax.sql.DataSource;
import jakarta.annotation.Resource;
public class UserDao {
    @Resource(name = "jdbc/mydb")
    private DataSource mydb;
    private Connection cnx;
    private Statement stm;
    private ResultSet rs;
      public boolean auth(String email,String passwd){
            try {
                cnx=mydb.getConnection();
                stm=cnx.createStatement();
                rs=stm.executeQuery("SELECT * FROM T_user WHERE
email='"+email+"' AND password='"+passwd+"';");
                if(rs.next()) return true;
            } catch (SQLException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            return false;
    }    }
```

Src/java : package estm.dsic.jee.services

UserServices.java

```
import estm.dsic.jee.controllers.Login;
import estm.dsic.jee.dal.UserDao;
import jakarta.enterprise.inject.Model;
import jakarta.faces.annotation.ManagedProperty;
import jakarta.faces.application.FacesMessage;
import jakarta.faces.context.FacesContext;
import jakarta.inject.Inject;
@Model
public class UserServices {
    @Inject FacesContext facesContext;
    @Inject
    @ManagedProperty(value = "#{login}")
    private Login login;
    @Inject
    private UserDao userDao;
    public String check(){
        if (userDao.auth(login.getEmail(),login.getPasswd())){
```

```
            return "app";
        }
        else {
            FacesMessage message=new FacesMessage("login or password
incorrect");
            facesContext.addMessage("form_login:id_login", message);
            return "login";
        }
    }
}
```

Projet Web Service REST/Persistance

pom.xml

```
 <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
             <version>8.0.30</version>
             </dependency>
        <dependency>
            <groupId>jakarta.platform</groupId>
            <artifactId>jakarta.jakartaee-api</artifactId>
            <version>9.1.0</version>
            <scope>provided</scope>
        </dependency>
        <dependency>
            <groupId>org.eclipse.microprofile</groupId>
            <artifactId>microprofile</artifactId>
            <version>5.0</version>
            <type>pom</type>
            <scope>provided</scope>
        </dependency>
```

server.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<server description="new server">
<featureManager>
    <!-- Enable features -->
    <feature>restfulWS-3.0</feature>
    <feature>jsonb-2.0</feature>
    <feature>jsonp-2.0</feature>
    <feature>cdi-3.0</feature>
    <feature>jdbc-4.3</feature>
    <feature>persistence-3.0</feature>
    </featureManager>
    <library id="jdbcLib">
        <fileset dir="jdbc" includes="*.jar"/>
    </library>

    <dataSource id="DefaultDataSource" jndiName="jdbc/mydb">
        <jdbcDriver libraryRef="jdbcLib"/>
        <connectionManager maxPoolSize="20" minPoolSize="5"
```

```
        connectionTimeout="10s" agedTimeout="30m"/>
        <properties serverName="localhost" portNumber="3306"
                    databaseName="db_notes"
                    user="root"
                    password="123456789"
                    />
    </dataSource>
<httpEndpoint httpPort="9080" httpsPort="9443" id="defaultHttpEndpoint" />
    <webApplication id="rest" location="rest.war" name="rest"/>
</server>
```

RestApplication.java

```
package estm.dsic.jee.rest;
import jakarta.ws.rs.ApplicationPath;
import jakarta.ws.rs.core.Application;
@ApplicationPath("/api")
public class RestApplication extends Application {

}
```

src/java : package estm.dsic.jee.rest.models

User.java

```
package estm.dsic.jee.rest.entities;
import java.io.Serializable;
import jakarta.persistence.Column;
import jakarta.persistence.Entity;
import jakarta.persistence.Id;
import jakarta.persistence.Table;
@Entity
@Table(name = "T_user")
public class User implements Serializable{
    @Id
    private String email;
    @Column
    private String name;
    @Column
    private String password;
    @Column
    private boolean isAdmin;
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {
        this.email = email;
    }
    public String getPassword() {
```

```
        return password;
    }
    public void setPassword(String password) {
        this.password = password;
    }
    public boolean isAdmin() {
        return isAdmin;
    }
    public void setAdmin(boolean isAdmin) {
        this.isAdmin = isAdmin;
    }
}
```

src/java : package estm.dsic.jee.rest.entities

UserEntity.java

```
import jakarta.persistence.PersistenceContext;
import jakarta.persistence.Query;
public class UserEntity {
    @PersistenceContext(name = "mydb")
    EntityManager em;
    public void add(User user){
        em.persist(user);
    }
    public User search(String email){
        return em.find(User.class, email);
    }
    public void update(User user){
       User u= em.find(User.class, user.getEmail());
       if (u!=null){
        em.merge(user);
       }
    }
    public List<User> getUsers(){
        Query query=em.createQuery("SELECT u FROM User u");
        return query.getResultList();
    }
}
```

src/java : package estm.dsic.jee.rest.services

UserServices.java

```
package estm.dsic.jee.rest.services;
import java.util.List;
import java.util.Vector;

import estm.dsic.jee.rest.dao.UserDao;
import estm.dsic.jee.rest.dao.UserEntity;
import estm.dsic.jee.rest.entities.User;
import jakarta.inject.Inject;
```

```
public class UserServices {
    @Inject
    private UserEntity userDao;
    public List<User> getAll(){
        return userDao.getUsers();
    }
}
```