|  |
| --- |
| FPT POLYTECHNIC |
| **DỰ ÁN 1** |
| NGÀNH CÔNG NGHỆ THÔNG TIN (ỨNG DỤNG PHẦN MỀM) |



|  |
| --- |
| Đà Nẵng 2019 |

|  |
| --- |
| GIẢNG VIÊN HƯỚNG DẪN: NGUYỄN VĂN ĐỊNH |
| SINH VIÊN THỰC HIỆN |
| 1. Đỗ Văn Liêm (PD02596), Trưởng nhóm 2. Nguyễn Văn Tân(PD02634) 3. Trương Thế Công(PD02627) |

MỤC LỤC

[1 Phân tích 2](#_Toc520132211)

[1.1 Hiện trạng 2](#_Toc520132212)

[1.2 Yêu cầu hệ thống 2](#_Toc520132213)

[1.3 Use case 2](#_Toc520132214)

[2 Thiết kế 2](#_Toc520132215)

[2.1 Mô hình triển khai 2](#_Toc520132216)

[2.2 Thiết kế CSDL 2](#_Toc520132217)

[2.2.1 Sơ đồ quan hệ thực thể 2](#_Toc520132218)

[2.2.2 Thiết kế chi tiết các thực thể 2](#_Toc520132219)

[2.3 Thiết kế giao diện 2](#_Toc520132220)

[2.3.1 Sơ đồ tổ chức giao diện 2](#_Toc520132221)

[2.3.2 Thiết kế giao diện cho các chức năng nghiệp vụ 2](#_Toc520132222)

[3 Thực hiện viết mã 2](#_Toc520132223)

[3.1 Viết mã tạo CSDL 2](#_Toc520132224)

[3.1.1 Tạo CSDL 2](#_Toc520132225)

[3.1.2 SQL truy vấn và thao tác 2](#_Toc520132226)

[3.1.3 Các thủ tục lưu tổng hợp thống kê 2](#_Toc520132227)

[3.2 Lập trình JDBC 3](#_Toc520132228)

[3.2.1 Lớp hỗ trợ 3](#_Toc520132229)

[3.2.2 Model class - Các lớp mô tả dữ liệu 3](#_Toc520132230)

[3.2.3 DAO Class - Các lớp truy xuất dữ liệu 3](#_Toc520132231)

[3.3 Viết mã cho ứng dụng 3](#_Toc520132232)

[3.3.1 Xử lý Form X 3](#_Toc520132233)

[3.3.2 Xử lý Form Y 3](#_Toc520132234)

[4 Kiểm thử 3](#_Toc520132235)

[4.1 Kiểm thử form X 3](#_Toc520132236)

[4.2 Kiểm thử form Y 3](#_Toc520132237)

[5 Đóng gói và triển khai 3](#_Toc520132238)

[5.1 Hướng dẫn chuyển đổi jar thành exe 3](#_Toc520132239)

[5.2 Hướng dẫn cài đặt triển khai 3](#_Toc520132240)

[5.3 Hướng dẫn sử dụng phần mềm 3](#_Toc520132241)

# Phân tích

## Hiện trạng

* Tình hình thực Tế:

+ Ngành bất động sản là một ngành có thị trường to lớn, một lượng giao dịch lớn diễn ra hằng ngày. Một số công ty hiện nay còn áp dụng cách quản lý bằng sổ sách dẫn đến cách quản lý, truy vấn dữ liệu cực kỳ khó khăn, ngoài ra còn có các vấn đề về bảo mật, chính xác thông tin. Việc ứng dụng 1 app vào việc quản lý trở nên bức thiết

* Ứng dụng : quản lý mua bán bất động sản .

## Yêu cầu hệ thống

* Yêu cầu về lưu trữ

+ Thông tin người dùng

+ Thông tin hình ảnh

+ Thông tin dự án

+ Thông tin sản phẩm

+ Thông tin đối tác

+ Thông tin chi nhánh quản lí BĐS

* Quản lý các chức năng:

+ Quản lý tài khoản người dùng

+ Quản lý thông tin Sản phẩm

+ Quản lý thông tin dự án

+ Quản lí thông tin chi nhánh

+ Quản lí thông tin đối tác

+ Phân tích dự án , sản phẩm

+ Thống kê doanh thu , dự án , đối tác

* Báo cáo thống kê
* Thống kê dự án theo năm

+ Số đối tác tiếp cận dự án

+ Số đối tác đầu tư

+ Số khách hàng đặt mua sản phẩm của dự án

. . .

* Thống kê doanh thu theo năm

+ Tổng doanh thu

+ Doanh thu cao nhất

+ Doanh thu thấp nhất

+ Doanh thu trung bình của năm

. . .

* Bảo mật

+ Tất cả mọi thành viên phải đăng nhập mới sử dụng được phần mềm

+ Người quản lí được phép thực hiện tất cả các chức năng

+ Nhân viên không được phép thêm, sửa, xóa dữ liệu.

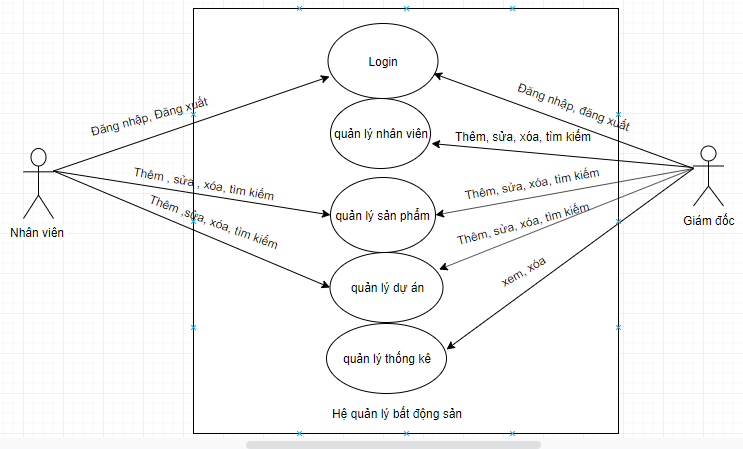
* Yêu cầu về môi trường công nghệ

+ Ứng dụng phải được thực với công nghệ Swing và JDBC chạy trên mọi hệ

điều hành với môi trường JDK tối thiểu 1.8

+ Hệ quản trị CSDL SQL Server 2008 trở lên

## Use case



- CHI TIẾT CÁC CHỨC NĂNG

\* Mỗi chức năng quản lý bao gồm các chức năng con

+ Xem: xem tất cả và chi tiết một mục

+ Thêm: thêm mới vào cơ sở dữ liệu

+ Xóa: xóa theo mã

+ Sửa: cập nhật dữ liệu đang xem

+ Tìm kiếm: tìm kiếm theo điều kiện

+ Điều hướng: di chuyển đến dữ liệu của bản ghi chi tiết trước và sau

+ Các chức năng thao tác dữ liệu (thêm, sửa, xóa) cần được kiểm lỗi hợp lý với dữ liệu

\*Chức năng tổng hợp thống kê số liệu

- Thống kê dự án phải có cấu trúc là :

+ Tên dự án

+ Số sản phẩm

+ Số nhà đầu tư

+ Số khách hàng

+ Doanh thu

* Thống kê đối tác phải có cấu trúc là

+ Logo

+ Tên đối tác

+ Số dự án tham gia

+ Số vốn đã đầu tư

* Thống kê doanh thu phải có cấu trúc là

+ Năm

+ Số dự án

+ Số sản phẩm

+ Tổng doanh thu

+ Doanh thu cao nhất

+ Doanh thu thấp nhất

+ Doanh thu trung bình trong năm

\* Chức năng đăng nhập và đăng xuất

+ Đăng nhập: để sử dụng phần mềm, bắt buộc phải đăng nhập

+ Đăng xuất: đăng nhập lại với người dùng khác hoặc dừng lại đi đâu đó

\*Chức năng phân tích

+ Phân tích dự án

+ Phân tích sản phẩm

+ Phân tích đánh giá

# Thiết kế

## Mô hình triển khai

Ứng dụng phần mềm được xây dựng để phục vụ cho nhiều người dùng nhưng cơ sở dữ liệu thì lưu trữ tập trung.

C:\Program Files\Microsoft Office\MEDIA\CAGCAT10\j0285750.wmf

SQL Sever

C:\Program Files\Microsoft Office\MEDIA\CAGCAT10\j0195384.wmf C:\Program Files\Microsoft Office\MEDIA\CAGCAT10\j0292020.wmf

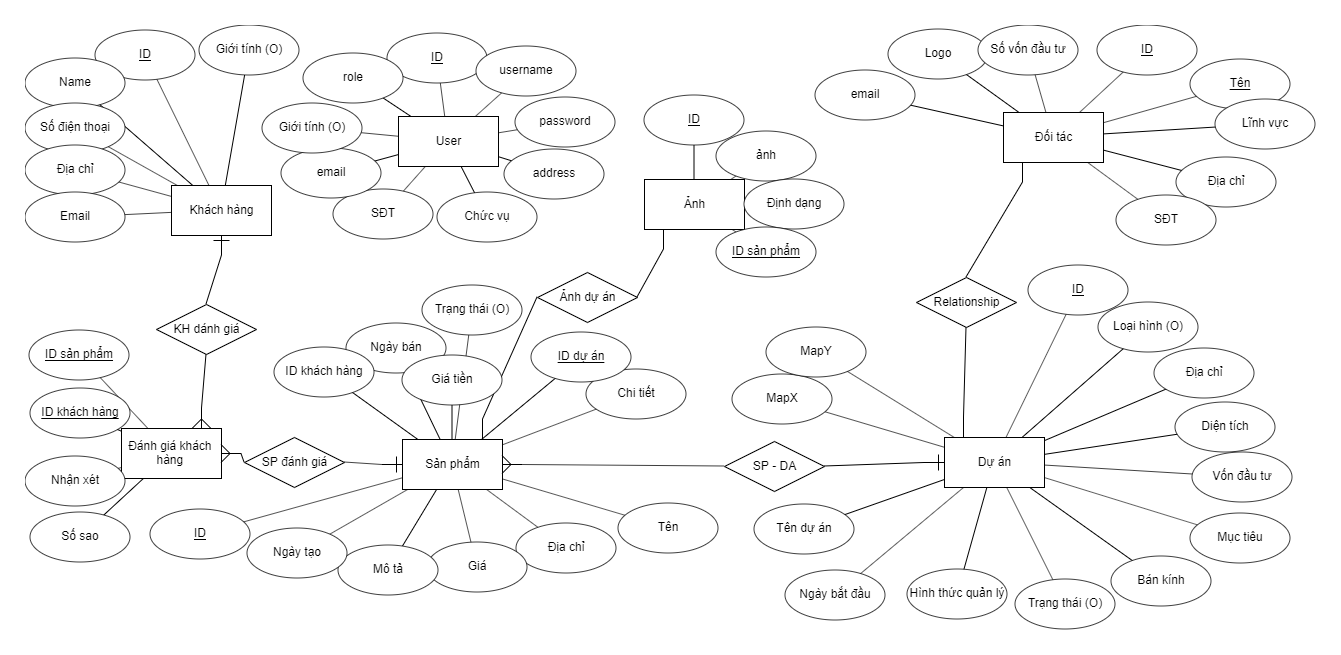
EduSys EduSys

+ Cần một máy cài SQL Server 2008+. Máy này cần hệ điều hành window xp trở lên

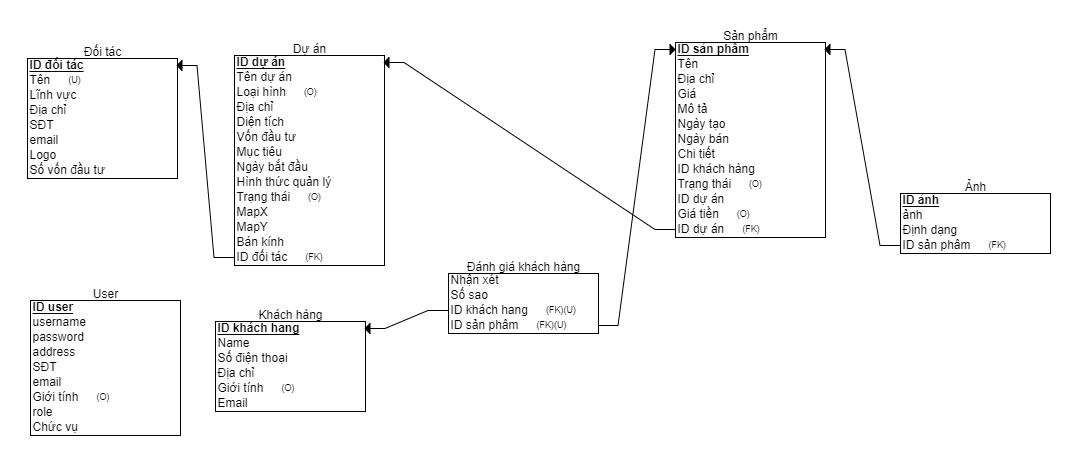
+ Các máy nhân viên cài phần mềm edusys. Các máy nhân viên cần JDK 1.8+ với hệ điều hành bất ký

## Thiết kế CSDL

### Sơ đồ quan hệ thực thể



**2.2.1.2 ERD Diagram level 2**

****

### Thiết kế chi tiết các thực thể

1 bảng NguoiDung lưu thông tin người sử dụng phần mềm có cấu trúc như sau

NguoiDung

|  |  |  |  |
| --- | --- | --- | --- |
| Tên cột | Kiểu | Ràng buộc | Ghi chú |
| IdNguoiDung | int identity | PK, not null | ID người dùng |
| TenNguoiDung | Nvarchar(50) | Not null | Tên người dùng |
| TenDangNhap | Varchar(50) | Not null | Tên đăng nhập |
| Password | Varchar(50) | Not null | Mật khẩu đăng nhập |
| DiaChi | Nvarchar(100) | Not null | Địa chỉ người dùng |
| Sdt | Varchar(15) | Not null | Số điện thoại |
| Email | Varchar(30) |  | Địa chỉ email |
| GioiTinh | Bit | Not null | Giới tính |
| ChucVu | Nvarchar(50) | Not null | Giám đốc hoặc nhân viên |
| Role | Int | Not null | Phân quyền |

2 Bảng KhachHang lưu thông tin khách hàng có cấu trúc như sau

KhachHang

|  |  |  |  |
| --- | --- | --- | --- |
| Tên cột | Kiểu | Ràng buộc | Ghi chú |
| IdKhachHang | int identity | PK, not null | ID khách hàng |
| TenKhachHang | nvarchar(50) | Not null | Tên khách hàng |
| GioiTinh | bit |  | Giới tính |
| Sdt | Varchar(20) | Not null | Số điện thoại |
| Email | Nvarchar(50) |  | Địa chỉ Email |
| DiaChi | Nvarchar(100) | Not null | Địa chỉ nhà |

3 Bảng DanhGia lưu đánh giá của khách hàng có cấu trúc như sau

DanhGia

|  |  |  |  |
| --- | --- | --- | --- |
| Tên cột | Kiểu | Ràng buộc | Ghi chú |
| IdKhachHang | int | Not null | ID khách hàng |
| SoSao | int |  |  |
| IdSanPham | int |  |  |
| NhanXet | nvarchar(100), |  | Đánh giá khách hàng |

4 Bảng DoiTac lưu thông tin của đối tác có cấu trúc như sau

DoiTac

|  |  |  |  |
| --- | --- | --- | --- |
| Tên cột | Kiểu | Ràng buộc | Ghi chú |
| IdDoiTac | int identity(0,1) | Pk, not null | ID đối tác |
| TenDoiTac | nvarchar(50) | Not null | Tên đối tác |
| LinhVuc | nvarchar(50) |  | Lĩnh vực bất động sản |
| DiaChi | nvarchar(100) | Not null | Địa chỉ |
| Sdt | varchar(20) | Not null | Số điện thoại |
| Email | varchar(50) |  | Địa chỉ Email |
| Logo | varchar(50) |  | Hình ảnh Logo |
| SoVonDaDauTu | float |  | Số vốn đã đầu tư |

5 Bảng dự án lưu thông tin dự án có cấu trúc như sau

DuAn

|  |  |  |  |
| --- | --- | --- | --- |
| Tên cột | Kiểu | Ràng buộc | Ghi chú |
| IdDuAn | int identity(0,1) | Pk, not null | ID dự án |
| TenDuAn | nvarchar(50) | Not null | Tên dự án |
| LoaiHinh | nvarchar(30) | Not null | Loại hình |
| DiaChi | nvarchar(50) | Not null | Địa chỉ dự án |
| DienTich | float | Not null | Diện tích dự án |
| ChiPhi | float | Not null | Chi phí dự án |
| NgayBatDau | date |  | Ngày bắt đầu |
| NgayKetThuc | date |  | Ngày kết thúc |
| HinhThucQuanLi | nvarchar(50) | Not null | Hình thức quản lý |
| HinhThucDauTu | nvarchar(50) | Not null | Hình thức đầu tư |
| IdDoiTac | int | Not null | ID đối tác |
| TrangThai | nvarchar(50) |  | Trạng thái dự án |
| MapX | float |  | Tọa độ dự án |
| MapY | float |  | Tọa độ dự án |
| BanKinh | float |  | Bán kính dự án |

6 bảng SanPham lưu thông tin sản phẩm có cấu trúc như sau

SanPham

|  |  |  |  |
| --- | --- | --- | --- |
| Tên cột | Kiểu | Ràng buộc | Ghi chú |
| IdSanPham | int identity(0,1) | PK, not null | ID sản phẩm |
| TenSanPham | nvarchar(50) | Not null | Tên sản phẩm |
| IdDuAn | int | Not null | ID dự án |
| DiaChi | nvarchar(50) | Not null | Địa chỉ sản phẩm |
| DienTich | float | Not null | Diện tích sản phẩm |
| GiaTien | float |  | Giá tiền sản phẩm |
| MoTa | nvarchar(50) |  | Mô tả sản phẩm |
| NgayTao | date |  | Ngày tạo sản phẩm |
| NgayBan | date |  | Ngày bán sản phẩm |
| ChiTiet | nvarchar(50) |  | Chi tiết sản phẩm |
| trangThai | nvarchar(50) | Not null | Trạng thái sản phẩm |
| IdKhachHang | int |  | ID khách hàng |

7 Bảng hình ảnh lưu thông tin hình ảnh có cấu trúc như sau

HinhAnh

|  |  |  |  |
| --- | --- | --- | --- |
| Tên cột | Kiểu | Ràng buộc | Ghi chú |
| IdHinhAnh | int identity(0,1) | Pk, not null | ID hình ảnh |
| HinhAnh | varchar(50) |  | Hình ảnh |
| CheDo | nvarchar(20) |  | Chế độ |
| IdSanPham | int | Not null | ID sản phẩm |

8 Bảng QuanLiChiNhanh lưu thông tin Chi nhánh có cấu trúc như sau

QuanLiChiNhanh

|  |  |  |  |
| --- | --- | --- | --- |
| Tên cột | Kiểu | Ràng buộc | Ghi chú |
| IdChiNhanh | int identity(0,1) | Pk, not null | ID chi nhánh |
| TenCN | nvarchar(50) | Not null | Tên chi nhánh |
| DiaChi | nvarchar(100) | Not null | Địa chỉ chi nhánh |
| Sdt | varchar(20) | Not null | Số điện thoại |
| Email | varchar(50) |  | Địa chỉ email |
| SoNV | int |  | số nhân viên |
| SoDuAnQL | int |  | Số dự án mà chi nhanh đang quản lí |

## Thiết kế giao diện

### Sơ đồ tổ chức giao diện

Giao diện chính là một cửa sổ chứa menu chính và toolbar. Thông qua đó để đi đến các giao diện thành viên để thực hiện các chức năng trong hệ thống.

Theo yêu cầu thì mọi nhân viên phải đăng nhập trước khi sử dụng ứng dụng nên form đăng nhập xuất hiện trước để yêu cầu đăng nhập. Ngoài ra mỗi ứng dụng trong thời gian khởi động cần có một màn hình chào cùng với thanh tiến trình để người có cảm giác ứng dụng đang khởi động.

MÀN HÌNH CHÀO

TỔNG HỢP PHÂN TÍCH

TỔNG HỢP THỐNG KÊ

QUẢN LÝ CHI NHÁNH

QUẢN LÝ ĐỐI TÁC

QUẢN LÝ DỰ ÁN

QUẢN LÝ SẢN PHẨM

QUẢN LÝ NGƯỜI DÙNG

HƯỚNG DẪN

ĐĂNG XUẤT

GIỚI THIỆU

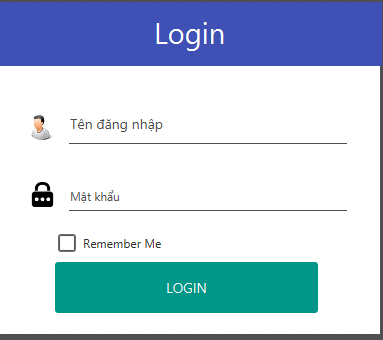
ĐỔI MẬT KHẨU

ĐĂNG NHẬP

GIAO DIỆN CHÍNH

### Thiết kế giao diện cho các chức năng nghiệp vụ

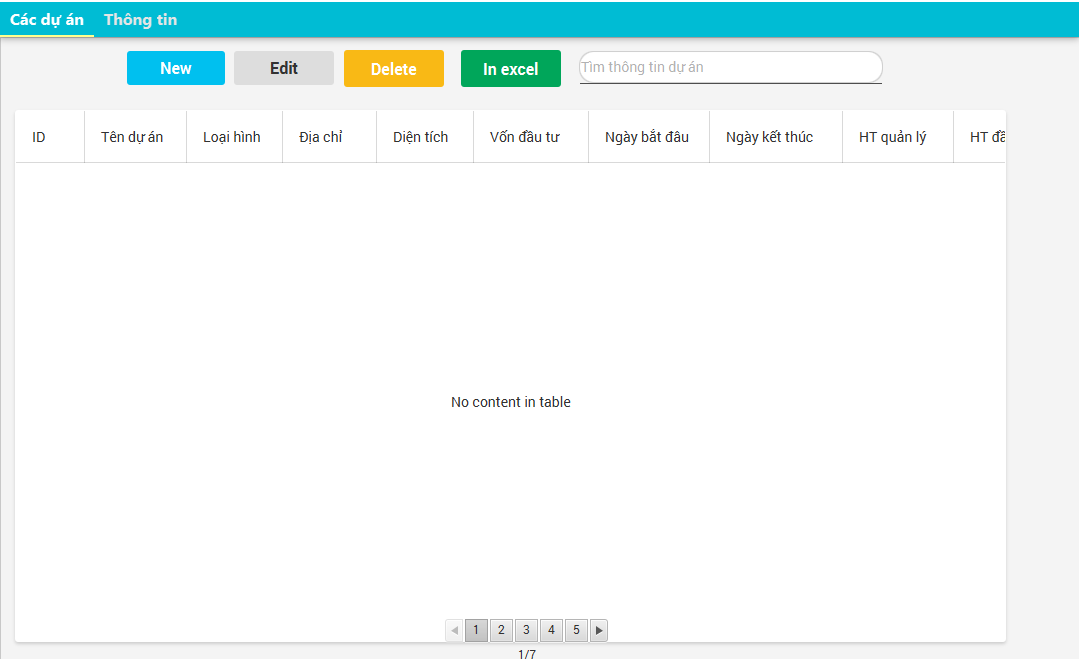
#### Thiết kế Form Login

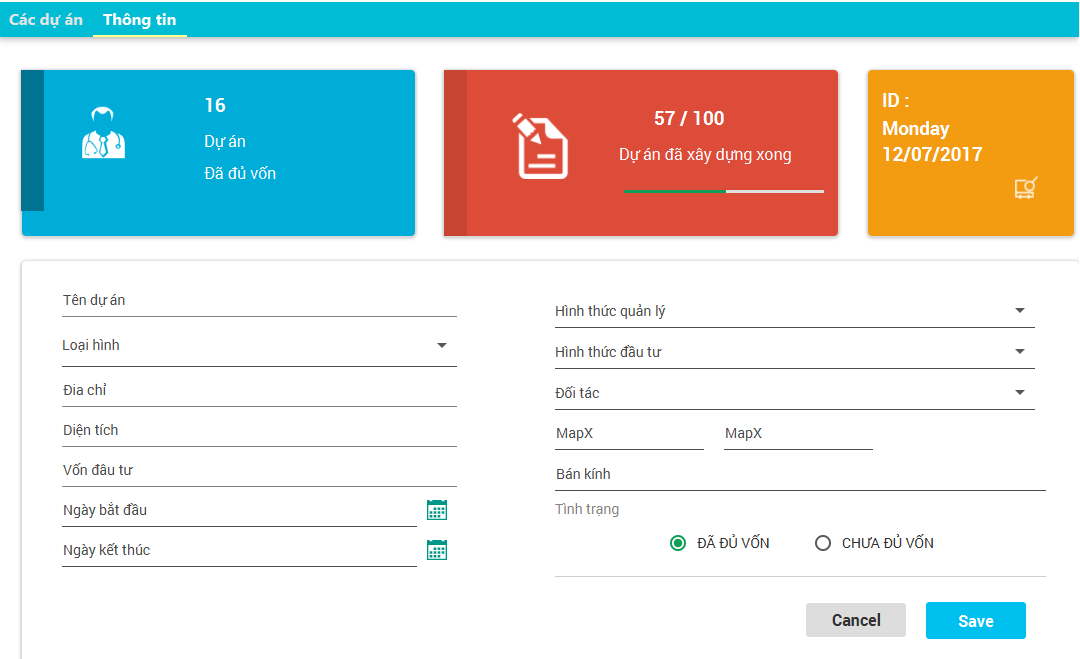


Các thành phần giao diện

|  |  |  |  |
| --- | --- | --- | --- |
|  | Componel | Kiểu | Thuộc tính |
| 1 | lbllogin | JLabel | Text:Login  Font:system 29px  defaultCloseOperation:DO\_NOTHING |
| 2 | Imgdangnhap | ImageView |  |
| 3 | Imgmatkhau | imgaeView |  |
| 4 | emailTextField | JFXTextField | Text:Tên đăng nhập |
| 5 | passwordTextField | JFXTextField | Text:Mật khẩu |
| 6 | rememberMeCheckBox | JFXCheckBox | Text:Remember Me |
| 7 | btnLogin | JFX Buton | Text:L |

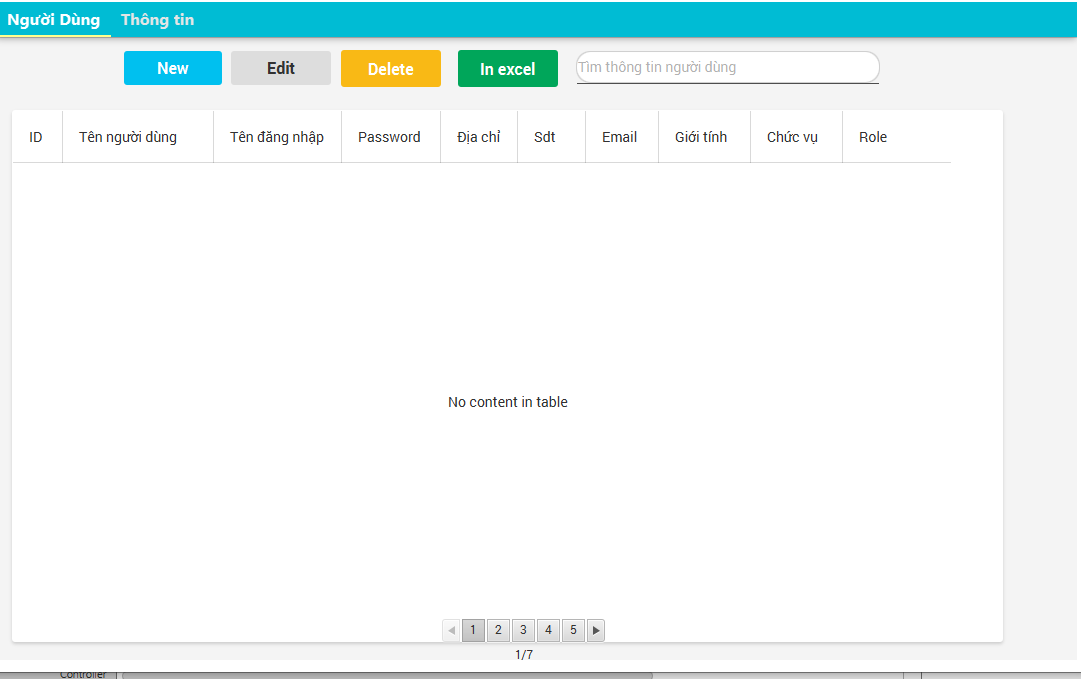
#### Thiết kế Form dự án

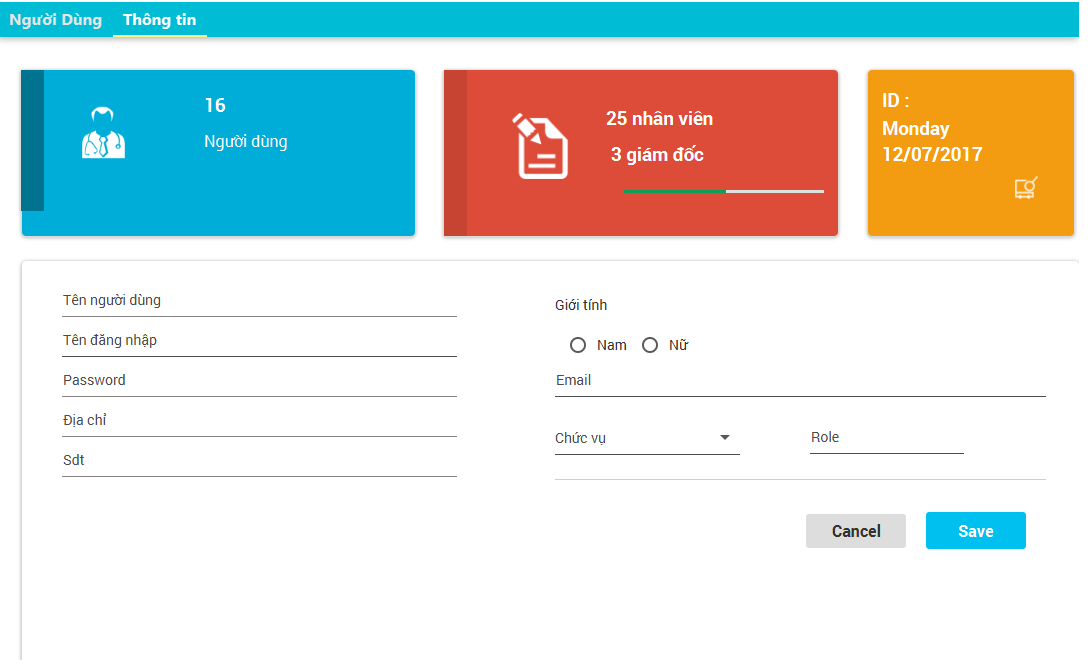




|  |  |  |  |
| --- | --- | --- | --- |
|  | Componel | Kiểu | Thuộc tính |
| 1 | btnAdd | JFXbuton | Text:New |
| 2 | btnEdit | JFXbuton | Text:Edit |
| 3 | btnDelete | JFXbuton | Text:Delete |
| 4 | btnin | JFXbuton | Text:in excel |
| 5 | clientSearchTextField | JFXTextField | Text:Tìm thông tin dự án |
| 6 | tableView | TableView | Model: như hình  Height: 25 |
| 7 | txtName | JFXTextField | Text: Tên dự án |
| 8 | comLoaiHinh | JFXComboBox | Text: Loại hình |
| 9 | txtDiaChi | JFXTextField | Text: Địa chỉ |
| 10 | txtDienTich | JFXTextField | Text: Diện tích |
| 11 | txtVonDauTu | JFXTextField | Text: Vốn đầu tư |
| 12 | dateStart | JPXDatePicker | Text: Ngày bắt đầu |
| 13 | dateEnd | JPXDatePicker | Text: Ngày kết thúc |
| 14 | comHTQuanLy | JFXComboBox | Text: Hình thức quản lý |
| 15 | comHTDauTu | JFXComboBox | Text: Hình thức đầu tư |
| 16 | comIDDoiTac | JFXComboBox | Text: Đối tác |
| 17 | txtMapX | JFXTextField | Text: MapX |
| 18 | txtMapY | JFXTextField | Text: MapY |
| 19 | txtBanKinh | JFXTextField | Text: Bán kính |
| 20 | rdDone | JFXRadioButon | Text: Đã đủ vốn |
| 21 | rdUndone | JFXRadioButon | Text: Chưa đủ vốn |
| 22 | btncancel | JFXbuton | Text:Cancel |
| 23 | btnsave | JFXbuton | Text:Save |

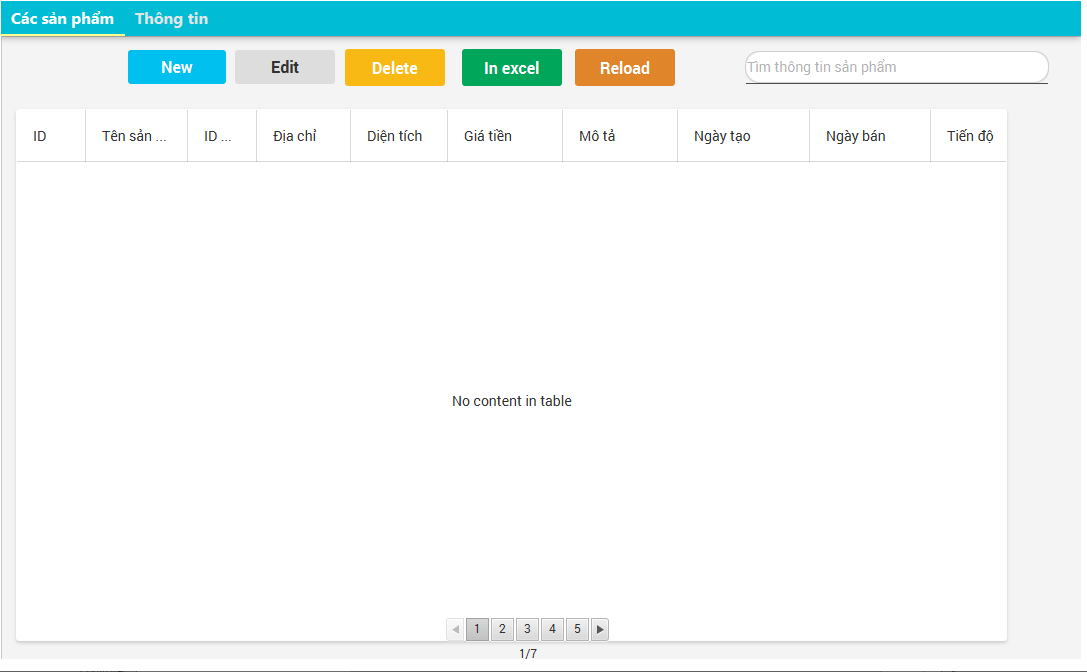
#### Thiết kế form người dùng

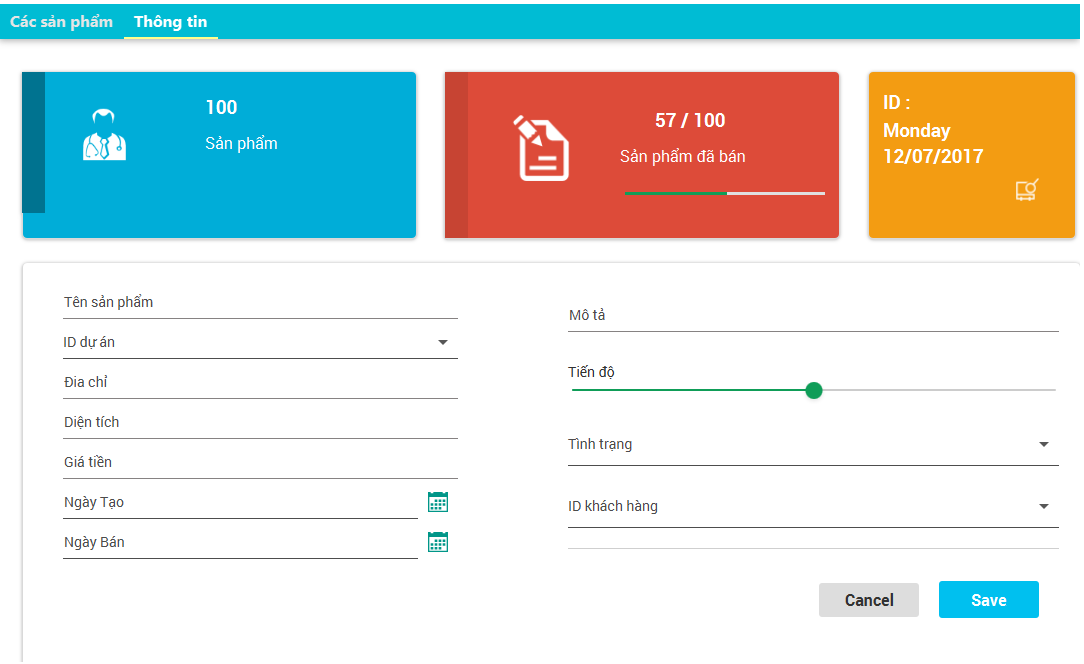




|  |  |  |  |
| --- | --- | --- | --- |
|  | Componel | Kiểu | Thuộc tính |
| 1 | btnAdd | JFXbuton | Text:New |
| 2 | btnEdit | JFXbuton | Text:Edit |
| 3 | btnDelete | JFXbuton | Text:Delete |
| 4 | btnin | JFXbuton | Text:in excel |
| 5 | clientSearchTextField | JFXTextField | Text:Tìm thông tin người dùng |
| 6 | tableView | TableView | Model: như hình  Height: 25 |
| 7 | txtName | JFXTextField | Text: Tên dự án |
| 8 | txttendangnhap | JFXTextField | Text: Tên đăng nhập |
| 9 | txtpassword | JFXTextField | Text: Password |
| 10 | txtDiachi | JFXTextField | Text: Địa chỉ |
| 11 | txtVonDauTu | JFXTextField | Text: Vốn đầu tư |
| 12 | txtSdt | JFXTextField | Text: Sdt |
| 13 | rdnam | JFXRadioButon | Text: nam |
| 14 | rdnu | JFXRadioButon | Text: nữ |
| 15 | txtEmail | JFXTextField | Text: Email |
| 16 | txtRole | JFXTextField | Text: Role |
| 17 | comChucvu | JFXComboBox | Text: Chức vụ |
| 18 | btncancel | JFXbuton | Text:Cancel |
| 19 | btnsave | JFXbuton | Text:Save |

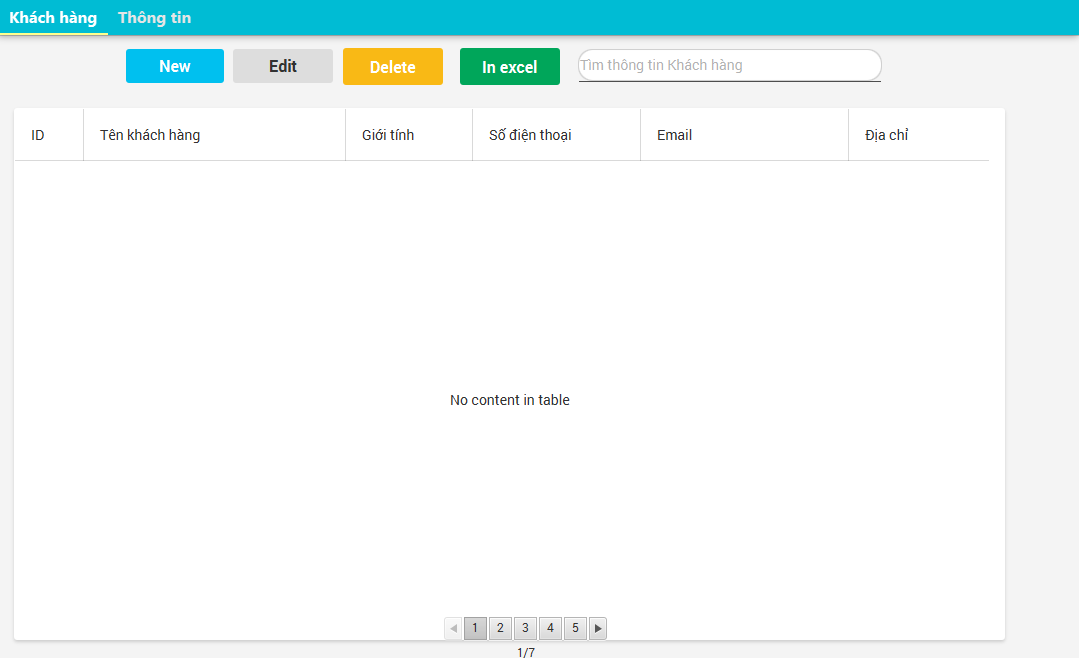
#### Thiết kế form sản phẩm

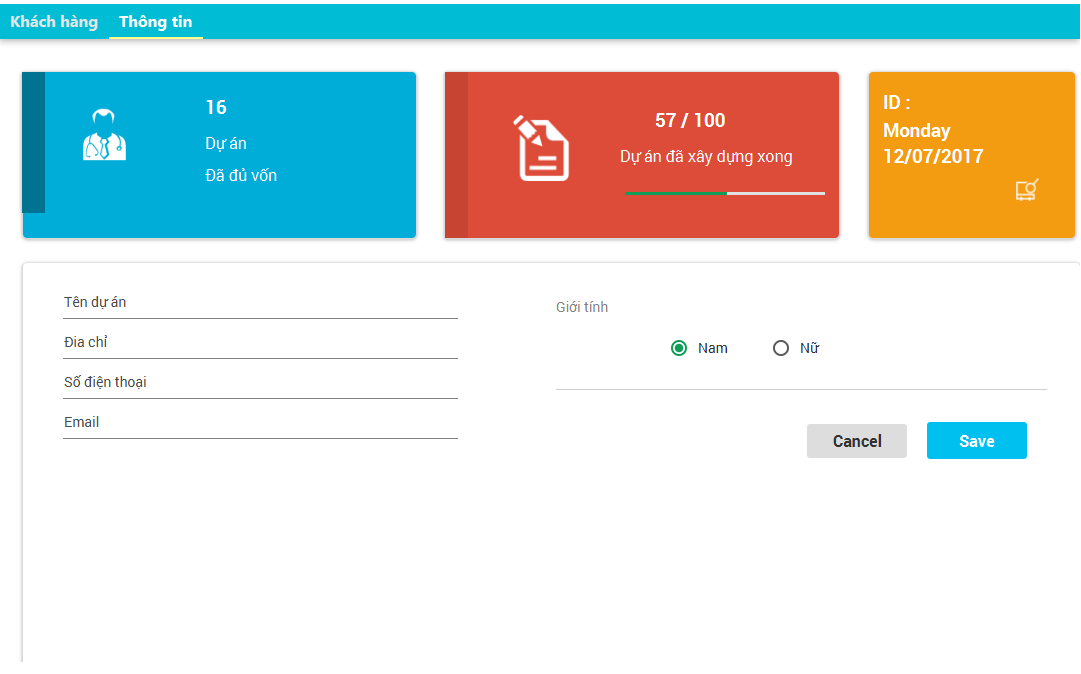




|  |  |  |  |
| --- | --- | --- | --- |
|  | Componel | Kiểu | Thuộc tính |
| 1 | btnAdd | JFXbuton | Text:New |
| 2 | btnEdit | JFXbuton | Text:Edit |
| 3 | btnDelete | JFXbuton | Text:Delete |
| 4 | btnin | JFXbuton | Text:in excel |
| 4 | btnreload | JFXbuton | Text:Reload |
| 5 | clientSearchTextField | JFXTextField | Text:Tìm thông tin sản phẩm |
| 6 | tableView | TableView | Model: như hình  Height: 25 |
| 7 | txtName | JFXTextField | Text: Tên sản phẩm |
| 8 | comIdDuAn | JFXComboBox | Text: Loại hình |
| 9 | txtDiaChi | JFXTextField | Text: Địa chỉ |
| 10 | txtDienTich | JFXTextField | Text: Diện tích |
| 11 | txtGiaTien | JFXTextField | Text: Giá tiền |
| 12 | dateStart | JPXDatePicker | Text: Ngày tạo |
| 13 | dateEnd | JPXDatePicker | Text: Ngày bán |
| 14 | txtMoTa | JFXTextField | Text: Mô tả |
| 15 |  | label | Text: Tiến độ |
| 16 | slideTienDo | JFXSlider |  |
| 17 | comTinhTrang | JFXComboBox | Text: Tình trạng |
| 18 | comIdKhachHang | JFXComboBox | Text: ID khách hàng |
| 19 | btncancel | JFXbuton | Text:Cancel |
| 20 | btnsave | JFXbuton | Text:Save |

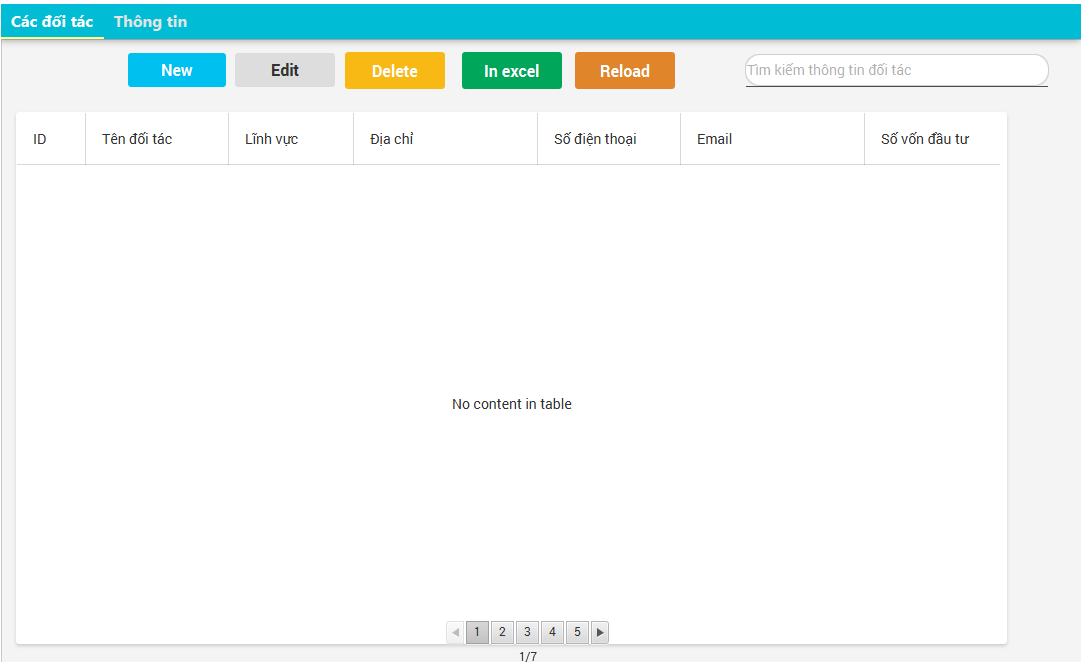
#### Thiết kế form khách hàng

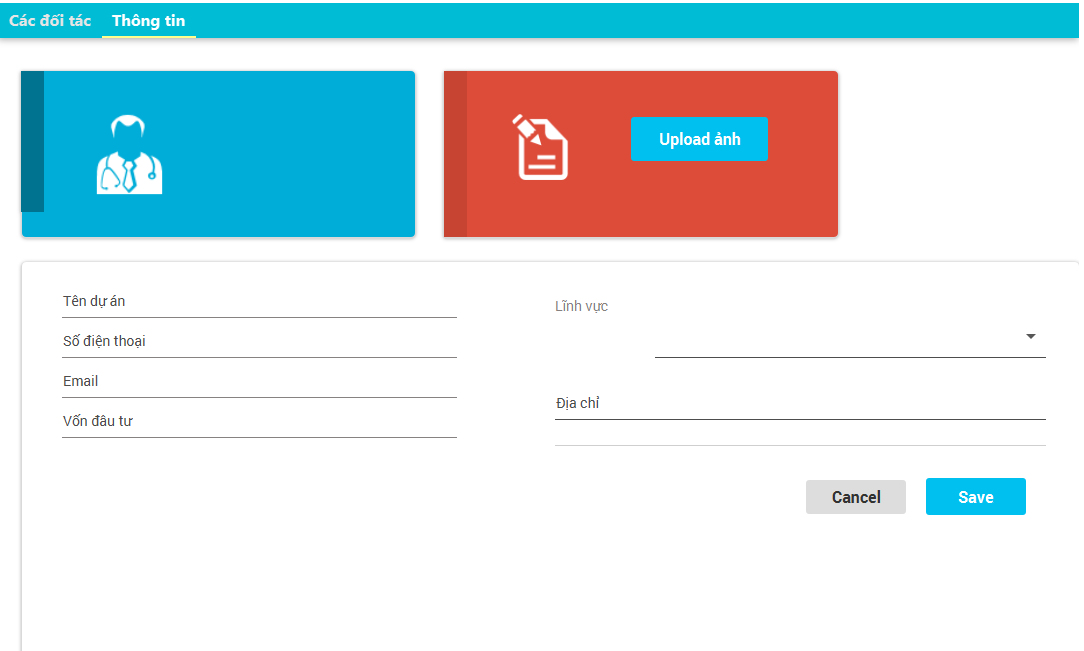




|  |  |  |  |
| --- | --- | --- | --- |
|  | Componel | Kiểu | Thuộc tính |
| 1 | btnAdd | JFXbuton | Text:New |
| 2 | btnEdit | JFXbuton | Text:Edit |
| 3 | btnDelete | JFXbuton | Text:Delete |
| 4 | btnin | JFXbuton | Text:in excel |
| 5 | clientSearchTextField | JFXTextField | Text:Tìm thông tin khách hàng |
| 6 | tableView | TableView | Model: như hình  Height: 25 |
| 7 | txtName | JFXTextField | Text: Tên sản phẩm |
| 8 | txtDiaChi | JFXTextField | Text: Địa chỉ |
| 9 | txtSdt | JFXTextField | Text: Sdt |
| 10 | txtEmail | JFXTextField | Text: Email |
| 11 | rdnam | JFXRadioButon | Text: nam |
| 12 | rdnu | JFXRadioButon | Text: nữ |
| 13 | btncancel | JFXbuton | Text:Cancel |
| 14 | btnsave | JFXbuton | Text:Save |

#### Thiết kế form đối tác





|  |  |  |  |
| --- | --- | --- | --- |
|  | Componel | Kiểu | Thuộc tính |
| 1 | btnAdd | JFXbuton | Text:New |
| 2 | btnEdit | JFXbuton | Text:Edit |
| 3 | btnDelete | JFXbuton | Text:Delete |
| 4 | btnin | JFXbuton | Text:in excel |
| 5 | clientSearchTextField | JFXTextField | Text:Tìm thông tin khách hàng |
| 6 | tableView | TableView | Model: như hình  Height: 25 |
| 7 | txtName | JFXTextField | Text: Tên đối tác |
| 8 | txtDiaChi | JFXTextField | Text: Địa chỉ |
| 9 | txtSdt | JFXTextField | Text: Sdt |
| 10 | txtEmail | JFXTextField | Text: Email |
| 11 | txtVonDauTu | JFXTextField | Text: Vốn đầu tư |
| 12 | comLinhVuc | JFXComboBox | Text: Lĩnh vực |
| 13 | btncancel | JFXbuton | Text:Cancel |
| 14 | btnsave | JFXbuton | Text:Save |
| 15 | Btnupload | JFXbuton | Text:Upload ảnh |

# Thực hiện viết mã

## Viết mã tạo CSDL

### Tạo CSDL

create database javafxbds;

use javafxbds;

create table Userr(

IdUser int identity(0,1) primary key not null,

NameUser nvarchar(50) not null,

UserName varchar(50) not null,

Password varchar(50) not null,

DiaChi nvarchar(100) not null,

Sdt varchar(15) not null,

Email varchar(30),

GioiTinh bit not null,

ChucVu nvarchar(50) not null,

Role int not null,

);

insert into Userr(NameUser,UserName,Password,DiaChi,Sdt,Email,GioiTinh,ChucVu,Role) values

(N'Nguyễn Văn Tân','VanTan','123',N'Quảng trị','0325772929','VanTan@gmail.com','true',N'Giám Đốc',1);

create table KhachHang(

IdKhachHang int identity(0,1) primary key not null,

TenKhachHang nvarchar(50) not null ,

GioiTinh bit,

DiaChi nvarchar(100) not null,

Sdt varchar(20) not null,

Email nvarchar(50),

);

insert into KhachHang(TenKhachHang,GioiTinh,DiaChi,Sdt,Email) values

(N'Nguyễn Văn A','true',N'Quảng trị','0327772999','Vana@gmail.com'),

(N'Nguyễn Văn B','true',N'Quảng Bình','0566778998','Basd@gmail.com');

create table DoiTac(

IdDoiTac int identity(0,1) primary key not null,

TenDoiTac nvarchar(30) not null,

LinhVuc nvarchar(30) ,

DiaChi nvarchar(60) not null,

Sdt varchar(15) not null,

Email varchar(15),

Logo Varbinary(max) ,

SoVonDaDauTu float,

)

insert into DoiTac(TenDoiTac,LinhVuc,DiaChi,Sdt,Email,SoVonDaDauTu) values

(N'Cty ABC',N'Xây Dựng',N'Quảng Nam','0325111222','abc@gmail.com',455465),

(N'Cty BDS',N'Bất Động Sản',N'Quảng Ngãi','0326777988','dda@gmail.com',78826);

create table DuAn(

IdDuAn int identity(0,1) primary key not null,

TenDuAn nvarchar(50) not null,

LoaiHinh nvarchar(30) not null,

DiaChi nvarchar(50) not null,

DienTich float not null,

ChiPhi float not null,

NgayBatDau date,

NgayKetThuc date,

HinhThucQuanLi nvarchar(50) not null,

HinhThucDauTu nvarchar(50) not null,

IdDoiTac int not null,

TrangThai nvarchar(50) ,

MapX float,

MapY float,

BanKinh float,

constraint fk\_DuAn foreign key (IdDoiTac) references DoiTac(IdDoiTac),

)

insert into DuAn(TenDuAn,LoaiHinh,DiaChi,DienTich,ChiPhi,NgayBatDau,NgayKetThuc,HinhThucQuanLi,HinhThucDauTu,IdDoiTac,TrangThai,MapX,MapY,BanKinh) values

(N'Chung cư ABCD',N'Chung Cư',N'08 Hà văn Tính',112,550,'2017-05-02','',N'Chung Cư',N'góp vốn',0,N'Đang xây dựng',54545,6565,4545),

(N'Chung cư sa',N'Chung Cư',N'03 Hà văn Ngọc',4545,10289,'2019-01-02','',N'Chung Cư',N'góp vốn ',0,N'Đang xây dựng',85698,5454,87887);

create table SanPham(

IdSanPham int identity(0,1) primary key not null,

TenSanPham nvarchar(50) not null,

IdDuAn int not null,

DiaChi nvarchar(50) not null,

DienTich float not null,

GiaTien float ,

MoTa nvarchar(50),

NgayTao date ,

NgayBan date,

ChiTiet nvarchar(50),

trangThai nvarchar(50) not null,

IdKhachHang int ,

constraint fk\_SanPham foreign key (IdKhachHang) references KhachHang(IdKhachHang),

constraint fk\_DuAn\_SP foreign key (IdDuAn) references DuAn(IdDuAn),

)

insert into SanPham(TenSanPham,IdDuAn,DiaChi,DienTich,GiaTien,MoTa,NgayTao,NgayBan,ChiTiet,trangThai,IdKhachHang) values

(N'Phòng 101',0,N'08 Hà Văn Tính',50,510,N'đang xây dựng','2017-05-02','',N'phòng đầy đủ tiện nghi',N'chưa bán',''),

(N'Phòng 02',1,N'03 Hà Văn Ngọc',40,390,N'đang xây dựng','2019-03-02','',N'phòng đầy đủ tiện nghi',N'chưa bán','');

create table DanhGia(

IdKhachHang int not null,

NhanXet nvarchar(100),

SoSao int ,

IdSanPham int,

constraint fk\_danhgia foreign key (IdKhachHang) references KhachHang(IdKhachHang),

constraint fk\_danhgia2 foreign key (IdSanPham) references SanPham(IdSanPham) ,

unique (IdKhachHang,IdSanPham)

);

create table HinhAnh(

IdHinhAnh int identity(0,1) primary key not null,

HinhAnh varchar(50) ,

CheDo nvarchar(20) ,

IdSanPham int not null,

constraint fk\_HinhAnh foreign key (IdSanPham) references SanPham(IdSanPham),

)

### SQL truy vấn và thao tác

#### SQL đối với Bảng User

|  |  |
| --- | --- |
| Hành Động | Câu lệnh SQL |
| Thêm mới | insert into Userr(NameUser,UserName,Password,DiaChi,Sdt,Email,GioiTinh,ChucVu,Role) |
| Cập nhật theo ID | update Userr set NameUser =? ,UserName =?, Password =?, DiaChi = ?, Sdt =? , Email =?, GioiTinh = ?,ChucVu =? , Role=? Where IdUser = ? |
| Xóa theo ID | Delete from User where IdUser =? |
| Truy vấn tất cả | Select\*from User |
| Truy vấn theo ID | Select \* from User where IdUser =? |

#### SQL đối với Bảng khách hàng

|  |  |
| --- | --- |
| Hành Động | Câu lệnh SQL |
| Thêm mới | insert into KhachHang(TenKhachHang,GioiTinh,DiaChi,Sdt,Email) values |
| Cập nhật theo ID | update KhachHang set TenKhachHang =? , GioiTinh =?, DiaChi =?, DiaChi = ?, Sdt =? ,  Email =?, Where IdKhachHang = ? |
| Xóa theo ID | Delete from KhachHang where IdKhachHang =? |
| Truy vấn tất cả | Select\*from KhachHang |
| Truy vấn theo ID | Select \* from KhachHang where IdKhachHang =? |

#### SQL đối với bảng DoiTac

|  |  |
| --- | --- |
| Hành Động | Câu lệnh SQL |
| Thêm mới | insert into DoiTac(TenDoiTac,LinhVuc,DiaChi,Sdt,Email,SoVonDaDauTu) values |
| Cập nhật theo ID | update DoiTac set TenDoiTac =? , LinhVuc =?, DiaChi = ?, Sdt =? , Email =?, SoVonDaDauTu = ? Where IdDoiTac = ? |
| Xóa theo ID | Delete from DoiTac where IdDoiTac =? |
| Truy vấn tất cả | Select\*from DoiTac |
| Truy vấn theo ID | Select \* from DoiTac where IdDoiTac =? |

#### SQL đối với bảng DuAn

|  |  |
| --- | --- |
| Hành Động | Câu lệnh SQL |
| Thêm mới | insert into DuAn(TenDuAn,LoaiHinh,DiaChi,DienTich,ChiPhi,NgayBatDau,NgayKetThuc,  HinhThucQuanLi,HinhThucDauTu,IdDoiTac,TrangThai,MapX,MapY,BanKinh) values |
| Cập nhật theo ID | update DuAn set TenDuAn =? , LoaiHinh =?, DiaChi =?, DienTich = ?, ChiPhi =? , NgayBatDau =?, NgayKetThuc = ?, HinhThucQuanLi =? , HinhThucDauTu =? Where IdDuAn = ? |
| Xóa theo ID | Delete from DuAn where IdDuAn =? |
| Truy vấn tất cả | Select\*from DuAn |
| Truy vấn theo ID | Select \* from DuAn where IdDuAn =? |

#### SQL đối với bảng SanPham

|  |  |
| --- | --- |
| Hành Động | Câu lệnh SQL |
| Thêm mới | insert into SanPham(TenSanPham,IdDuAn,DiaChi,DienTich,GiaTien,MoTa,NgayTao,  NgayBan,ChiTiet,trangThai,IdKhachHang) values |
| Cập nhật theo ID | update SanPham set TenSanPham =? , IdDuAn =?, DienTich =?, DiaChi = ?, GiaTien =? ,  NgayBan =?, ChiTiet = ?, trangThai =? , IdKhachHang =? Where IdSanPham = ? |
| Xóa theo ID | Delete from SanPham where IdSanPham =? |
| Truy vấn tất cả | Select\*from SanPham |
| Truy vấn theo ID | Select \* from SanPham where IdSanPham =? |

#### SQL đối với bảng DanhGia

|  |  |
| --- | --- |
| Hành Động | Câu lệnh SQL |
| Thêm mới | insert into Userr(IdKhachHang, NhanXet, SoSao, IdSanPham) values |
| Cập nhật theo ID | update DanhGia set IdKhachHang =? , NhanXet =?, SoSao =?, IdSanPham = ?,  Where IdDanhGia = ? |
| Xóa theo ID | Delete from DanhGia where IdDanhGia =? |
| Truy vấn tất cả | Select\*from DanhGia |
| Truy vấn theo ID | Select \* from DanhGia where IdDanhGia =? |

#### SQL đối với bảng HinhAnh

|  |  |
| --- | --- |
| Hành Động | Câu lệnh SQL |
| Thêm mới | insert into HinhAnh ( HinhAnh, CheDo, IdSanPham) |
| Cập nhật theo ID | update HinhAnh set HinhAnh =? , CheDo =?, IdSanPham Where IdHinhAnh = ? |
| Xóa theo ID | Delete from HinhAnh where IdHinhAnh =? |
| Truy vấn tất cả | Select\*from HinhAnh |
| Truy vấn theo ID | Select \* from HinhAnh where IdHinhAnh =? |

### Các thủ tục lưu tổng hợp thống kê

#### Procedure DuAn

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROC [dbo].[sp\_ThongKeDA](@Year INT)

AS BEGIN

SELECT

TenDuAn DuAn,

COUNT(DISTINCT sp.IdSanPham) SoSP,

COUNT(dt.IdDoiTac) SoDT,

COUNT(kh.IdKhachHang) SoKH,

SUM(da.ChiPhi) DoanhThu

FROM DuAn da

JOIN SanPham sp ON da.IdDoiTac=sp.IdDuAn

JOIN DoiTac dt ON da.IdDoiTac=dt.IdDoiTac

Join KhachHang kh ON sp.IdKhachHang=kh.IdKhachHang

WHERE YEAR(NgayTao) = @Year

GROUP BY TenDuAn

END

#### Procedure DoanhThu

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROC [dbo].[sp\_ThongKeDoanhThu](@Year INT)

AS BEGIN

SELECT

YEAR(NgayTao) Nam,

COUNT(DISTINCT da.IdDuAn) SoDuAn,

COUNT(sp.IdSanPham) SoSP,

SUM(sp.GiaTien) DoanhThu,

MAX(sp.GiaTien) CaoNhat,

MIN(sp.GiaTien) ThapNhat,

AVG(sp.GiaTien) TrungBinh

FROM DuAn da

JOIN SanPham sp ON da.IdDuAn=sp.IdSanPham

WHERE YEAR(NgayTao) = @Year

GROUP BY NgayTao

END

#### Procedure DoanhThu

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE PROC [dbo].[sp\_ThongKeDoiTac]

AS BEGIN

SELECT

TenDoiTac DoiTac,

COUNT(DISTINCT da.IdDoiTac) SoDuAnDauTu, /\* số dự án đã đầu tư của đối tác \*/

SUM(dt.SoVonDaDauTu) SoVonDaDauTu

FROM DoiTac dt

JOIN DuAn da ON dt.IdDoiTac=da.IdDoiTac

GROUP BY TenDoiTac

END

## Lập trình JDBC

### Lớp hỗ trợ

#### Lớp tiện ích DB connector

**package** service;  
  
**import** java.sql.Connection;  
**import** java.sql.SQLException;  
  
**import static** java.sql.DriverManager.*getConnection*;  
  
**public class** DBConnector {  
 **public static** Connection Connection() **throws** SQLException {  
 Connection connection = *getConnection*(**"jdbc:sqlserver://localhost:1433;databaseName=javafxbds;"**, **"sa"**, **"songlong"**);  
 **return** connection;  
 }  
}

\*Kết nối JDBC\*

#### Lớp tiện ích Notification

**package** service;  
  
**import** javafx.geometry.Pos;  
**import** javafx.util.Duration;  
**import** org.controlsfx.control.Notifications;  
  
**public class** Notification {  
 **public void** notification(String title, String content, **int** type) {  
 Notifications notifications = Notifications.*create*().title(title).text(content).graphic(**null**)  
 .hideAfter(Duration.*seconds*(5)).position(Pos.***BOTTOM\_RIGHT***);  
 **if** (type == 0) {  
 notifications.showInformation();  
 } **else if** (type == 1) {  
 notifications.showError();  
 }  
 }  
}

\*hiện thị thông báo\*

#### Lớp tiện ích PaginatedList

**package** service;  
  
**import** java.util.ArrayList;  
**import** java.util.Collections;  
**import** java.util.List;  
  
**public class** PaginatedList<T> {  
 **public static final** Integer ***DEFAULT\_PAGE\_SIZE*** = 10;  
 **public** List<T> **list**;  
 **public** List<List<T>> **listOfPages**;  
 **public** Integer **pageSize** = ***DEFAULT\_PAGE\_SIZE***;  
 **public** Integer **currentPage** = 0;  
  
 **public** PaginatedList() {  
 }  
  
 **public** PaginatedList(List<T> list) {  
 **this**.**list** = list;  
 initPages();  
 }  
  
 **public** PaginatedList(List<T> list, **int** pageSize) {  
 **this**.**list** = list;  
 **this**.**pageSize** = pageSize;  
 initPages();  
 }  
  
 **public** List<T> getPage(**int** pageNumber) {  
 **if** (**listOfPages** == **null** ||  
 pageNumber > **listOfPages**.size() ||  
 pageNumber < 1) {  
 **return** Collections.*emptyList*();  
 }  
 **currentPage** = pageNumber;  
 List<T> page = **listOfPages**.get(--pageNumber);  
 **return** page;  
 }  
  
 **public void** sortPage() {  
 }  
  
 **public void** setPage(List<T> list, **int** pageNumber) {  
 List<List<T>> listOfPages = **this**.**listOfPages**;  
 **for** (**int** i = 1; i <= numberOfPages(); i++) {  
 **if** (i == pageNumber) {  
 listOfPages.set(--pageNumber, list);  
 **break**;  
 }  
 }  
 **this**.**listOfPages** = listOfPages;  
 restorePageToList();  
 }  
  
 **public int** numberOfPages() {  
 **if** (**listOfPages** == **null**) {  
 **return** 0;  
 }  
 **return listOfPages**.size();  
 }  
  
 **public** List<T> nextPage() {  
 List<T> page = getPage(++**currentPage**);  
 **return** page;  
 }  
  
 **public** List<T> previousPage() {  
 List<T> page = getPage(--**currentPage**);  
 **return** page;  
 }  
  
 **public void** initPages() {  
 **if** (**list** == **null** || **listOfPages** != **null**) {  
 **return**;  
 }  
  
 **if** (**pageSize** <= 0 || **pageSize** > **list**.size()) {  
 **pageSize** = **list**.size();  
 }  
  
 Integer numOfPages = (**int**) Math.*ceil*((**double**) **list**.size() / (**double**) **pageSize**);  
 **listOfPages** = **new** ArrayList<List<T>>(numOfPages);  
 **for** (**int** pageNum = 0; pageNum < numOfPages; ) {  
 **int** from = pageNum \* **pageSize**;  
 **int** to = Math.*min*(++pageNum \* **pageSize**, **list**.size());  
 **listOfPages**.add(**list**.subList(from, to));  
 }  
 }  
  
 **public void** restorePageToList() {  
 **list**.clear();  
 **for** (List<T> listPage : **listOfPages** ) {  
 **for** (T model : listPage  
 ) {  
 **list**.add(model);  
 }  
 }  
 }  
  
 **public static void** main(String[] args) {  
 List<Integer> list = **new** ArrayList<Integer>();  
 **for** (**int** i = 1; i <= 62; i++) {  
 list.add(i);  
 }  
  
 PaginatedList<Integer> paginatedList = **new** PaginatedList<Integer>(list);  
 **while** (**true**) {  
 List<Integer> page = paginatedList.nextPage();  
 **if** (page == **null** || page.isEmpty()) {  
 **break**;  
 }  
 **for** (Integer value : page) {  
 System.***out***.println(value);  
 }  
 System.***out***.println(**"------------"**);  
 }  
 }  
}

\*đánh dấu số trang\*

### Model class - Các lớp mô tả dữ liệu

#### Model đánh giá khách hàng

**package** model;  
  
**public class** DanhGiaKhachHang {  
 **private int idKhachHang**;  
 **private int soSaoBinhChon**;  
 **private** String **nhanXet**;  
 **private int idSanPham**;  
  
 **public** DanhGiaKhachHang() {  
 }  
  
 **public** DanhGiaKhachHang(**int** idKhachHang, **int** soSaoBinhChon, String nhanXet, **int** idSanPham) {  
 **this**.**idKhachHang** = idKhachHang;  
 **this**.**soSaoBinhChon** = soSaoBinhChon;  
 **this**.**nhanXet** = nhanXet;  
 **this**.**idSanPham** = idSanPham;  
 }  
  
 **public int** getIdKhachHang() {  
 **return idKhachHang**;  
 }  
  
 **public void** setIdKhachHang(**int** idKhachHang) {  
 **this**.**idKhachHang** = idKhachHang;  
 }  
  
 **public int** getSoSaoBinhChon() {  
 **return soSaoBinhChon**;  
 }  
  
 **public void** setSoSaoBinhChon(**int** soSaoBinhChon) {  
 **this**.**soSaoBinhChon** = soSaoBinhChon;  
 }  
  
 **public** String getNhanXet() {  
 **return nhanXet**;  
 }  
  
 **public void** setNhanXet(String nhanXet) {  
 **this**.**nhanXet** = nhanXet;  
 }  
  
 **public int** getIdSanPham() {  
 **return idSanPham**;  
 }  
  
 **public void** setIdSanPham(**int** idSanPham) {  
 **this**.**idSanPham** = idSanPham;  
 }  
}

#### Model đối tác

**package** model;  
  
**import** java.io.InputStream;  
  
**public class** DoiTac {  
 **private int idDoiTac**;  
 **private** String **tenDoitac**;  
 **private** String **linhVuc**;  
 **private** String **diaChi**;  
 **private** String **sdt**;  
 **private** String **email**;  
 **private** InputStream **logo**;  
 **private** Double **soVonDaDauTu**; *// số vốn đã đầu tư* **public** DoiTac(**int** idDoiTac, String tenDoitac, String linhVuc, String diaChi, String sdt, String email, Double soVonDaDauTu) {  
 **this**.**idDoiTac** = idDoiTac;  
 **this**.**tenDoitac** = tenDoitac;  
 **this**.**linhVuc** = linhVuc;  
 **this**.**diaChi** = diaChi;  
 **this**.**sdt** = sdt;  
 **this**.**email** = email;  
 **this**.**soVonDaDauTu** = soVonDaDauTu;  
 }  
  
 **public** DoiTac() {  
 }  
  
 **public** DoiTac(**int** idDoiTac, String tenDoitac, String linhVuc, String diaChi, String sdt, String email, InputStream logo, Double soVonDaDauTu) {  
 **this**.**idDoiTac** = idDoiTac;  
 **this**.**tenDoitac** = tenDoitac;  
 **this**.**linhVuc** = linhVuc;  
 **this**.**diaChi** = diaChi;  
 **this**.**sdt** = sdt;  
 **this**.**email** = email;  
 **this**.**logo** = logo;  
 **this**.**soVonDaDauTu** = soVonDaDauTu;  
 }  
  
 **public int** getIdDoiTac() {  
 **return idDoiTac**;  
 }  
  
 **public void** setIdDoiTac(**int** idDoiTac) {  
 **this**.**idDoiTac** = idDoiTac;  
 }  
  
 **public** String getTenDoitac() {  
 **return tenDoitac**;  
 }  
  
 **public void** setTenDoitac(String tenDoitac) {  
 **this**.**tenDoitac** = tenDoitac;  
 }  
  
 **public** String getLinhVuc() {  
 **return linhVuc**;  
 }  
  
 **public void** setLinhVuc(String linhVuc) {  
 **this**.**linhVuc** = linhVuc;  
 }  
  
 **public** String getDiaChi() {  
 **return diaChi**;  
 }  
  
 **public void** setDiaChi(String diaChi) {  
 **this**.**diaChi** = diaChi;  
 }  
  
 **public** String getSdt() {  
 **return sdt**;  
 }  
  
 **public void** setSdt(String sdt) {  
 **this**.**sdt** = sdt;  
 }  
  
 **public** String getEmail() {  
 **return email**;  
 }  
  
 **public void** setEmail(String email) {  
 **this**.**email** = email;  
 }  
  
 **public** InputStream getLogo() {  
 **return logo**;  
 }  
  
 **public void** setLogo(InputStream logo) {  
 **this**.**logo** = logo;  
 }  
  
 **public** Double getSoVonDaDauTu() {  
 **return soVonDaDauTu**;  
 }  
  
 **public void** setSoVonDaDauTu(Double soVonDaDauTu) {  
 **this**.**soVonDaDauTu** = soVonDaDauTu;  
 }  
}

#### Model image

**package** model;  
  
**import** java.io.InputStream;  
  
**public class** Image {  
 **private int idImager**;  
 **private** InputStream **image**;  
 **private** String **regimeImager**;  
 **private int idSanPham**;  
  
 **public** Image() {  
 }  
  
 **public** Image(**int** idImager, InputStream image, String regimeImager, **int** idSanPham) {  
 **this**.**idImager** = idImager;  
 **this**.**image** = image;  
 **this**.**regimeImager** = regimeImager;  
 **this**.**idSanPham** = idSanPham;  
 }  
  
 **public int** getIdImager() {  
 **return idImager**;  
 }  
  
 **public void** setIdImager(**int** idImager) {  
 **this**.**idImager** = idImager;  
 }  
  
 **public** InputStream getImage() {  
 **return image**;  
 }  
  
 **public void** setImage(InputStream image) {  
 **this**.**image** = image;  
 }  
  
 **public** String getRegimeImager() {  
 **return regimeImager**;  
 }  
  
 **public void** setRegimeImager(String regimeImager) {  
 **this**.**regimeImager** = regimeImager;  
 }  
  
 **public int** getIdSanPham() {  
 **return idSanPham**;  
 }  
  
 **public void** setIdSanPham(**int** idSanPham) {  
 **this**.**idSanPham** = idSanPham;  
 }  
}

#### Model Tổng hợp doanh thu

**package** model;  
  
**public class** TongHopDoanhThu {  
 **private int Nam**;  
 **private int SoDuAn**;  
 **private int SoSp**;  
 **private double TongDoanhThu**;  
 **private double DoanhThuCaoNhat**;  
 **private double DoanhThuThapNhat**;  
 **private double DoanhThuTB**; *// doah thu trung bình năm* **public** TongHopDoanhThu() {  
 }  
  
 **public** TongHopDoanhThu(**int** nam, **int** soDuAn, **int** soSp, **double** tongDoanhThu, **double** doanhThuCaoNhat, **double** doanhThuThapNhat, **double** doanhThuTB) {  
 **Nam** = nam;  
 **SoDuAn** = soDuAn;  
 **SoSp** = soSp;  
 **TongDoanhThu** = tongDoanhThu;  
 **DoanhThuCaoNhat** = doanhThuCaoNhat;  
 **DoanhThuThapNhat** = doanhThuThapNhat;  
 **DoanhThuTB** = doanhThuTB;  
 }  
  
 **public int** getNam() {  
 **return Nam**;  
 }  
  
 **public void** setNam(**int** nam) {  
 **Nam** = nam;  
 }  
  
 **public int** getSoDuAn() {  
 **return SoDuAn**;  
 }  
  
 **public void** setSoDuAn(**int** soDuAn) {  
 **SoDuAn** = soDuAn;  
 }  
  
 **public int** getSoSp() {  
 **return SoSp**;  
 }  
  
 **public void** setSoSp(**int** soSp) {  
 **SoSp** = soSp;  
 }  
  
 **public double** getTongDoanhThu() {  
 **return TongDoanhThu**;  
 }  
  
 **public void** setTongDoanhThu(**double** tongDoanhThu) {  
 **TongDoanhThu** = tongDoanhThu;  
 }  
  
 **public double** getDoanhThuCaoNhat() {  
 **return DoanhThuCaoNhat**;  
 }  
  
 **public void** setDoanhThuCaoNhat(**double** doanhThuCaoNhat) {  
 **DoanhThuCaoNhat** = doanhThuCaoNhat;  
 }  
  
 **public double** getDoanhThuThapNhat() {  
 **return DoanhThuThapNhat**;  
 }  
  
 **public void** setDoanhThuThapNhat(**double** doanhThuThapNhat) {  
 **DoanhThuThapNhat** = doanhThuThapNhat;  
 }  
  
 **public double** getDoanhThuTB() {  
 **return DoanhThuTB**;  
 }  
  
 **public void** setDoanhThuTB(**double** doanhThuTB) {  
 **DoanhThuTB** = doanhThuTB;  
 }  
}

#### Model tổng hợp đối tác

**package** model;  
  
**public class** TongHopDoiTac {  
 **private** String **tenDoitac**;  
 **private int SoDuAnDaDauTu**;  
 **private double SoVonDaDauTu**;  
  
 **public** TongHopDoiTac() {  
 }  
  
 **public** TongHopDoiTac(String tenDoitac, **int** soDuAnDaDauTu, **double** soVonDaDauTu) {  
 **this**.**tenDoitac** = tenDoitac;  
 **SoDuAnDaDauTu** = soDuAnDaDauTu;  
 **SoVonDaDauTu** = soVonDaDauTu;  
 }  
  
 **public** String getTenDoitac() {  
 **return tenDoitac**;  
 }  
  
 **public void** setTenDoitac(String tenDoitac) {  
 **this**.**tenDoitac** = tenDoitac;  
 }  
  
 **public int** getSoDuAnDaDauTu() {  
 **return SoDuAnDaDauTu**;  
 }  
  
 **public void** setSoDuAnDaDauTu(**int** soDuAnDaDauTu) {  
 **SoDuAnDaDauTu** = soDuAnDaDauTu;  
 }  
  
 **public double** getSoVonDaDauTu() {  
 **return SoVonDaDauTu**;  
 }  
  
 **public void** setSoVonDaDauTu(**double** soVonDaDauTu) {  
 **SoVonDaDauTu** = soVonDaDauTu;  
 }  
}

#### Model tổng hợp dự án

**package** model;  
  
**public class** TongHopDuAn {  
 **private int Nam**;  
 **private** String **tenDuAn**;  
 **private int SoSP**;  
 **private int SoNhaDauTu**;  
 **private int SoKH**;  
 **private** Double **DoanhThu**;  
  
 **public** TongHopDuAn() {  
 }  
  
 **public** TongHopDuAn(**int** nam, String tenDuAn, **int** soSP, **int** soNhaDauTu, **int** soKH, Double doanhThu) {  
 **Nam** = nam;  
 **this**.**tenDuAn** = tenDuAn;  
 **SoSP** = soSP;  
 **SoNhaDauTu** = soNhaDauTu;  
 **SoKH** = soKH;  
 **DoanhThu** = doanhThu;  
 }  
  
 **public int** getNam() {  
 **return Nam**;  
 }  
  
 **public void** setNam(**int** nam) {  
 **Nam** = nam;  
 }  
  
 **public** String getTenDuAn() {  
 **return tenDuAn**;  
 }  
  
 **public void** setTenDuAn(String tenDuAn) {  
 **this**.**tenDuAn** = tenDuAn;  
 }  
  
 **public int** getSoSP() {  
 **return SoSP**;  
 }  
  
 **public void** setSoSP(**int** soSP) {  
 **SoSP** = soSP;  
 }  
  
 **public int** getSoNhaDauTu() {  
 **return SoNhaDauTu**;  
 }  
  
 **public void** setSoNhaDauTu(**int** soNhaDauTu) {  
 **SoNhaDauTu** = soNhaDauTu;  
 }  
  
 **public int** getSoKH() {  
 **return SoKH**;  
 }  
  
 **public void** setSoKH(**int** soKH) {  
 **SoKH** = soKH;  
 }  
  
 **public** Double getDoanhThu() {  
 **return DoanhThu**;  
 }  
  
 **public void** setDoanhThu(Double doanhThu) {  
 **DoanhThu** = doanhThu;  
 }  
}

### DAO Class - Các lớp truy xuất dữ liệu

#### Lớp DAO làm việc với Bảng đánh giá khách hàng

**package** service;  
  
**import** model.DanhGiaKhachHang;  
  
**import** java.io.FileNotFoundException;  
**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** DanhGiaKhachHangService {  
 **private static** Connection *com*;  
  
 **static** {  
 **try** {  
 *com* = DBConnector.*Connection*();  
 } **catch** (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
  
 **private** Notification **notification** = **new** Notification();  
  
 **public** DanhGiaKhachHangService() **throws** SQLException {  
 }  
  
 **public static** DanhGiaKhachHang getDanhGiaKhachHangFromResultSet(ResultSet rs) **throws** SQLException {  
 DanhGiaKhachHang danhGiaKhachHang = **new** DanhGiaKhachHang();  
 danhGiaKhachHang.setIdKhachHang(rs.getInt(**"IdKhachHang"**));  
 danhGiaKhachHang.setNhanXet(rs.getString(**"NhanXet"**));  
 danhGiaKhachHang.setSoSaoBinhChon(rs.getInt(**"SoSao"**));  
 danhGiaKhachHang.setIdSanPham(rs.getInt(**"IdSanPham"**));  
 **return** danhGiaKhachHang;  
 }  
  
 **public static** DanhGiaKhachHang findByMaDanhGiaKhachHang(**int** maDanhGiaKhachHang) {  
 DanhGiaKhachHang danhGiaKhachHang = **new** DanhGiaKhachHang();  
 **try** {  
 ResultSet rs = *com*.createStatement().executeQuery(**"select** *\** **from DanhGia"**);  
 **boolean** isValid = **false**;  
 **while** (rs.next()) {  
 **if** (rs.getInt(**"IdKhachHang"**) == maDanhGiaKhachHang) {  
 isValid = **true**;  
 danhGiaKhachHang = *getDanhGiaKhachHangFromResultSet*(rs);  
 }  
 }  
 **if** (!isValid) {  
 **return** danhGiaKhachHang;  
 }  
 rs.getStatement().close();  
 } **catch** (SQLException e) {  
 e.printStackTrace();  
 }  
 **return** danhGiaKhachHang;  
 }  
  
 **public static** DanhGiaKhachHang findByMaKhachHangAndMaSanPham(**int** maKhachHang, **int** maSanPham) {  
 DanhGiaKhachHang danhGiaKhachHang = **new** DanhGiaKhachHang();  
 **try** {  
 ResultSet rs = *com*.createStatement().executeQuery(**"select** *\** **from DanhGia where IdKhachHang = "** + maKhachHang +  
 **"and IdSanPham = "** + maSanPham);  
 **boolean** isValid = **false**;  
 **while** (rs.next()) {  
 danhGiaKhachHang = *getDanhGiaKhachHangFromResultSet*(rs);  
 }  
 **if** (!isValid) {  
 **return** danhGiaKhachHang;  
 }  
 rs.getStatement().close();  
 } **catch** (SQLException e) {  
 e.printStackTrace();  
 }  
 **return** danhGiaKhachHang;  
 }  
  
 **public static** List<DanhGiaKhachHang> getAll() {  
 List<DanhGiaKhachHang> list = **new** ArrayList<>();  
 **try** {  
 ResultSet rs = *com*.createStatement().executeQuery(**"select** *\** **from DanhGia"**);  
 **boolean** isValid = **false**;  
 **while** (rs.next()) {  
 DanhGiaKhachHang danhGiaKhachHang;  
 danhGiaKhachHang = *getDanhGiaKhachHangFromResultSet*(rs);  
 list.add(danhGiaKhachHang);  
 isValid = **true**;  
 }  
 **if** (!isValid) {  
 **return** list;  
 }  
 rs.getStatement().close();  
 } **catch** (SQLException ex) {  
 ex.printStackTrace();  
 }  
 **return** list;  
 }  
  
 **public static** List<String> getAllID() {  
 List<String> list = **new** ArrayList<>();  
 **try** {  
 ResultSet rs = *com*.createStatement().executeQuery(**"select** *\** **from DanhGia"**);  
 **boolean** isValid = **false**;  
 **while** (rs.next()) {  
 String id = rs.getString(**"IdKhachHang"**);  
 list.add(id);  
 isValid = **true**;  
 }  
 **if** (!isValid) {  
 **return** list;  
 }  
 rs.getStatement().close();  
 } **catch** (SQLException ex) {  
 ex.printStackTrace();  
 }  
 **return** list;  
 }  
  
 **public static void** deleteByMaDanhGiaKhachHang(String maDanhGiaKhachHang) **throws** SQLException {  
 System.***out***.println(**"DELETE FROM DanhGia WHERE IdKhachHang = "** + maDanhGiaKhachHang);  
 *com*.createStatement().executeUpdate(**"DELETE FROM DanhGia WHERE IdKhachHang = "** + maDanhGiaKhachHang);  
 }  
  
 **public static void** save(DanhGiaKhachHang danhGiaKhachHang) **throws** SQLException, FileNotFoundException {  
 DanhGiaKhachHang danhGiaKhachHangExsit = *findByMaKhachHangAndMaSanPham*(danhGiaKhachHang.getIdKhachHang(), danhGiaKhachHang.getIdSanPham());  
 **if** (danhGiaKhachHangExsit.getIdKhachHang() != 0) {  
 PreparedStatement pst = *com*.prepareStatement(**"UPDATE DanhGia SET NhanXet = N'"** + danhGiaKhachHang.getNhanXet() +  
 **"', SoSao = "** + danhGiaKhachHang.getSoSaoBinhChon() +  
 **", IdSanPham = "** + danhGiaKhachHang.getIdSanPham() +  
 **" where IdKhachHang = "** + danhGiaKhachHang.getIdKhachHang());  
 pst.execute();  
 } **else** {  
 System.***out***.println(**"INSERT INTO DanhGia ( NhanXet, SoSao, IdKhachHang, IdSanPham) VALUES (N'"** +  
 danhGiaKhachHang.getNhanXet() + **"', "** +  
 danhGiaKhachHang.getSoSaoBinhChon() + **","** +  
 danhGiaKhachHang.getIdKhachHang() + **","** +  
 danhGiaKhachHang.getIdSanPham() + **");"**);  
 PreparedStatement pst = *com*.prepareStatement(  
 **"INSERT INTO DanhGia ( NhanXet, SoSao, IdKhachHang, IdSanPham) VALUES (N'"** +  
 danhGiaKhachHang.getNhanXet() + **"', "** +  
 danhGiaKhachHang.getSoSaoBinhChon() + **","** +  
 danhGiaKhachHang.getIdKhachHang() + **","** +  
 danhGiaKhachHang.getIdSanPham() + **");"**);  
 pst.execute();  
 }  
 }  
}

#### Lớp DAO làm việc với Bảng đối tác

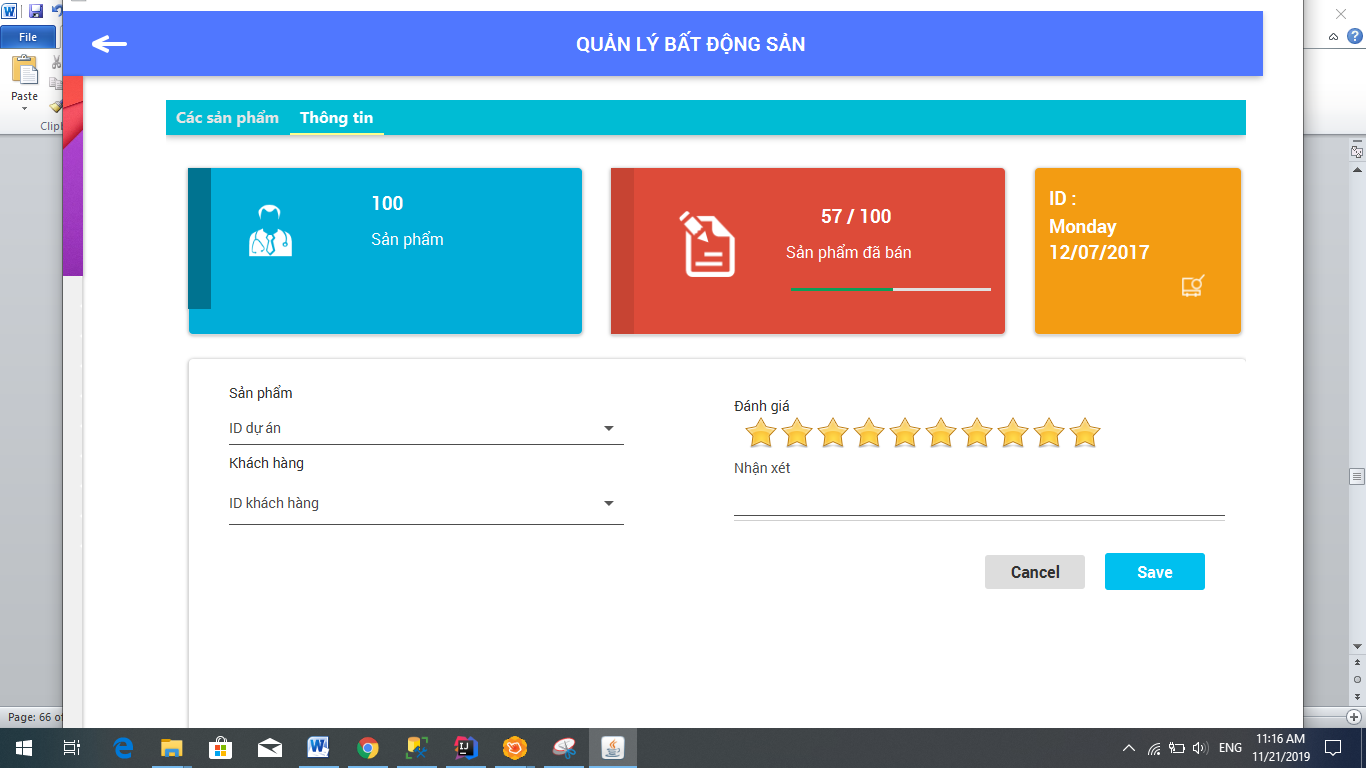
**package** service;  
  
  
**import** model.DoiTac;  
  
**import** java.io.\*;  
**import** java.sql.Connection;  
**import** java.sql.PreparedStatement;  
**import** java.sql.ResultSet;  
**import** java.sql.SQLException;  
**import** java.text.DecimalFormat;  
**import** java.util.ArrayList;  
**import** java.util.List;  
  
  
**public class** DoiTacService {  
 **private static** Connection *com*;  
  
 **static** {  
 **try** {  
 *com* = DBConnector.*Connection*();  
 } **catch** (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
  
 **private** Notification **notification** = **new** Notification();  
  
 **public static** DoiTac getDoiTacFromResultSet(ResultSet rs) **throws** SQLException {  
 DoiTac doiTac = **new** DoiTac();  
 **byte**[] bytes = rs.getBytes(**"Logo"**);  
 **if** (bytes != **null**) {  
 InputStream targetStream = **new** ByteArrayInputStream(bytes);  
 doiTac.setLogo(targetStream);  
 }  
 doiTac.setIdDoiTac(rs.getInt(**"IdDoiTac"**));  
 doiTac.setTenDoitac(rs.getString(**"TenDoiTac"**));  
 doiTac.setLinhVuc(rs.getString(**"LinhVuc"**));  
 doiTac.setDiaChi(rs.getString(**"DiaChi"**));  
 doiTac.setSdt(rs.getString(**"Sdt"**));  
 doiTac.setEmail(rs.getString(**"Email"**));  
 doiTac.setSoVonDaDauTu(rs.getDouble(**"SoVonDaDauTu"**));  
 **return** doiTac;  
 }  
  
  
 **public static** DoiTac findByMaDoiTac(**int** IdDoiTac) {  
 DoiTac doiTac = **new** DoiTac();  
 **try** {  
 ResultSet rs = com.createStatement().executeQuery(**"select** *\** **from doiTac"**);  
 **boolean** isValid = **false**;  
 **while** (rs.next()) {  
 **if** (rs.getInt(**"IdDoiTac"**) == IdDoiTac) {  
 isValid = **true**;  
 doiTac = getDoiTacFromResultSet(rs);  
 }  
 }  
 **if** (!isValid) {  
 **return** doiTac;  
 }  
 rs.getStatement().close();  
 } **catch** (SQLException e) {  
 e.printStackTrace();  
 }  
 **return** doiTac;  
 }  
  
 **public static** List<DoiTac> getAll() {  
 List<DoiTac> list = **new** ArrayList<>();  
 **try** {  
 ResultSet rs = *com*.createStatement().executeQuery(**"select** *\** **from doiTac"**);  
 **boolean** isValid = **false**;  
 **while** (rs.next()) {  
 DoiTac doiTac;  
 doiTac = *getDoiTacFromResultSet*(rs);  
 list.add(doiTac);  
 isValid = **true**;  
 }  
 **if** (!isValid) {  
 **return** list;  
 }  
 rs.getStatement().close();  
 } **catch** (SQLException ex) {  
 ex.printStackTrace();  
 }  
 **return** list;  
 }  
  
 **public static** List<String> getAllID() {  
 List<String> list = **new** ArrayList<>();  
 **try** {  
 ResultSet rs = com.createStatement().executeQuery(**"select** *\** **from doiTac"**);  
 **boolean** isValid = **false**;  
 **while** (rs.next()) {  
 String id = rs.getString(**"IdDoiTac"**);  
 list.add(id);  
 isValid = **true**;  
 }  
 **if** (!isValid) {  
 **return** list;  
 }  
 rs.getStatement().close();  
 } **catch** (SQLException ex) {  
 ex.printStackTrace();  
 }  
 **return** list;  
 }  
  
 **public static void** deleteByMaDoiTac(String IdDoiTac) **throws** SQLException {  
 com.createStatement().executeUpdate(**"DELETE FROM doiTac WHERE (IdDoiTac = '"** + IdDoiTac + **"')"**);  
 }  
  
  
 **public static void** save(DoiTac doiTac, File file) **throws** SQLException, IOException {  
 DoiTac doiTacExist = findByMaDoiTac(doiTac.getIdDoiTac());  
 **if** (doiTacExist.getIdDoiTac() != 0) {  
 **if** (file == **null**) {  
 PreparedStatement pst = com.prepareStatement(**"UPDATE doiTac SET tenDoiTac = ?,"** +  
 **"LinhVuc = ?,"** +  
 **"DiaChi = ?,"** +  
 **"Sdt = ?,"** +  
 **"Email = ?,"** +  
 **"SoVonDaDauTu = ? where IDDoiTac = ?"**);  
 pst.setString(1, doiTac.getTenDoitac());  
 pst.setString(2, doiTac.getLinhVuc());  
 pst.setString(3, doiTac.getDiaChi());  
 pst.setString(4, doiTac.getSdt());  
 pst.setString(5, doiTac.getEmail());  
 pst.setDouble(6, doiTac.getSoVonDaDauTu());  
 pst.setInt(7, doiTac.getIdDoiTac());  
 pst.execute();  
 } **else** {  
 PreparedStatement pst = com.prepareStatement(**"UPDATE doiTac SET tenDoiTac = ?,"** +  
 **"LinhVuc = ?,"** +  
 **"DiaChi = ?,"** +  
 **"Sdt = ?,"** +  
 **"Email = ?,"** +  
 **"SoVonDaDauTu = ?, "** +  
 **"logo = ?"** +  
 **" where IDDoiTac = ?"**);  
 pst.setString(1, doiTac.getTenDoitac());  
 pst.setString(2, doiTac.getLinhVuc());  
 pst.setString(3, doiTac.getDiaChi());  
 pst.setString(4, doiTac.getSdt());  
 pst.setString(5, doiTac.getEmail());  
 pst.setDouble(6, doiTac.getSoVonDaDauTu());  
 InputStream inputStream = **new** FileInputStream(file);  
 pst.setBinaryStream(7, inputStream, (**int**) file.length());  
 pst.setInt(8, doiTac.getIdDoiTac());  
 pst.execute();  
 }  
 } **else** {  
 **if** (file == **null**) {  
 DecimalFormat df = **new** DecimalFormat(**"###"**);  
 PreparedStatement pst = com.prepareStatement(  
 **"INSERT INTO doiTac ( TenDoiTac, LinhVuc, DiaChi, Sdt, Email, "** +  
 **" SoVonDaDauTu) VALUES (?,?,?,?,?,?)"**);  
 pst.setString(1, doiTac.getTenDoitac());  
 pst.setString(2, doiTac.getLinhVuc());  
 pst.setString(3, doiTac.getDiaChi());  
 pst.setString(4, doiTac.getSdt());  
 pst.setString(5, doiTac.getEmail());  
 pst.setDouble(6, doiTac.getSoVonDaDauTu());  
 pst.execute();  
 } **else** {  
 InputStream inputStream = **new** FileInputStream(file);  
 PreparedStatement pst = com.prepareStatement(  
 **"INSERT INTO doiTac ( TenDoiTac, LinhVuc, DiaChi, Sdt, Email, "** +  
 **" SoVonDaDauTu, logo) VALUES (?,?,?,?,?,?,?)"**);  
 pst.setString(1, doiTac.getTenDoitac());  
 pst.setString(2, doiTac.getLinhVuc());  
 pst.setString(3, doiTac.getDiaChi());  
 pst.setString(4, doiTac.getSdt());  
 pst.setString(5, doiTac.getEmail());  
 pst.setDouble(6, doiTac.getSoVonDaDauTu());  
 pst.setBinaryStream(7, inputStream, (**int**) file.length());  
 pst.execute();  
 inputStream.close();  
 }  
 }  
 }  
  
 **public static void** main(String[] args) {  
 List<String> ids = getAllID();  
 System.out.println(ids.size());  
 }  
}

#### Lớp dao làm việc với bảng Image

**package** service;  
  
  
**import** model.Image;  
  
**import** java.io.File;  
**import** java.io.FileNotFoundException;  
**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** ImageService {  
 **private static** Connection *com*;  
  
 **static** {  
 **try** {  
 *com* = DBConnector.*Connection*();  
 } **catch** (SQLException e) {  
 e.printStackTrace();  
 }  
 }  
  
 **private** Notification **notification** = **new** Notification();  
  
 **public static** Image getImageFromResultSet(ResultSet rs) **throws** SQLException {  
 Image image = **new** Image();  
 image.setIdImager(rs.getInt(**"IdHinhAnh"**));  
 image.setImage(rs.getBinaryStream(**"HinhAnh"**));  
 image.setRegimeImager(rs.getString(**"CheDo"**));  
 image.setIdSanPham(rs.getInt(**"IdSanPham"**));  
 **return** image;  
 }  
  
 **public static** Image findByMaImage(**int** IdImage) {  
 Image image = **new** Image();  
 **try** {  
 ResultSet rs = com.createStatement().executeQuery(**"select** *\** **from image"**);  
 **boolean** isValid = **false**;  
 **while** (rs.next()) {  
 **if** (rs.getInt(**"IdHinhAnh"**) == IdImage) {  
 isValid = **true**;  
 image = getImageFromResultSet(rs);  
 }  
 }  
 **if** (!isValid) {  
 **return** image;  
 }  
 rs.getStatement().close();  
 } **catch** (SQLException e) {  
 e.printStackTrace();  
 }  
 **return** image;  
 }  
  
 **public static** List<Image> getAll() {  
 List<Image> list = **new** ArrayList<>();  
 **try** {  
 ResultSet rs = com.createStatement().executeQuery(**"select** *\** **from image"**);  
 **boolean** isValid = **false**;  
 **while** (rs.next()) {  
 Image image;  
 image = getImageFromResultSet(rs);  
 list.add(image);  
 isValid = **true**;  
 }  
 **if** (!isValid) {  
 **return** list;  
 }  
 rs.getStatement().close();  
 } **catch** (SQLException ex) {  
 ex.printStackTrace();  
 }  
 **return** list;  
 }  
  
 **public static void** deleteByMaImage(String IdImage) **throws** SQLException {  
 *com*.createStatement().executeUpdate(**"DELETE FROM [javafx].[image] WHERE ('IdSanPham' = '"** + IdImage + **"')"**);  
 }  
  
 **public static void** save(Image image, File file) **throws** SQLException, FileNotFoundException {  
 Image imageExist = *findByMaImage*(image.getIdImager());  
 **if** (imageExist.getIdImager() != 0) {  
 PreparedStatement pst = *com*.prepareStatement(**"UPDATE [javafx].[image] SET 'HinhAnh' = '"** + image.getImage() +  
 **"', 'IdHinhAnh' = '"** + image.getIdImager() +  
 **"', 'HinhAnh' = '"** + image.getImage() +  
 **"', 'CheDo' = '"** + image.getRegimeImager() +  
 **"', 'IdSanPham' = '"** + image.getIdSanPham() +  
 **"')"**);  
 pst.execute();  
 } **else** {  
 PreparedStatement pst = *com*.prepareStatement(  
 **"INSERT INTO [javafx].[image] ('IdHinhAnh', 'HinhAnh', 'CheDo', 'IdSanPham') VALUES ('"** +  
 image.getIdImager() + **"', '"** +  
 image.getImage() + **"',' "** +  
 image.getRegimeImager() + **"',' "** +  
 image.getIdSanPham() + **"','"** +  
 **"')"**);  
 pst.execute();  
 }  
 }  
}

## Viết mã cho ứng dụng

### Xử lý Form đánh giá khách hàng



**package** controller;  
  
**import** com.jfoenix.controls.\*;  
**import** javafx.beans.value.ChangeListener;  
**import** javafx.beans.value.ObservableValue;  
**import** javafx.collections.FXCollections;  
**import** javafx.collections.ObservableList;  
**import** javafx.event.ActionEvent;  
**import** javafx.fxml.FXML;  
**import** javafx.fxml.Initializable;  
**import** javafx.scene.Node;  
**import** javafx.scene.control.\*;  
**import** javafx.scene.control.cell.PropertyValueFactory;  
**import** model.DanhGiaKhachHang;  
**import** org.controlsfx.control.Rating;  
**import** service.\*;  
  
**import** java.io.IOException;  
**import** java.net.URL;  
**import** java.sql.SQLException;  
**import** java.time.LocalDate;  
**import** java.time.format.DateTimeFormatter;  
**import** java.util.ArrayList;  
**import** java.util.Calendar;  
**import** java.util.List;  
**import** java.util.ResourceBundle;  
**import** java.util.stream.Collectors;  
  
**public class** DanhGiaController **implements** Initializable {  
 @FXML  
 JFXTextField **clientSearchTextFieldd**;  
  
 List<DanhGiaKhachHang> **danhGiaKhachHangList** = **new** ArrayList<>();  
 @FXML  
 **private** TableView<DanhGiaKhachHang> **tableView**;  
 @FXML  
 **private** TableColumn<DanhGiaKhachHang, String> **colIDKhachHang**;  
 @FXML  
 **private** TableColumn<DanhGiaKhachHang, String> **colSoSao**;  
 @FXML  
 **private** TableColumn<DanhGiaKhachHang, String> **colNhanXet**;  
 @FXML  
 **private** TableColumn<DanhGiaKhachHang, String> **colSanPham**;  
 **private** ObservableList<DanhGiaKhachHang> **danhGiaKhachHangs** = FXCollections.*observableArrayList*();  
 @FXML  
 **private** JFXButton **btnEdit**;  
 @FXML  
 **private** JFXButton **btnDelete**;  
 @FXML  
 Pagination **clientPagination**;  
 PaginatedList<DanhGiaKhachHang> **pagingList**;  
 @FXML  
 **private** JFXTextArea **txtNhanXet**;  
 @FXML  
 **private** JFXComboBox **comIdSanPham**;  
 @FXML  
 **private** JFXComboBox **comIdKhachHang**;  
 @FXML  
 JFXTabPane **tabView**;  
 @FXML  
 Rating **rating**;  
  
 **private** DanhGiaKhachHang **danhGiaKhachHang**;  
  
 Notification **notification** = **new** Notification();  
  
 **public** DanhGiaController() **throws** SQLException {  
 }  
  
 @FXML  
 **public void** filter() {  
 String search = **clientSearchTextFieldd**.getText();  
 setTableView();  
 List<DanhGiaKhachHang> result;  
 result = **danhGiaKhachHangs**.stream()  
 .filter(item -> String.*valueOf*(item.getIdSanPham()).contains(search)  
 || item.getNhanXet().contains(search)  
 )  
 .collect(Collectors.*toList*());  
 **danhGiaKhachHangs** = FXCollections.*observableList*(result);  
 **tableView**.setItems(**danhGiaKhachHangs**);  
 **if** (**clientSearchTextFieldd**.getText().isEmpty()) {  
 setTableView(0);  
 }  
 }  
  
 **private** Node setTableView(Integer integer) {  
 setTableView();  
 **int** fromIndex = integer \* 10;  
 **int** toIndex = Math.*min*(fromIndex + 10, **danhGiaKhachHangs**.size());  
 **tableView**.setItems(**danhGiaKhachHangs**);  
 **return tableView**;  
 }  
  
 **public** Node setTableView() {  
*// btnDelete.setDisable(true);  
// btnEdit.setDisable(true);* **tableView**.getItems().clear();  
 **danhGiaKhachHangList**.clear();  
 **danhGiaKhachHangList** = DanhGiaKhachHangService.*getAll*();  
 **pagingList** = **new** PaginatedList<>(**danhGiaKhachHangList**);  
 **clientPagination**.setPageCount(**pagingList**.**listOfPages**.size());  
 **danhGiaKhachHangs** = FXCollections.*observableList*(**danhGiaKhachHangList**);  
 **colIDKhachHang**.setCellValueFactory(**new** PropertyValueFactory<>(**"idKhachHang"**));  
 **colSoSao**.setCellValueFactory(**new** PropertyValueFactory<>(**"soSaoBinhChon"**));  
 **colNhanXet**.setCellValueFactory(**new** PropertyValueFactory<>(**"nhanXet"**));  
 **colSanPham**.setCellValueFactory(**new** PropertyValueFactory<>(**"idSanPham"**));  
 **tableView**.setItems(**danhGiaKhachHangs**);  
 **return tableView**;  
 }  
  
 **public void** selectItem() **throws** SQLException, IOException {  
 DanhGiaKhachHang NV = **tableView**.getSelectionModel().getSelectedItem();  
 **btnEdit**.setDisable(**false**);  
 **btnDelete**.setDisable(**false**);  
 }  
  
 **public void** creatNew(ActionEvent actionEvent) **throws** IOException, SQLException {  
 **this**.**danhGiaKhachHang** = **new** DanhGiaKhachHang();  
 changeStage();  
 }  
  
 **public void** changeStage() **throws** IOException, SQLException {  
 SingleSelectionModel<Tab> selectionModel = **tabView**.getSelectionModel();  
 selectionModel.select(1);  
 **if** (**this**.**danhGiaKhachHang**.getNhanXet() != **null**) {  
 **txtNhanXet**.setText(**this**.**danhGiaKhachHang**.getNhanXet());  
 }  
 **comIdSanPham**.getSelectionModel().select(**this**.**danhGiaKhachHang**.getIdSanPham());  
 **comIdKhachHang**.getSelectionModel().select(**this**.**danhGiaKhachHang**.getIdKhachHang());  
 **rating**.setRating(**this**.**danhGiaKhachHang**.getSoSaoBinhChon());  
 }  
  
 **public void** edit(ActionEvent actionEvent) **throws** IOException, SQLException {  
 **this**.**danhGiaKhachHang** = **tableView**.getSelectionModel().getSelectedItem();  
 changeStage();  
 }  
  
 **public void** deleteItem(ActionEvent actionEvent) **throws** SQLException {  
 DanhGiaKhachHang DanhGiaKhachHang = **tableView**.getSelectionModel().getSelectedItem();  
 DanhGiaKhachHangService.*deleteByMaDanhGiaKhachHang*(String.*valueOf*(DanhGiaKhachHang.getIdKhachHang()));  
 setTableView();  
 }  
  
 **public void** onSave(ActionEvent actionEvent) **throws** SQLException, IOException {  
 String sanPhamId = **"Chưa đặt"**;  
 **if** (**comIdSanPham**.getSelectionModel().selectedItemProperty().getValue() != **null**) {  
 sanPhamId = **comIdSanPham**.getSelectionModel().selectedItemProperty().getValue().toString();  
 }  
 String IDkhachHang = **""**;  
 **if** (**comIdKhachHang**.getSelectionModel().selectedItemProperty().getValue() != **null**) {  
 IDkhachHang = **comIdKhachHang**.getSelectionModel().selectedItemProperty().getValue().toString();  
 }  
 DanhGiaKhachHang danhGiaKhachHang = **new** DanhGiaKhachHang();  
 danhGiaKhachHang.setIdKhachHang(**this**.**danhGiaKhachHang**.getIdKhachHang());  
 danhGiaKhachHang.setSoSaoBinhChon((**int**) Math.*round*(**rating**.getRating()));  
 danhGiaKhachHang.setNhanXet(**txtNhanXet**.getText());  
 danhGiaKhachHang.setIdSanPham(Integer.*valueOf*(sanPhamId));  
 **if** (!IDkhachHang.equals(**""**)) {  
 danhGiaKhachHang.setIdKhachHang(Integer.*parseInt*(IDkhachHang));  
 }  
 DanhGiaKhachHangService.*save*(danhGiaKhachHang);  
 **notification**.notification(**"Save thành công"**, **"Đã lưu vào database"**, 0);  
 setTableView();  
 }  
  
 **public void** printIntoExcel() **throws** IOException {  
*// excel.danhGiaKhachHang = this.DanhGiaKhachHangList;  
// excel.printDanhGiaKhachHangToExcel();* }  
  
 @Override  
 **public void** initialize(URL location, ResourceBundle resources) {  
 setTableView();  
 **clientPagination**.setPageFactory(**this**::setTableView);  
 List<String> listProjectId = ProductService.*getAllID*();  
 ObservableList<String> doiTacData = FXCollections.*observableArrayList*(listProjectId);  
 **comIdSanPham**.setItems(doiTacData);  
 List<String> listKhachHangId = KhachHangService.*getAllID*();  
 ObservableList<String> khachHangdata = FXCollections.*observableArrayList*(listKhachHangId);  
 **comIdKhachHang**.setItems(khachHangdata);  
 }  
}

### Xử lý Form đối tác

**package** controller;  
  
**import** com.jfoenix.controls.\*;  
**import** javafx.beans.value.ChangeListener;  
**import** javafx.beans.value.ObservableValue;  
**import** javafx.collections.FXCollections;  
**import** javafx.collections.ObservableList;  
**import** javafx.event.ActionEvent;  
**import** javafx.fxml.FXML;  
**import** javafx.fxml.Initializable;  
**import** javafx.scene.Node;  
**import** javafx.scene.control.\*;  
**import** javafx.scene.control.cell.PropertyValueFactory;  
**import** javafx.scene.image.Image;  
**import** javafx.scene.image.ImageView;  
**import** javafx.stage.FileChooser;  
**import** javafx.stage.Stage;  
**import** model.DoiTac;  
**import** service.DoiTacService;  
**import** service.Notification;  
**import** service.PaginatedList;  
  
**import** java.io.\*;  
**import** java.net.URL;  
**import** java.sql.SQLException;  
**import** java.util.ArrayList;  
**import** java.util.List;  
**import** java.util.ResourceBundle;  
**import** java.util.regex.Matcher;  
**import** java.util.regex.Pattern;  
**import** java.util.stream.Collectors;  
  
**public class** DoiTacController **implements** Initializable {  
 @FXML  
 JFXTextField **clientSearchTextField**;  
 @FXML  
 JFXTextField **txtName**;  
 @FXML  
 JFXTextField **txtDiaChi**;  
 @FXML  
 JFXTextField **txtVonDauTu**;  
 @FXML  
 JFXTextField **txtSDT**;  
 @FXML  
 JFXTextField **txtEmail**;  
 @FXML  
 JFXComboBox **comLinhVuc**;  
  
 **public** DoiTac **doiTac** = **new** DoiTac();  
 List<DoiTac> **doiTacList** = **new** ArrayList<>();  
 @FXML  
 **private** TableView<DoiTac> **tableView**;  
 @FXML  
 **private** TableColumn<DoiTac, String> **colID**;  
 @FXML  
 **private** TableColumn<DoiTac, String> **colName**;  
 @FXML  
 **private** TableColumn<DoiTac, String> **colDiaChi**;  
 @FXML  
 **private** TableColumn<DoiTac, String> **colLinhVuc**;  
 @FXML  
 **private** TableColumn<DoiTac, String> **colSDT**;  
 @FXML  
 **private** TableColumn<DoiTac, String> **colEmail**;  
 @FXML  
 **private** TableColumn<DoiTac, String> **colSoVonDauTu**;  
 **private** ObservableList<DoiTac> **doiTacs** = FXCollections.*observableArrayList*();  
 @FXML  
 **private** JFXButton **btnEdit**;  
 @FXML  
 **private** JFXButton **btnDelete**;  
 @FXML  
 **private** JFXButton **btnAdd**;  
 @FXML  
 Pagination **clientPagination**;  
 PaginatedList<DoiTac> **pagingList**;  
 @FXML  
 JFXTabPane **tabView**;  
 Notification **notification** = **new** Notification();  
  
 InputStream **inputStream**;  
 File **file**;  
 **final** FileChooser **fileChooser** = **new** FileChooser();  
 @FXML  
 ImageView **imageView**;  
 String **EMAIL\_REGEX** = **"^(.+)@(.+)$"**;  
 String **PHONE\_REGEX** = **"^[0]+[9,12,16,18,19]+[0-9]{8}$"**;  
 Pattern **emailPattern** = Pattern.*compile*(**EMAIL\_REGEX**);  
 Pattern **phonePattern** = Pattern.*compile*(**PHONE\_REGEX**);  
 Matcher **matcher**;  
  
 **public** DoiTacController() **throws** SQLException {  
 }  
  
*// public void getNhanVien(NhanVien nhanVien) {  
// this.nhanVien = nhanVien;  
// }* @FXML  
 **public void** filter() {  
 String search = **clientSearchTextField**.getText();  
 System.***out***.println(search);  
 setTableView();  
 List<DoiTac> result;  
 result = **doiTacs**.stream()  
 .filter(item -> String.*valueOf*(item.getIdDoiTac()).contains(search)  
 || item.getTenDoitac().contains(search)  
 || (item.getSoVonDaDauTu() != 0 && item.getSoVonDaDauTu().toString().contains(search))  
 || item.getDiaChi().contains(search))  
 .collect(Collectors.*toList*());  
 **doiTacs** = FXCollections.*observableList*(result);  
 **tableView**.setItems(**doiTacs**);  
 **if** (**clientSearchTextField**.getText().isEmpty()) {  
 setTableView(0);  
 }  
 }  
  
 **private** Node setTableView(Integer integer) {  
 setTableView();  
 **int** fromIndex = integer \* 10;  
 **int** toIndex = Math.*min*(fromIndex + 10, **doiTacs**.size());  
 ObservableList<DoiTac> doiTacObservableArray = FXCollections.*observableArrayList*(**doiTacs**.subList(fromIndex, toIndex));  
 **tableView**.setItems(**doiTacs**);  
 **return tableView**;  
 }  
  
 **public** Node setTableView() {  
 **btnDelete**.setDisable(**true**);  
 **btnEdit**.setDisable(**true**);  
 **tableView**.getItems().clear();  
 **doiTacList**.clear();  
 **doiTacList** = DoiTacService.*getAll*();  
 **pagingList** = **new** PaginatedList<>(**doiTacList**);  
 **clientPagination**.setPageCount(**pagingList**.**listOfPages**.size());  
 **doiTacs** = FXCollections.*observableList*(**doiTacList**);  
 **colID**.setCellValueFactory(**new** PropertyValueFactory<>(**"idDoiTac"**));  
 **colName**.setCellValueFactory(**new** PropertyValueFactory<>(**"tenDoitac"**));  
 **colLinhVuc**.setCellValueFactory(**new** PropertyValueFactory<>(**"linhVuc"**));  
 **colDiaChi**.setCellValueFactory(**new** PropertyValueFactory<>(**"diaChi"**));  
 **colSDT**.setCellValueFactory(**new** PropertyValueFactory<>(**"sdt"**));  
 **colEmail**.setCellValueFactory(**new** PropertyValueFactory<>(**"email"**));  
 **colSoVonDauTu**.setCellValueFactory(**new** PropertyValueFactory<>(**"soVonDaDauTu"**));  
 **tableView**.setItems(**doiTacs**);  
 **return tableView**;  
 }  
  
 **public void** selectItem() **throws** SQLException, IOException {  
 **btnEdit**.setDisable(**false**);  
 **btnDelete**.setDisable(**false**);  
 }  
  
 **public void** changeStage(ActionEvent actionEvent) **throws** IOException, SQLException {  
 **this**.**doiTac** = DoiTacService.*findByMaDoiTac*(**this**.**doiTac**.getIdDoiTac());  
 OutputStream os = **new** FileOutputStream(**new** File(**"photo.jpg"**));  
 **byte**[] content = **new byte**[1024];  
 **int** size = 0;  
 **if** (**this**.**doiTac**.getLogo() != **null**) {  
 **inputStream** = **this**.**doiTac**.getLogo();  
 **while** ((size = **inputStream**.read(content)) != -1) {  
 os.write(content, 0, size);  
 }  
 }  
 **imageView**.setImage(**new** Image(**"file:photo.jpg"**, 400, 550, **true**, **true**));  
 SingleSelectionModel<Tab> selectionModel = **tabView**.getSelectionModel();  
 selectionModel.select(1);  
 **txtName**.setText(**this**.**doiTac**.getTenDoitac());  
 **txtDiaChi**.setText(**this**.**doiTac**.getDiaChi());  
 **txtSDT**.setText(**this**.**doiTac**.getSdt());  
 **txtEmail**.setText(**this**.**doiTac**.getEmail());  
 **if** (**this**.**doiTac**.getSoVonDaDauTu() != **null**) {  
 **txtVonDauTu**.setText(Double.*valueOf*(**this**.**doiTac**.getSoVonDaDauTu()).toString());  
 } **else** {  
 **txtVonDauTu**.setText(**""**);  
 }  
 **comLinhVuc**.getSelectionModel().select(**this**.**doiTac**.getLinhVuc());  
  
 }  
  
 **public void** onSave(ActionEvent actionEvent) **throws** SQLException, IOException {  
 **matcher** = **emailPattern**.matcher(**txtEmail**.getText());  
 **if** (!**matcher**.matches()) {  
 **notification**.notification(**"Lỗi"**, **"email không hợp lê"**, 1);  
 **return**;  
 }  
 **matcher** = **phonePattern**.matcher(**txtSDT**.getText());  
 **if** (!**matcher**.matches()) {  
 **notification**.notification(**"Lỗi"**, **"phone không hợp lê"**, 1);  
 **return**;  
 }  
 String loaiHinh = **"Chưa đặt"**;  
 **if** (**comLinhVuc**.getSelectionModel().selectedItemProperty().getValue() != **null**) {  
 loaiHinh = **comLinhVuc**.getSelectionModel().selectedItemProperty().getValue().toString();  
 }  
 DoiTac doiTac = **new** DoiTac(**this**.**doiTac**.getIdDoiTac(), **txtName**.getText(), loaiHinh,  
 **txtDiaChi**.getText(), **txtSDT**.getText(),  
 **txtEmail**.getText(), Double.*parseDouble*(**txtVonDauTu**.getText()));  
  
 DoiTacService.*save*(doiTac, **this**.**file**);  
 **notification**.notification(**"Save thành công"**, **"Đã lưu vào database"**, 0);  
 setTableView();  
 }  
  
 **public void** edit(ActionEvent actionEvent) **throws** IOException, SQLException {  
 **this**.**doiTac** = **tableView**.getSelectionModel().getSelectedItem();  
  
 changeStage(actionEvent);  
 }  
  
 **public void** deleteItem(ActionEvent actionEvent) **throws** SQLException {  
 DoiTac doiTac = **tableView**.getSelectionModel().getSelectedItem();  
 DoiTacService.*deleteByMaDoiTac*(String.*valueOf*(doiTac.getIdDoiTac()));  
 setTableView();  
 }  
  
 **public void** printIntoExcel() **throws** IOException {  
 }  
  
 **public void** creatNew(ActionEvent actionEvent) **throws** IOException, SQLException {  
 **this**.**doiTac** = **new** DoiTac();  
 changeStage(actionEvent);  
 }  
  
 @FXML  
 **public void** onCancel(ActionEvent actionEvent) {  
 SingleSelectionModel<Tab> selectionModel = **tabView**.getSelectionModel();  
 selectionModel.select(0);  
 **this**.**doiTac** = **new** DoiTac();  
 }  
  
 @Override  
 **public void** initialize(URL location, ResourceBundle resources) {  
 setTableView();  
 **clientPagination**.setPageFactory(**this**::setTableView);  
 List<String> listDoiTacId = DoiTacService.*getAllID*();  
 **txtVonDauTu**.textProperty().addListener(**new** ChangeListener<String>() {  
 @Override  
 **public void** changed(ObservableValue<? **extends** String> observable, String oldValue,  
 String newValue) {  
 **txtVonDauTu**.setText(newValue.replaceAll(**"[^\\d\*|\\d+\\,\\d]"**, **""**));  
 **if** (!newValue.matches(**"\\d\*|\\d+\\,\\d\*"**) && !newValue.contains(**"."**)) {  
 **txtVonDauTu**.setText(newValue.replaceAll(**"[^\\d\*|\\d+\\,\\d]"**, **""**));  
 **notification**.notification(**"Ký tự nhập không phải là số"**, **"Vui lòng nhập lại"**, 1);  
 }  
 **try** {  
 **if** (Double.*parseDouble*(newValue) < 0) {  
 **txtVonDauTu**.setText(newValue.replaceAll(newValue, oldValue));  
 **notification**.notification(**"Số nhập phải lớn hơn 0"**, **"Vui lòng nhập lại"**, 1);  
 }  
 } **catch** (Exception e) {  
  
 }  
 }  
 });  
 }  
  
 **public void** uploadImage(ActionEvent actionEvent) {  
 Stage stage = (Stage) ((Node) actionEvent.getSource()).getScene().getWindow();  
 **file** = **fileChooser**.showOpenDialog(stage);  
 **if** (**file** != **null**) {  
 **imageView**.setImage(**new** Image(**file**.toURI().toString(), 1200, 1500, **true**, **true**));  
 }  
 }  
}

### Xử lý form Login

**package** controller;  
  
**import** com.jfoenix.controls.JFXButton;  
**import** com.jfoenix.controls.JFXCheckBox;  
**import** com.jfoenix.controls.JFXPasswordField;  
**import** com.jfoenix.controls.JFXTextField;  
**import** javafx.fxml.FXML;  
**import** javafx.fxml.FXMLLoader;  
**import** javafx.fxml.Initializable;  
**import** javafx.scene.Node;  
**import** javafx.scene.Parent;  
**import** javafx.scene.Scene;  
**import** javafx.stage.Stage;  
**import** model.User;  
**import** service.Notification;  
**import** service.UserService;  
  
**import** java.io.IOException;  
**import** java.net.URL;  
**import** java.sql.SQLException;  
**import** java.util.ResourceBundle;  
  
**public class** LoginController **implements** Initializable {  
 @FXML  
 **private** JFXTextField **emailTextField**;  
  
 @FXML  
 **private** JFXPasswordField **passwordTextField**;  
  
 @FXML  
 **private** JFXCheckBox **rememberMeCheckBox**;  
  
 @FXML  
 **private** JFXButton **btnLogin**;  
  
  
 **public** LoginController() **throws** SQLException {  
 }  
  
 **public** Notification **notification** = **new** Notification();  
  
 **public void** login(javafx.event.ActionEvent actionEvent) **throws** IOException, SQLException {  
 User user = UserService.*findByUserName*(**emailTextField**.getText());  
  
 **if** (user.getUserName() != **null** && user.getPassWord().equals(**passwordTextField**.getText())) {  
 FXMLLoader loader = **new** FXMLLoader(getClass().getResource(**"view/HomeView.fxml"**));  
 Parent root = loader.load();  
 Stage appStage = (Stage) ((Node) actionEvent.getSource()).getScene().getWindow();  
 appStage.hide();  
 Stage stage = **new** Stage();  
 stage.setScene(**new** Scene(root));  
 stage.show();  
 **notification**.notification(**"Đăng nhập thành công"**, **"Vai Trò : "** + user.getRole(), 0);  
 } **else** {  
 **notification**.notification(**"Đăng nhập thất bại"**, **"UserName hoặc mật khẩu sai"** + user.getUserName() +**" -- "** + user.getPassWord(),1);  
 }  
 }  
  
  
 @Override  
 **public void** initialize(URL location, ResourceBundle resources) {  
  
 }  
  
}

#### Xử lý form main

**package** controller;  
  
**import** controller.assets.efflect.FunLevelGauge;  
**import** javafx.application.Application;  
**import** javafx.application.Platform;  
**import** javafx.fxml.FXMLLoader;  
**import** javafx.geometry.Insets;  
**import** javafx.scene.Parent;  
**import** javafx.scene.Scene;  
**import** javafx.scene.layout.StackPane;  
**import** javafx.stage.Stage;  
  
**import** java.io.IOException;  
**import** java.util.Timer;  
**import** java.util.TimerTask;  
  
**public class** Main **extends** Application {  
 **private** FunLevelGauge **gauge**;  
 **double countdown** = 0;  
 Stage **primaryStage**;  
  
 **public void** setPrimaryStage(Stage primaryStage) {  
 **this**.**primaryStage** = primaryStage;  
 }  
  
 @Override  
 **public void** init() {  
 **gauge** = **new** FunLevelGauge();  
 **gauge**.setPrefSize(400, 400);  
 **gauge**.setLevel(0);  
 }  
  
 @Override  
 **public void** start(Stage primaryStage) **throws** Exception {  
 setPrimaryStage(primaryStage);  
 StackPane pane = **new** StackPane(**gauge**);  
 pane.setPadding(**new** Insets(10));  
 Scene scene = **new** Scene(pane);  
 primaryStage.setTitle(**"Quản lý ...."**);  
 primaryStage.setScene(scene);  
 primaryStage.show();  
 Timer t = **new** Timer();  
 TimerTask tt = **new** TimerTask() {  
 @Override  
 **public void** run() {  
 **countdown** = **countdown** + 0.5;  
 System.***out***.println(**countdown**);  
 **gauge**.setLevel(**countdown**);  
 **if** (**countdown** > 0.9) {  
 Platform.*runLater*(**new** Runnable() {  
 @Override  
 **public void** run() {  
 Parent root = **null**;  
 **try** {  
 root = FXMLLoader.*load*(getClass().getResource(**"view/Login.fxml"**));  
 } **catch** (IOException e) {  
 e.printStackTrace();  
 }  
 primaryStage.setScene(**new** Scene(root));  
 primaryStage.show();  
 }  
 });  
 t.cancel();  
 t.purge();  
 }  
 }  
 };  
 t.schedule(tt, 500, 2000);  
 }  
  
 **public static void** main(String[] args) {  
 *launch*(args);  
 }  
}

# Kiểm thử

## Kiểm thử form X

## Kiểm thử form Y

# Đóng gói và triển khai

## Hướng dẫn chuyển đổi jar thành exe

## Hướng dẫn cài đặt triển khai

## Hướng dẫn sử dụng phần mềm