**Hệ thống thanh toán JPayment**

1. **Tổng quan**

Jpayment với mục tiêu thanh toán qua điện thoại. Người tiêu dùng ra đường không cần mang theo tiền mặt hay thẻ.

Hệ thống gồm 2 phần

* Phần 1: Payment (thanh toán trực tuyến): Cung cấp cơ chế thanh toán trực tuyến để tích hợp vào các hệ thống mua hàng của bên thứ 3
* Phần 2: Shop (Các cửa hàng với các thiết bị POS): Hỗ trợ hiện thực hóa payment. Cung cấp phần mềm quản lý miễn phí dành cho các cửa hàng vừa và nhỏ, chưa có điều kiện đầu tư 1 hệ thống quản lý bán hàng riêng

1. **Chức năng**
2. **Payment**

* Thanh toán bằng QR code
* Quản lý người dùng
  + Đăng ký
  + Thông tin cá nhân
  + Thông tin thanh toán
* Quản lý lịch sử giao dịch

1. **Quản lý POS bán hàng**

* Quản lý shop của người dùng
  + Thêm, xóa, sửa
  + Mỗi tài khoản có thể mở được nhiều shop
* Quản lý thông tin mặt hàng của shop
* Quản lý bán hàng
* Quản lý thiết bị POS
* Thống kê doanh thu theo mốc thời gian tự chọn

1. **Kiến trúc – công nghệ**
2. **Sơ lược - mục tiêu**

* Riêng phần payment Tiến sẽ lo hết, bà con chỉ cần làm tốt về phần quản lý shop. Nếu dư thời gian thì triển khai thêm các model phụ khác.
* Mục tiêu của phần quản lý shop là hỗ trợ hiện thực hóa Jpayment. Cung cấp 1 site miễn phí cho các doanh nghiệp vừa và nhỏ để họ bán hàng và nhận thanh toán thông qua Jpayment. Doanh nghiệp muốn sử dụng phần mềm quản lý shop của Jpayment thì sẽ trải qua các bước sau:

B1. Mở tài khoản

B2. Tạo shop (có thể tạo nhiều shop)

B3. Đăng sản phẩm cho shop(Cung cấp thông tin về giá, mô tả ngắn, hình)

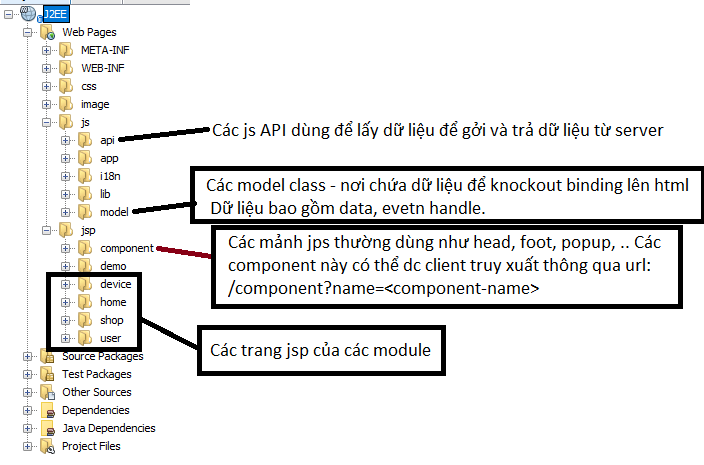
B4. Cấu hình máy POS cho shopa tại điểm bán hàng để người mua quét QR thanh toán

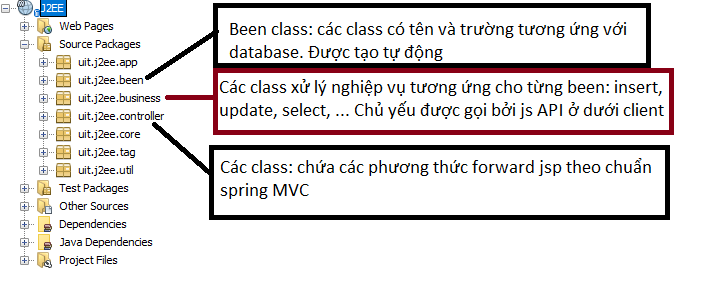
* Vậy phần quản lý shop tập trung chủ yếu vào phần quản lý các thông tin của shop, bao gồm quản lý sản phẩm của shop, quản lý POS của shop, thông kê, …

1. **Các công nghệ sử dụng**

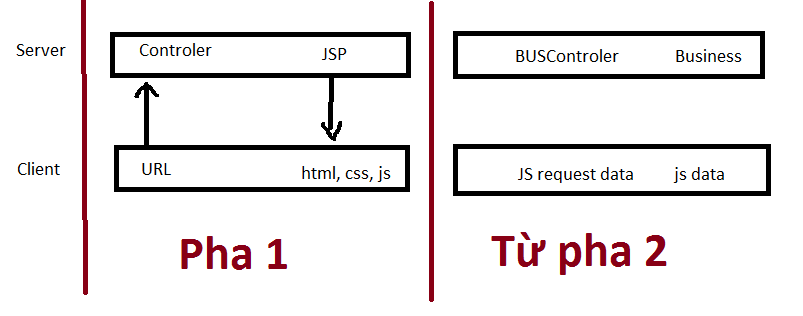
* Server:
  + J2EE: servlet, jsp, custom tag.
  + Spring MVC: hỗ trợ triển khai mô hình MVC tốt hơn
  + Hibernate: Truy xuất csdl
  + Lambda expression: Hỗ trợ lọc kết quả từ hibernate. Phần này các dev có thể không cần quan tâm, đổi lại tự viết hibernate hơi cực hơn chút đỉnh
* Client:
  + Html: Ngôn ngữ dựng GUI cho web
  + Css: giúp thay đổi html về kích thước, màu sắc, nền, …
  + Javascript: Ngôn ngữ lập trình dùng để thực hiện các xử lý ở tại browser, gọi tắt là js
  + Bootstrap: là 1 thư viện css, hỗ trợ việc dựng GUI nhanh hơn
  + Jquery: Là 1 thư viện js, giúp thực hiện việc thao tác với dom nhanh và ngắn hơn. Dom đại khái là các thẻ html
  + Knockout: là 1 thư viện js. Mục tiêu chính là để binding dữ liệu từ html xuống JS và ngược lại. Với knockout, việc tương tác với các tag html trực quan hơn, dễ hơn, và an toàn hơn.

1. **Cấu trúc project**

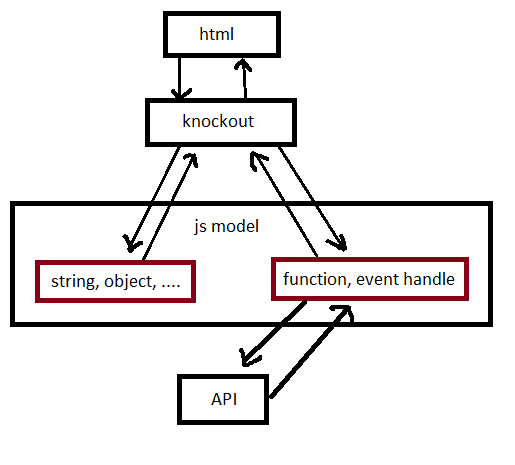
-client: 

-Server: 

1. **Kiến trúc hệ thống**

*Các yêu cầu của người dùng đều trải qua ít nhất 2 pha*

* Pha 1: lấy giao diện
* Từ pha 2 trở đi: Khi đã có GUI, người dùng có thể tương tác với GUI, các sự kiện tương ứng sẽ được gọi và data sẽ được chuyển lên và nhận về từ server bằng cơ chế ajax

1. **Kiến trúc client**

* Knockout (ko) sẽ binding dữ liệu từ trên html xuống model. Dữ liệu bao gồm:
  + js data: String,number, object, …
  + Function: các hàm bắt sự kiện click, press, ….
* Các function trong model sẽ được gọi khi sự kiện tương ứng xảy ra. Trong function sẽ xử lý. Nếu cần chuyển hay nhận từ server, nó sẽ gọi tới các API.
* API chỉ đóng vai trò chuyển và nhận dữ liệu giữa client và server. Khi client muốn liên lạc với server, cần cung cấp cho API dữ liệu muốn chuyển lên, và 1 callback funtion để nhận dữ liệu từ server trả về

***Chú ý: tất cả dữ liệu server trả về đều được đóng gói trong 1 đối tượng. Trong đó:***

* ***Thuộc tính status code cho biết phía server sử lý thành công hay lỗi***
* ***Thuộc tính data chứa dữ liệu mà clietn muốn nhận***

1. **Kiến trúc server**

MVC: triển khai như spring MVC bình thường. Cần 1 controller và 1 jsp cho 1 trang

1. **Hướng dẫn triển khai module mới**

***Phần này sẽ lấy module đăng nhập để làm ví dụ***

1. **Kết nối csdl**

* Vào mysql, import script
* Tạo user: jpayment, password: jpayment. Cấp full quyền

1. **Tạo controller trên server**

***Controller sẽ làm nhiệm vụ chọn file jsp để trả về cho mỗi url***

* Mở packet uit.j2ee.controller
* Tạo 1 controller class:

*@Controller*

*public class UserCTR {*

*@RequestMapping("/user")*

*public ModelAndView loginOrRegister() {*

*return new ModelAndView("user/login-register");*

*}*

*}*

“/user” là đường dẫn user nhập vào browser

**“***user/login-register***”** đường dẫn file jsp trong thư mục jps. *Chú ý là không thêm đuôi .jsp vào.*

* Mỗi model chỉ tạo 1 controller class, nếu có nhiều trang thì làm nhiều method controller là được

1. **Tạo trang jsp**

***JSP chủ yếu chứa code html***

* Vào thư mục “Web Pages/jsp” tạo 1 folder user, trong folder đó tạo file jsp tên là *login-register.jsp*
* Trong trang jsp, viết code html như html bình thường

*Xong xuôi có thể test đường dẫn “/user” xem đã hiển thị trang jsp vừa tạo chưa. Ok rồi thì qua pha thứ 2*

1. **Tạo business (Bắt đầu pha 2)**

***Các business class thực hiện các nghiệp vụ của hệ thống, chủ yếu được dùng bởi client thông qua ajax***

* Mở packet uit.j2ee.business, tạo 1 business class:

public class UserBUS extends Business<User> {

public BUSRespone Login(String email, String password) throws Exception{

try {

Object re = LoginDAO(email, password);

if (re == null) {

return new BUSRespone(null, BUSStatus.busFailed());

} else {

return new BUSRespone("login success");

}

} catch (Exception ex) {

return new BUSRespone(null, BUSStatus.exceptiond(), ex);

}

}}

*Chú ý, tất cả các method đều nên bọc kết quả về vào 1 đối tượng BUSRespone, Để khi client nhận được kết quả thì biết được quá trình xử lý trên server thành công hay thất bại, hay exception, …*

* Business class cần kế thừa từ class Business để có sử dụng lại một số hàm thao tác với csdl, lấy thông tin request, response
* Trong ví dụ trên có truyền class User là kiểu genneric, các dev khi triển khai business class cho bảng nào thì trền thay thế tương ứng

1. **Tạo API**

**API ở đây là các JS API**

* Mở thư mục “Web pages/js/api”. Tạo js file đặt tên là APIUser.js
* Thêm vào 2 dòng sau:

var APIUser = {};

APIUser.BUSClassName = "UserBUS"; // tên business class

* Thêm tiếp vào dưới 1 method có nội dung như sau:

APIUser.Login = function (email, password, onSuccess, onError, skipIfEror) {

Ajax.J2EE(

APIUser.BUSClassName, //tên business class

"Login", //tên method cần gọi trong business class

["email", "password"], // tên tham số của method

[email, password], // dữ liệu tương ứng cho tham số của method

Ajax.callback.object, // kiểu dữ liệu trả về là object

onSuccess, // function js dc gọi khi server trả kết quả về

onError, // nếu phía server bị crash hay vấn đề gì đó sẽ gọi onError

skipIfEror // flag cho biết nếu dữ liệu trả về không đúng kiểu yeu cầu thì có gọi onSuccess hay không

);

}

1. **Tạo model**

**Model là nơi tập trung dữ liệu để knockout binding lên html.**

* Mở thư mục “Web pages/js/model”. Tạo js file đặt tên là login-register.js
* Code:

function MODEL\_LOGIN() {

var t = this;

var data = {

first: "",

lastName: "",

email: "",

password: ""

}

t.user = ko.observable(data);

t.loginError = ko.observable("");

// do login

t.login = function () {

// check valide

var user = t.user();

if (user.email == false || user.password == false) {

t.loginError("Bạn vui lòng điền email và password");

return;

}

// call api

APIUser.Login(user.email, hex\_md5(user.password), function (data) {

if (data.statusCode == BUSResponeCode.ok) {

window.location.href = APP.contextPath;

} else if (data.statusCode == BUSResponeCode.busFailed) {

t.loginError("Sai email hoặc password");

} else {

console.warn(data);

t.loginError("Đăng nhập chưa thành công");

Popup.failed();

}

});

}

}

1. **Binding knockout**

* Mở lại trang jsp, thêm thuộc tính data-bind vào các thẻ cần show dữ liệu. vd1:

<input data-bind="value: user().email”>

*Thẻ trên sẽ được binding dữ liệu vào trường email của data trong model*

Vd2:

<input data-bind="click: login">

*Thẻ trên sẽ được binding sự kiện click vào function login trong model*

* Xong xuôi quay lại trang js model, thêm 2 dòng code sau:

var MODEL = new MODEL\_LOGIN();

ko.applyBindings(MODEL);