FCTUC FACULDADE DE CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE DE COIMBRA

Projeto de Sistemas Distribuídos

IBei: Leilões Invertidos

2016/2017

Luis Ramos, 2008112685
Joel Pires, 2014195242

Índice

Introdução

Este projeto consiste numa aplicação que segue a lógica de um leilão invertido, isto é, os leilões são criados por quem quer efetivamente comprar o produto (com o valor inicial correspondente ao preço que está disposto a pagar) e os vendedores licitam oferecendo preços de venda sucessivamente mais baixos. Assim, ganha o vendedor que oferecer o valor mais baixo.

Requisitos Funcionais

- **Login:** Apenas utilizadores autenticados com username e password na base de dados poderão ter acesso à plataforma.

- **Logout:** Utilizadores previamente autenticados podem encerrar a sua sessão completamente do sistema.

- **Registo:** Um novo registo envolver que o utilizador forneça um *username* que não exista já na base de dados (*username* unívoco portanto).

- **Criar Leilão:** Cada leilão tem associado a si então um título, um código do artigo a leiloar (IBAN de 13 dígitos), um título, uma descrição e a data, hora e minuto do seu início. Caso o código do artigo não exista na base de dados do sistema, então o seu registo é criado na base de dados. Para além disso, o comprador indica o preço máximo que está disposto a pagar.

- **Pesquisar Leilões:** Através do código do artigo envolvido no leilão é possível listar todos os leilões que possuem esse artigo. Apenas é apresentada uma breve descrição do leilão composta por título e o respetivo código do artigo no entanto, é possível consultar todos os detalhes relativos ao leilão.

- **Consultar Detalhes de um Leilão:** Cada leilão têm um id associado e, por isso, inserindo esse mesmo id o utilizador terá acesso a todas as propriedades do leilão, inclusivamente todas as licitações e mensagens escritas no seu mural.

- **Listar todos os Leilões em que o Utilizador tenha atividade:** Uma listagem de todos os leilões que o utilizador criou ou no qual licitou e/ou escreveu mensagens. Apenas é apresentada uma breve descrição do leilão composta por título e o respetivo código do artigo no entanto, é possível consultar todos os detalhes relativos ao leilão.

- **Licitar um Leilão:** Um utilizador pode licitar num ou mais leilões desde que:

    - ele não seja o criador dos leilões em causa;

- o montante a que sugere vender o artigo do leilão seja inferior ao estipulado pelo comprador e inferior à licitação mais baixa.

- O leilão ainda não tenha terminado

- **Editar Parâmetros de um Leilão:** é possível editar qualquer parâmetro de um leilão nesta operação à exceção das mensagens e licitações a ele associados. Editar a data de término para uma data anterior à atual também não é permitido.

- **Escrever mensagens no mural do Leilão:** Cada leilão tem um mural onde constam mensagens de utilizadores.

- **Listar Utilizadores Online:** Uma lista completa de todos os utilizadores que se encontram online é disponibilizada ao utilizador.

- **Término do Leilão:** Assim que o leilão termina, é impossível licita-lo uma vez que o vencedor é determinado.

- **Notificação Imediata de Licitação Melhor:** Todos os utilizadores que se encontrem online são notificados imediatamente caso tenha havido uma licitação melhor num leilão que eles tinham licitado.

- **Notificação Imediata de Mensagens:** Quer o criador do leilão como aqueles que escreveram no seu mural recebem imediatamente notificações das mensagens dos utilizadores que escreveram mensagens nesse leilão.

- **Notificação de Mensagens a utilizadores offline:** Caso tanto o criador do leilão como aqueles que escreveram no seu mural não estejam online, assim que fiquem online recebem essas mesmas notificações de mensagens.

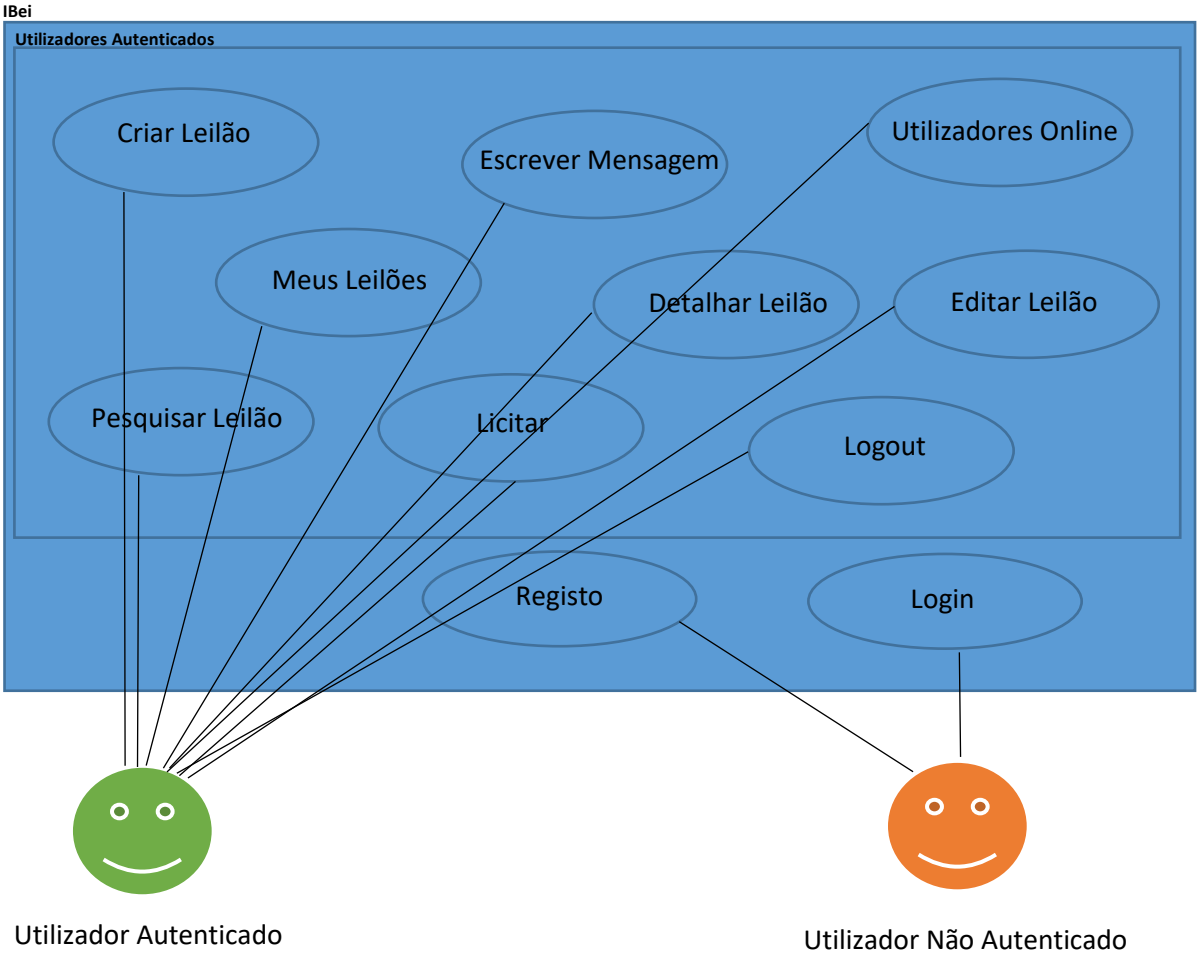
- **Associar Conta ao Facebook:** Uma pessoa que se autentique no nosso sistema pode associar a conta do Facebook e assim poderá tanto logar no Facebook como criar posts automaticamente

- **Login com o Facebook:** Um utilizador pode perfeitamente entrar no nosso sistema apenas por se logar na sua conta do Facebook.

- **Postar Leilão no Facebook:** Quando um utilizador cria um leilão, o sistema posta automaticamente no Facebook com o link do leilão, desde que ele tenha a conta do Facebook associada

- **Mostrar Preço mais baixo no eBay:** Ao detalhar um leilão, é possível ver qual o preço mais baixo a que o artigo desse leilão se encontra no eBay.

O Diagrama dos Casos de Uso é o seguinte então:

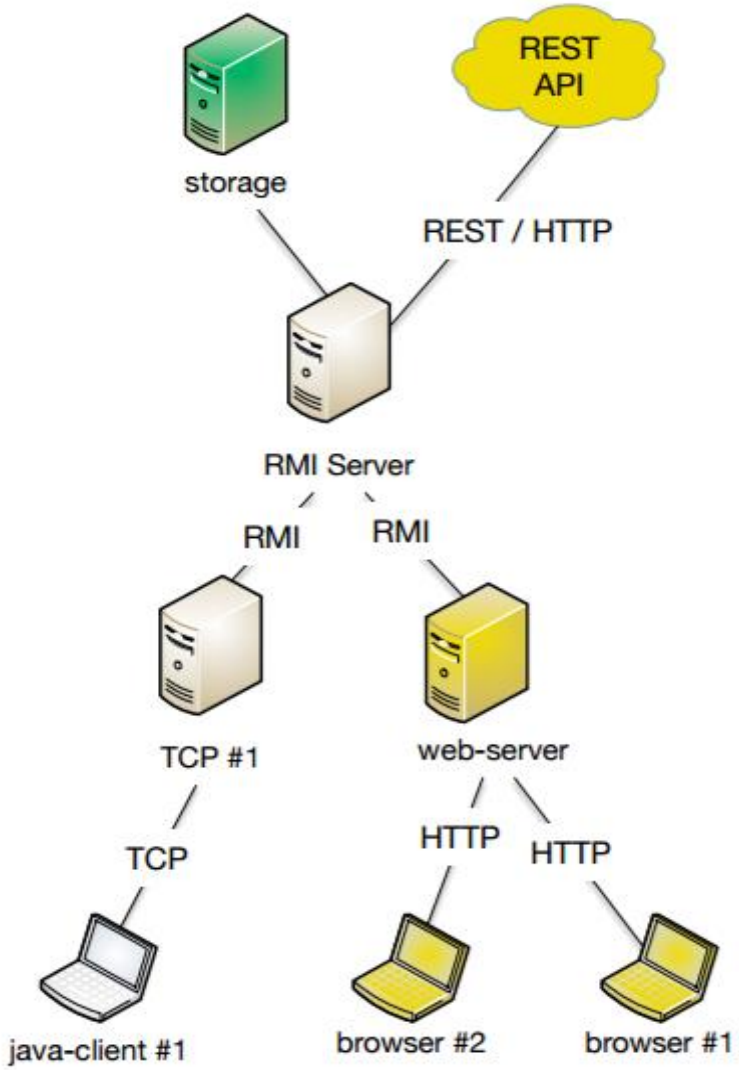


Requisitos Não Funcionais

Alguns requisitos não funcionais mereceram a nossa atenção aquando da implementação do sistema: Entre eles encontram-se:

- **Usabilidade:** O sistema deve ser fácil de utilizar e bastante intuitivo.
- **Fiabilidade:** O sistema deve ser fiável sem problemas de concorrência
- **Escalabilidade:** Temos que pensar em grande e, portanto, temos que preparar o nosso sistema para ser escalável, ou seja, prepara-lo para ser usado por um grande número de pessoas simultaneamente.
- **Segurança:** O nosso sistema deve preservar a integridade dos dados e, em circunstância alguma, perder ou adulterar informação.
- **Privacidade:** Alguns dados precisam de ser encriptados antes de serem expostos na nossa base de dados

Arquitetura



Esta figura mostra a arquitetura geral do projeto. O sistema deverá também aceitar qualquer número de clientes TCP, desde que estes sigam o protocolo especificado.

O servidor RMI devem ter os dados persistidos de forma a que nunca se percam, mesmo que os processos sejam terminados. Para tal, vamos usar uma base de dados.

O web Server liga-se à base de dados por RMI. A aplicação web iBei vai correr num servidor HTTP (Apache Tomcat) que atuará como um cliente RMI para com o servidor RMI. Os clientes irão usar browsers para se ligarem ao servidor web para pedirem páginas através de HTTP. Para as notificações em tempo real usamos WebSockets.

Para além disso, há uma integração com outros dois serviços web: o Facebook e o eBay. O Facebook dinamizará a partilha social dos leilões, bem como fornecer uma alternativa ao login por username e password. Já o eBay será usado para obter um preço de referência para as licitações.

Duplicação de Pedidos

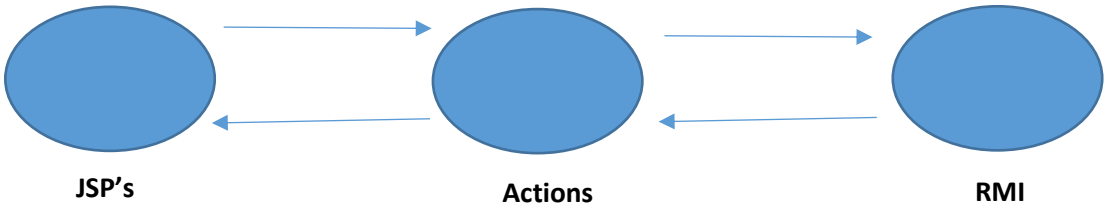
Uma das dificuldades que tivemos durante a execução do projeto teve que ver com a possibilidade de haver duplicação de pedidos dos clientes. Se o servidor RMI fosse abaixo precisamente quando o pedido fosse despachado (e, por sua vez, a base de dados alterada) mas imediatamente antes de o RMI devolver a resposta ("ok:true" ou "ok:false"), o pedido seria repetido.

Optamos por utilizar **UUID's.** No nosso servidor e em cada Action da nossa aplicação, um UUID único que identifica o pedido é gerado. O UUID juntamente com uma flag (que identifica se a base de dados foi alterada ou não) é mandado para o RMI. O RMI, assim que a base de dados é alterada, altera essa flag para 1 e envia de novo para o servidor.

Assim, mesmo que o RMI vá abaixo, ao receber novamente o pedido do servidor, ele vai consultar se a flag da base de dados está a '1' ou a '0', isto é, se o pedido foi efetivamente tratado ou não.

Integração do Struts com o Servidor RMI

Sumariamente, o Struts permite-nos injetar conteúdo de java nos JSP's e permite-nos executar ações que fazem os pedidos ao RMI.



O struts revela-se então muito útil porque nos ajuda a implementar uma estrutura MVC. Através da declaração de uma *<action>* nós podemos associar diferentes JSP's para diferentes tipos de resultados ("success", "error", etc). Podemos ainda associar uma classe Java a ser executada por essa action. É nessa classe de Java que se fazem os pedidos ao RMI e que se atualizam variáveis que nos irão ser úteis mostrar. Através da integração e da libraria de tags que o struts nos oferece, conseguimos referencias variáveis das classes do Java nos JSP's correspondentes.

Eis o exemplo do nosso "login" no sistema para mostrar as vantagens do encapsulamento que o struts oferece:

```xml
<action name="login" class="web.action.LoginAction" method="execute">
    <interceptor-ref name="newStack"/>
    <result name="success">WEB-INF/home.jsp</result>
    <result name="error">WEB-INF/loginerror.jsp</result>
    <result name="stay">login.jsp</result>
</action>
```

Struts.xml

```java
public class LoginAction implements SessionAware {
    private Map<String, Object> session;

    private String username;
    private String password;

    public String execute() throws Exception {
        String reply;

        if(username == null || password == null) return "stay";

        Bean myBean = new Bean();
        myBean.setUsername(getUsername());
        myBean.setPassword(getPassword());

        reply = myBean.login();

        if(reply.equals(Action.SUCCESS)) {
            ArrayList<String> notifications = myBean.startUpNotifications();

            session.put("notifications", notifications);
            session.put("username", getUsername());
            session.put("loggedin", true);
        }

        return reply;
    }
```

LoginAction.java

```jsp
<form action="login.action" method="POST" class="form login">
    <div class="form__field">
        <label for="login__username"><svg class="icon"><use xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href="#user"></use></svg><span class="hidden
        <input id="login__username" type="text" name="username" class="form__input" placeholder="Username" required>
    </div>

    <div class="form__field">
        <label for="login__password"><svg class="icon"><use xmlns:xlink="http://www.w3.org/1999/xlink" xlink:href="#lock"></use></svg><span class="hidden
        <input id="login__password" type="password" name="password" class="form__input" placeholder="Password" required>
    </div>

    <div class="form__field">
        <input type="submit" value="Log in">
    </div>
</form>
```

login.jsp

Integração de WebSockets com Struts e RMI

Os WebSockets são importantes como protocolo de comunicação bidirecional através de única conexão TCP. Assim um cliente e um servidor podem enviar mensagens independentemente dos outros, especialmente útil no nosso projeto porque nós queríamos notificar os clientes com mensagens.

O WebSocket define um handshake de abertura de mensagens em camadas sobre TCP, isto é, duas maquinas afirma uma à outra que a reconheceu e esta pronta para iniciar a comunicação.

A API Java para WebSocket permite que páginas web usem este protocolo para comunicação bidirecional com o host remoto (ao contrário do que acontece com o HTTP em que há necessidade de criar uma nova conexão TCP para cada troca de mensagens entre cliente e servidor).

Através do Websocket o RMI consegue estar "aware" do servidor e, por isso, é possível fazer *subscribe* ao *RMI* e assim notificar os clientes ligados ao servidor.

Para além disso, o Struts foi útil para a função online_users(). Através do Struts utilizámos uma *action* que pede ao RMI os utilizadores online. Ora, os Websockets permitem pôr essa informação na *session* e o Struts permite-nos injetar informação da *session* para os *JSP's*.

```
<action name="onlineusers" class="web.action.OnlineUsersAction" method="execute">
    <interceptor-ref name="newStack"/>
    <result name="success">WEB-INF/onlineusers.jsp</result>
    <result name="error">WEB-INF/onlineuserserror.jsp</result>
</action>
```

Struts.xml

```
<body>
    <c:forEach items="${sessionScope.onlinebean.onlineUsersList}" var="someone">
        <c:out value="${someone.username}"/><br>
    </c:forEach>
    <s:url action="login" var="url"/>
    <s:a href="%{url}">Home</s:a>

    <div id="container">
        <div id="notifications"></div>
    </div>
</body>
```

onlineusers.jsp

```java
public class OnlineUsersAction implements SessionAware {
    private Map<String, Object> session;
    private String username;


    public String execute() {
        setUsername(session.get("username").toString());
        session.remove("notifications");

        Bean myBeans = new Bean();

        myBeans.setUsername(getUsername());
        myBeans.setOnlineUsersList(new ArrayList<>());

        String reply = myBeans.onlineusers();

        if(reply.equals(Action.SUCCESS)) {
            session.put("onlinebean", myBeans);
            return Action.SUCCESS;
        } else return Action.ERROR;
    }
```

onlineusersAction.java

```java
@OnOpen
public void open(Session session, EndpointConfig config) {
    this.wsSession = session;
    this.httpSession = (HttpSession) config.getUserProperties().get(HttpSession.class.getName());

    try {
        this.wsHelper = new WebSocketHelper(this, httpSession);
    } catch(RemoteException ignored) {}
}
```

websocket.java

Integração de APIs REST

No nosso projeto usámos duas Restful APIs, a Finding API do eBay e a Graph API do Facebook.

No primeiro caso, aquilo que se fez foi basicamente um chamada *GET* através de uma conexão HTTP à API do Ebay com os determinados parâmetros que queríamos (e consequente *parse* e processamento da resposta) na *action* da Detail Auction.

Relativamente ao segunda caso, algo mais complicado foi feito, isto porque tivemos que usar serviços do *OAuth* e a biblioteca *ScribeJava.* Esta biblioteca permite-nos criar um serviço

OAuth que verifica e autoriza o nosso acesso à API do Facebook pelo utilizador, dando-nos um token privado e temporário (que nós armazenamos na base de dados), em vez de as próprias credenciais da conta do Facebook do utilizador. Eis o nosso código:

```
OAuthService service = new ServiceBuilder().provider(FacebookApi.class).apiKey(appid).apiSecret(appsecret).callback("http://localhost:8080/facebooklogin/").scope("publish_actions

String authorizationUrl = service.getAuthorizationUrl(null);

setAuthorizationUrl(authorizationUrl);
```

```
OAuthService service = new ServiceBuilder().provider(FacebookApi.class).apiKey(appid).apiSecret(appsecret).callback("http://localhost:8080/facebooklogin/").scope("publish_actior

Verifier verifier = new Verifier(code);

Token accessToken = service.getAccessToken(null, verifier);
String token = accessToken.getToken();
String PROTECTED_RESOURCE_URL = "https://graph.facebook.com/me";
OAuthRequest request = new OAuthRequest(Verb.GET, PROTECTED_RESOURCE_URL, service);
service.signRequest(accessToken, request);
Response response = request.send();
```

Testes de Software

Funcionalidade	DESCRIÇÃO DO TESTE	PASSOU?
LOGIN/LOGOUT	Utilizador que faça login inserindo credenciais que não estão na base de dados não entra no sistema	Passou
	Utilizador não autenticado não entrar no sistema alterando o URL	Passou
	Utilizador autenticado que faz logout fica com a sua sessão completamente terminada	Passou
	Utilizador que faça login com credenciais na base de dados entra na *home page* do sistema	Passou
	Utilizador autenticado não pode voltar para o URL de login sem primeiro passar por pelo logout	Passou
REGISTO	Utilizador autenticado não pode voltar para o URL de registo sem primeiro fazer logout	Passou
	Utilizador não autenticado pode registar-se com um username que ainda não existe na base de dados	Passou
	Utilizador não autenticado não se pode registar com um username que já exista na base de dados	Passou
CRIAR LEILÃO	Utilizador não consegue criar leilão com uma data anterior à atual	Passou
	Utilizador não consegue criar leilão com um montante que não seja válido	Passou
	Caso o utilizador insira um código de um artigo que ainda não existe, esse artigo é criado juntamente com o leilão	Passou
	O utilizador não consegue criar o leilão caso o código do artigo inserido não tenha 13 dígitos	Passou
	O utilizador consegue criar o leilão com todos os *inputs* válidos	Passou

PESQUISAR LEILÃO	O utilizador vê uma mensagem de erro caso pesquise por um código que não tenha 13 dígitos ou caso não haja nenhum leilão associado ao código	Passou
	O utilizador consegue criar o leilão com todos os *inputs* válidos	Passou
MEUS LEILÕES	Utilizador, caso esteja associado a leilões, consegue listar todos os leilões que criou ou no qual licitou e/ou escreveu mensagens.	Passou
	Utilizador que não esteja associado a leilão nenhum vê uma mensagem de erro.	Passou
ESCREVER MENSAGEM	Utilizador que tenta escrever mensagem para um id de leilão que não existe, vê uma mensagem de erro.	Passou
	Uma mensagem é submetida corretamente caso o utilizador insira um id de leilão válido	Passou
DETALHAR LEILÃO	Utilizador que tenta detalhar um leilão cujo id não existe, vê uma mensagem de erro.	Passou
	Utilizador que insere um id de leilão válido tem acesso a todas as propriedades desse leilão, incluindo todas as licitações e mensagens escritas no seu mural.	Passou
UTILIZADORES ONLINE	O sistema consegue encontrar os utilizadores que estão online	Passou
LICITAR	Quando o utilizador insere um montante inválido (montante não inteiro positivo) vê uma mensagem de erro	Passou
	Quando o utilizador insere um id de leilão que não existe, vê uma mensagem de erro	Passou
	Quando o utilizador insere um montante igual ou superior ao estabelecido pelo comprador ou ao da licitação mais baixa, vê uma mensagem de erro	Passou
	Quando o utilizador insere um id de leilão que já acabou, vê uma mensagem de erro	Passou

	Quando o utilizador todos os parâmetros válidos, a licitação é registada	Passou
EDITAR LEILÃO	Se o utilizador edita a data do leilão para uma anterior à atual, então vê uma mensagem de erro	Passou
	Se o utilizador tenta editar um id de leilão que não existe, então vê uma mensagem de erro	Passou
	Se o utilizador tenta mudar o código do artigo do leilão para um que não tenha 13 dígitos, vê uma mensagem de erro.	Passou
	Se o utilizador tenta mudar o montante do leilão para um montante inválido (não inteiro positivo), vê uma mensagem de erro	Passou
	O utilizador consegue editar uma ou mais propriedades do leilão e o sistema altera o leilão e regista no histórico de alterações.	Passou
DUPLICAÇÃO DE PEDIDOS (UUID)	Ao mandar o RMI abaixo precisamente no tempo em que, depois de tratar do pedido, ia retornar a resposta ao servidor, o pedido não é duplicado e o cliente recebe sempre a resposta correta.	Passou
NOTIFICAÇÕES IMEDIATAS	O utilizador consegue receber uma notificação imediatamente, independentemente da página em que se encontra.	Passou
	Um utilizador que licitou num leilão é notificado imediatamente mais que uma vez se houver mais que uma licitação melhor que a dele	Passou
	O utilizador que escreveu no mural de vários leilões é notificado múltiplas vezes quando múltiplos clientes também comentam no mural desses leilões.	Passou
NOTIFICAÇÕES NÃO IMEDIATAS	O utilizador que escreveu no mural de vários leilões e que entretanto ficou offline é notificado múltiplas vezes quando múltiplos clientes	Passou

	também comentam no mural desses leilões assim que fica online.	
ASSOCIAR COM O FACEBOOK	O utilizador que carrega em "associar com o facebook" pode postar no Facebook assim que cria um leilão	Passou
	O utilizador que carrega em "associar com o facebook" estando ele já associado ou logado pelo facebook, verá uma mensage de erro.	Passou
LOGIN COM O FACEBOOK	O utilizador ao optar por "login with Facebook" consegue entrar no nosso sistema.	Passou
	O utilizador ao optar por "login with Facebook" consegue fazer post automaticamente na sua conta assim que cria um leilão.	Passou
POSTAR LEILÃO NO FACEBOOK	Um utilizador que se logou com a sua conta do Facebook ou que se logou no nosso sistema mas associou a sua conta do Facebook, consegue fazer post automaticamente do leilão que criou.	Passou
PREÇO DO EBAY	Uma pessoa, ao detalhar um leilão com um código de um artigo que existe no eBay, tem acesso ao preço mais barato desse artigo no ebay.	Passou
	Uma pessoa, ao detalhar um leilão com um código de um artigo que não existe no eBay, é avisado que não foi encontrado o artigo no eBay.	Passou

Todos estes testes de software (à exceção dos testes relativos às últimas 4 funcionalidades) foram efetuados entre 2 computadores ligados por cabo LAN e com vários clientes foram abertos simultaneamente.

Conclusão

Este trabalho foi importante porque pudemos trabalhar com WebSockets e APIs, mecanismos estes que continuam a ser cada vez mais usados e são importantíssimos para construir uma aplicação web rica e com elevado grau de interação.

Distribuir o nosso sistema em vários componentes que correm em máquinas diferentes foi um verdadeiro desafio. Conseguimos ainda aprofundar os nossos conhecimentos em Java e construir uma plataforma do qual nos orgulhamos.