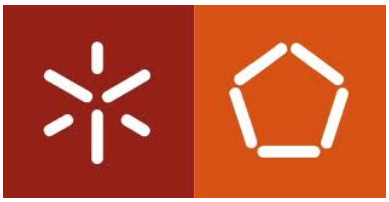

Universidade do Minho
Mestrado em Engenharia Informática
Análise e Conceção de Software

Aplicação WEB usando Servlets e JSP

Bikes4u

Relatório do 2º trabalho de Arquiteturas de Software

6 de Julho de 2012

GRUPO 5

- Ana Sampaio
- André Ribeiro
- Andreia Silva
- Cedric Pimenta
- Rafael Abreu

Resumo

Este relatório diz respeito ao segundo trabalho prático realizado no âmbito do módulo de Arquiteturas de Software, designado Bikes4u.

Com este trabalho pretendia-se desenvolver uma aplicação *web* capaz de realizar vendas, reservas e alugueres de bicicletas, possibilitando uma empresa deste ramo do negócio realizar o seu conjunto de atividades junto dos seus clientes via internet.

Partindo de um conjunto de módulos de gestão do negócio claramente definidos, começamos por estabelecer os principais uses cases envolvidos nessas atividades, organizando-os em packages. Posteriormente definimos o diagrama de classes.

Antes ainda de partirmos para a implementação, realizámos um esboço de como são implementadas as principais operações na tecnologia utilizada. Todas as decisões de implementação são abordadas ao longo deste relatório.

1 Introdução

Este trabalho prático surge no contexto do módulo de Arquiteturas de Software, inserido na Unidade Curricular de Análise e Conceção de Software (ACS). De forma a aplicar os conceitos abordados nas aulas deste módulo, foi proposto este trabalho prático, que corresponde a uma aplicação web, cuja implementação recorre a tecnologia de Servlets e JavaServer Pages (JSP).

De uma forma geral, o objetivo desta aplicação consiste em tornar possível a gestão do negócio de uma empresa de aluguer e venda de bicicletas, a partir de uma aplicação web. Assim, um funcionário da empresa pode efetuar o registo da venda de uma bicicleta, processar uma reserva através da oficialização do aluguer, e registar o início e fim da manutenção de uma bicicleta.

A aplicação deve ainda possibilitar que um cliente registado faça reservas de bicicletas, especificando apenas o intervalo de tempo da reserva, e a quantidade de bicicletas que pretende de uma dada categoria. Neste processo, é necessário ter em conta a disponibilidade das bicicletas em stock no período da reserva.

Para a realização deste projeto, começamos por definir os principais use cases organizados em packages, que correspondem às principais operações que podem ser realizadas na aplicação. Por sua vez, diagrama de classes estabelece as classes e os relacionamentos entre elas.

Para nos auxiliar na identificação dos Servlets, foram desenhados diagramas fluxo que representam, passo a passo, a implementação das 5 principais funcionalidades identificadas.

2 Diagrama de Use Cases

O inicio do projeto foi marcado pelo desenvolvimento do diagrama de use cases, conforme ilustra a figura seguinte, onde estão representadas as operações que podem ser realizadas no sistema. Destas, destacam-se as funcionalidades de Adicionar uma Bicicleta, Efetuar uma Reserva, Registar um Aluguer, Registar uma venda e Registar o início da Manutenção de uma bicicleta.

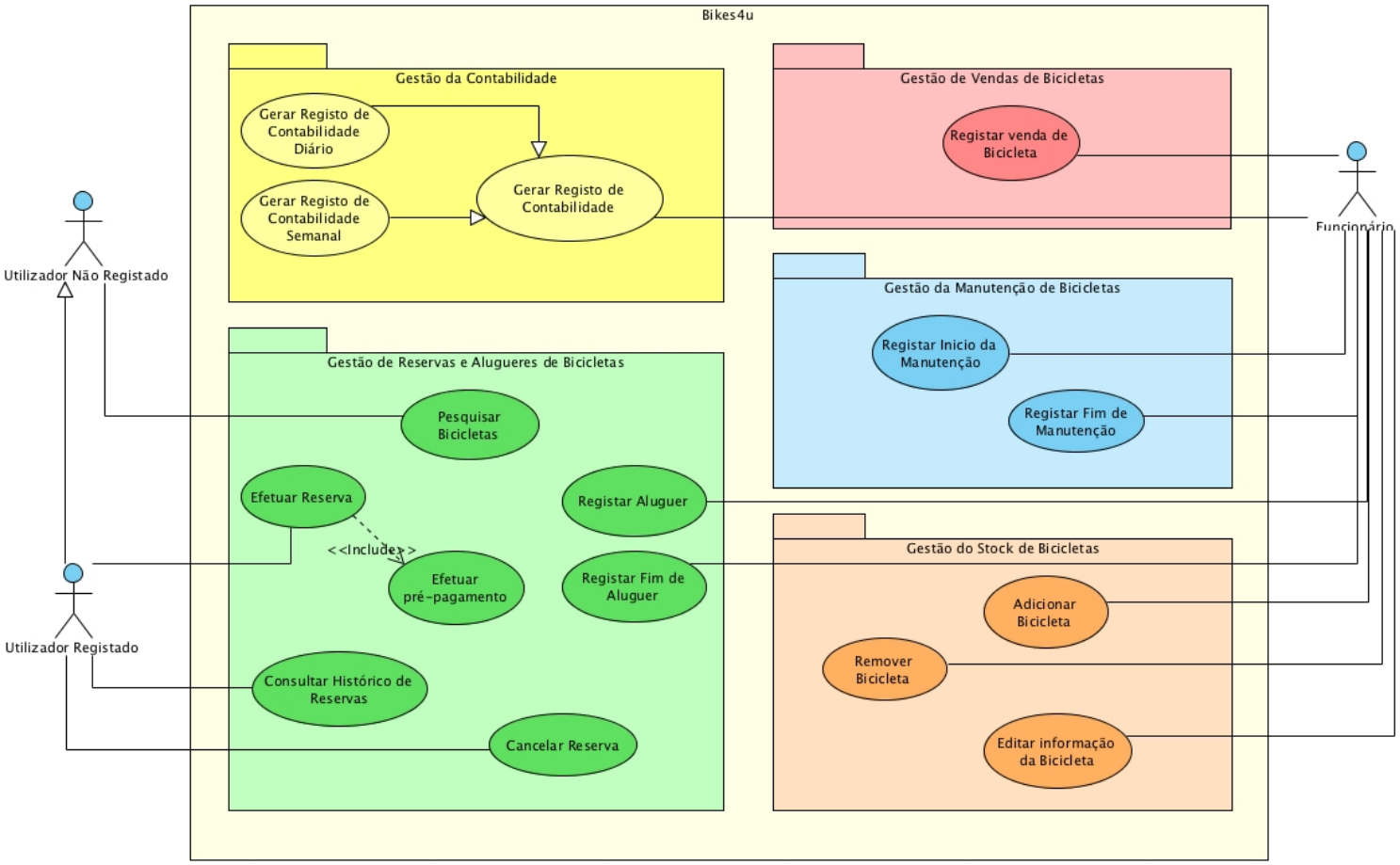

Figure 1. Diagrama de Use Cases

A divisão em packages foi efetuada tendo como base os módulos de gestão do negócio, de acordo com o enunciado do projeto:

1. Módulo de reserva de aluguer de bicicletas;
2. Módulo de gestão efetiva de alugueres com registo contabilístico diário e mensal;
3. Módulo de manutenção de bicicletas;
4. Módulo de gestão de vendas e gestão de stocks de bicicletas novas ou em 2ª mão.

Após compreendermos em que consiste cada um destes módulos, o tipo de informação que gerem e as operações envolvidas nessa gestão, concluímos que alguns deles reúnem duas componentes distintas que podem ser separadas. É o caso do segundo módulo, que reúne a gestão de alugueres e o registo contabilístico; e do quarto módulo, uma vez que agrupa dois tipos de funcionalidade distintas, a gestão de vendas e a gestão de stocks de bicicletas.

Dado que uma reserva é a atividade que antecede um aluguer, resolvemos criar um package de gestão de reservas e alugueres, já que estão fortemente interligadas.

Por fim, o módulo de manutenção está perfeitamente adequado para dar origem ao package respetivo — manutenção de bicicletas.

Assim, os packages que identificamos são os seguintes:

- Gestão de Reservas e Alugueres de Bicicletas;
- Gestão de Vendas de Bicicletas;
- Gestão da Contabilidade;
- Gestão da Manutenção de Bicicletas;
- Gestão de Stock de Bicicletas.

2.1 Gestão de Reservas e Alugueres de Bicicletas

Uma das funcionalidades essenciais da aplicação é a possibilidade de um cliente efetuar uma reserva de uma ou várias bicicletas, garantindo a disponibilidade dessa(s) bicicleta(s) no período escolhido. Para isso, o cliente deverá estar registado.

Durante o processo de reserva, o cliente terá que escolher a(s) categoria(s) que lhe interessar, e indicar a respetiva quantidade. Para que este processo seja concluído, o cliente deverá proceder ao seu pré-pagamento. Um reserva pode ser cancelada até 24horas antes da data de início do aluguer.

A reserva de um cliente no site dará, à partida, origem a um aluguer. O registo efetivo de um aluguer é feito pelo funcionário da loja, estabelecendo as bicicletas que são alugadas, tendo em conta os detalhes da reserva.

O cliente deverá deslocar-se à loja para ir buscar a(s) bicicleta(s) reservadas. Naturalmente, estas passam a estar indisponíveis para qualquer requisição. No final do período do aluguer, quando cliente a(s) devolver, o funcionário regista que o aluguer chegou ao fim. As bicicletas passam a estar novamente disponíveis.

2.2 Gestão de Vendas de Bicicletas

O negócio desta empresa não se baseia apenas em alugueres de bicicletas. Existem também uma vertente de comercialização deste produto, e como tal um funcionário poderá registar a ocorrência da venda de uma bicicleta a um dado cliente.

2.3 Gestão da Contabilidade

Para uma melhor gestão da tesouraria da empresa, o grupo decidiu que a contabilidade deverá reunir os detalhes dos alugueres e das vendas realizadas. A

aplicação permite gerar relatórios contabilísticos, que possibilitam controlar de uma forma mais eficaz os lucros da empresa.

2.4 Gestão da Manutenção de Bicicletas

Visto que o negócio central da empresa se concentra no aluguer e comercialização de bicicletas, é essencial que esta garanta a correta manutenção dos seus artigos. Por isso, no caso de uma bicicleta estar danificada, é possível envia-la para reparação.

Quando o funcionário envia uma bicicleta para a manutenção, a data de início é automaticamente registada no sistema como sendo a data desse mesmo dia. Nestas condições, a bicicleta fica indisponível, tanto para efeitos de venda como aluguer.

Na manutenção, são registadas as várias operações de reparação, o seu custo e ainda o período de reparação.

Quando a manutenção terminar, o funcionário é responsável por registar seu o término, passando a bicicleta a estar disponível na loja para novas requisições.

2.5 Gestão de Stock de Bicicletas

O funcionário é responsável por controlar o stock de bicicletas. Esta operação deverá ser permanentemente efectuada, para que o stock seja controlado de uma forma constante, e se encontre devidamente atualizado, garantindo assim a total satisfação de alugueres e vendas.

Deste modo, o funcionário pode adicionar uma nova bicicleta, editar a sua informação, ou ainda mantendo-a irreversivelmente indisponível, caso este já não exista definitivamente.

3 Diagrama de Classes

A partir do diagrama de Classes obtemos uma representação da estrutura e relacionamentos das classes que servem de modelo para os objetos do sistema. Para cada classe identificamos os seus atributos.

Nesta fase tomamos importantes decisões de implementação, que estão descritas a seguir.

Para termos a informação atualiza do estado em que a bicicleta se encontra, isto é, para sabermos se, num dado momento, uma determinada bicicleta está, por exemplo, disponível, em manutenção, ou alugada, existe uma variável *estado* associada à bicicleta que nos fornece essa informação. Esta variável pode tomar os seguintes valores:

- Disponível – a bicicleta está na loja e disponível para eventuais requisições;
- Alugada – a bicicleta foi alugada a um dado cliente;
- Em manutenção – a bicicleta está a ser reparada;
- Vendida – a bicicleta foi vendida.

Quando é efetuada uma reserva, o cliente define o período de aluguer e escolhe a(s) categoria(s) da(s) bicicleta(s) que pretende, de uma lista de bicicletas disponíveis nesse intervalo de tempo.

Uma reserva tem o período mínimo de 24horas, pelo que o preço consiste no preço de aluguer por dia. O preço de aluguer depende unicamente da categoria da bicicleta. A diferentes categorias de bicicletas correspondem diferentes preços de aluguer.

O diagrama que se segue ilustra o diagrama de classes desenvolvido.

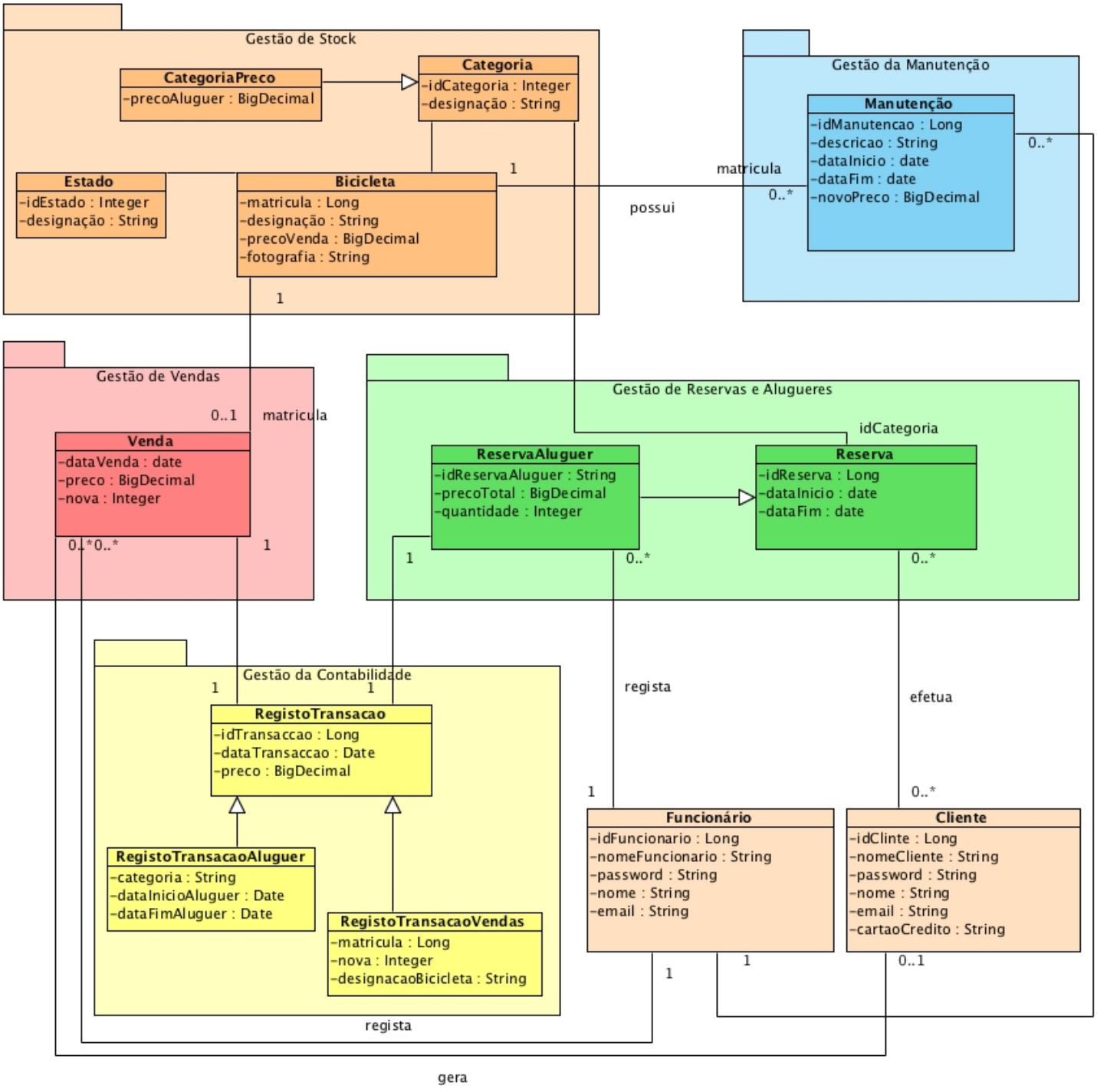

Figure 2. Diagrama de Classes

4 Implementação

4.1 Fluxo de controlo para implementação dos Servlets

Servlets são programas Java que correm em servidores *java-capable*, funcionando como uma camada intermédia entre pedidos do browser e a base de dados.

Para a implementação dos nossos servlets utilizamos os métodos doX(), responsáveis por definir o comportamento correspondente a cada HTTP *request* do browser. Assim, num HttpServlet cada tipo de HTTP *request* é convertido na invocação do método específico que irá tratar o pedido.

De seguida apresentamos o fluxo de controlo em que nos baseamos para implementar alguns dos Servlets, para 5 das operações mais importantes.

4.1.1 Adicionar uma nova Bicicleta

Para adicionar uma nova bicicleta, o funcionário faz o request, que é tratado pelo RedirectAdicionarBicicleta. Este servlet faz redirect para a ListaBikeStock.jsp, apresentando a lista de bicicletas existentes e um formulário, que o funcionário terá que preencher com os dados da nova bicicleta.

Após o funcionário escolher a opção “OK”, o servlet NovaBikeAction faz forward para a lista de bicicletas ListaBikeStock.jsp. Em caso de erro, pela existência de campos em por preencher, o servlet reencaminha para a página de erro (erro.jsp).

Usamos *forward* para respostas mais rápidas, em que atualização da página resultante é simplesmente repetir o pedido original. Quando realmente é necessário guarantir proteção dos dados, como é o caso da ação envolver uma operação na base de dados, utilizamos *redirect*. Os objetos do *request* original já não estarão disponíveis no segundo pedido, evitando que, por exemplo, seja depenhada novamente a mesma operação sobre a base de dados.

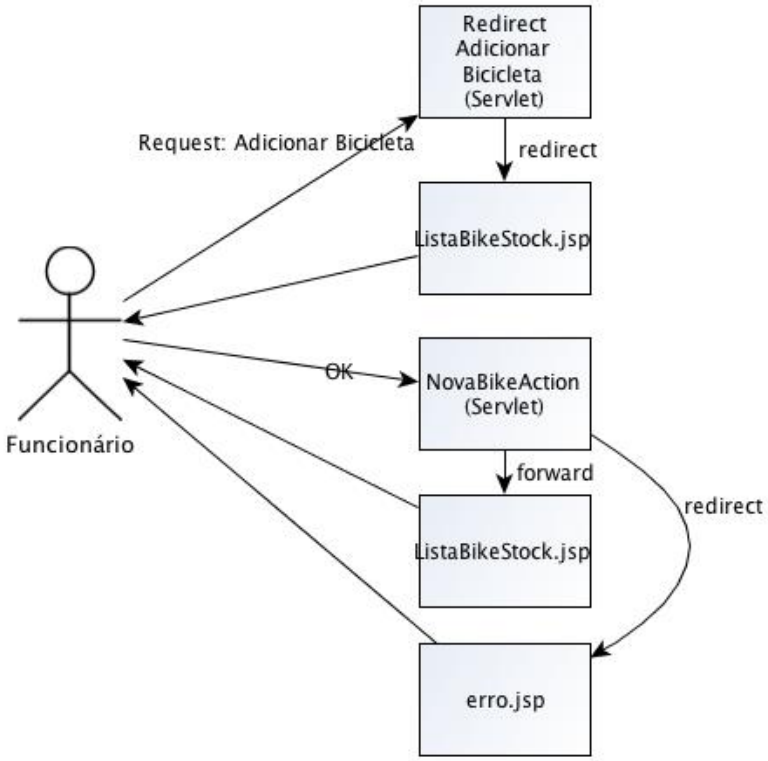

Figure 3. Fluxo de controlo Adicionar Bicicleta

4.1.2 Efetuar uma Reserva

Na operação de efetuar uma reserva de bicicleta por um cliente, o servlet EscolherAluguerAction reencaminha a respetiva página. O utilizador deverá definir o período de tempo da reserva e escolher "ver disponibilidade" das bicicletas que existem nesse espaço de tempo. O servlet ReservasAction encaminha para uma página que apresenta uma lista das categorias das bicicletas disponíveis. O cliente deverá escolher a categoria e a quantidade, e por fim selecionar "Reservar". O servlet NovaReservaAction é responsável por registar a nova reserva na base de dados.

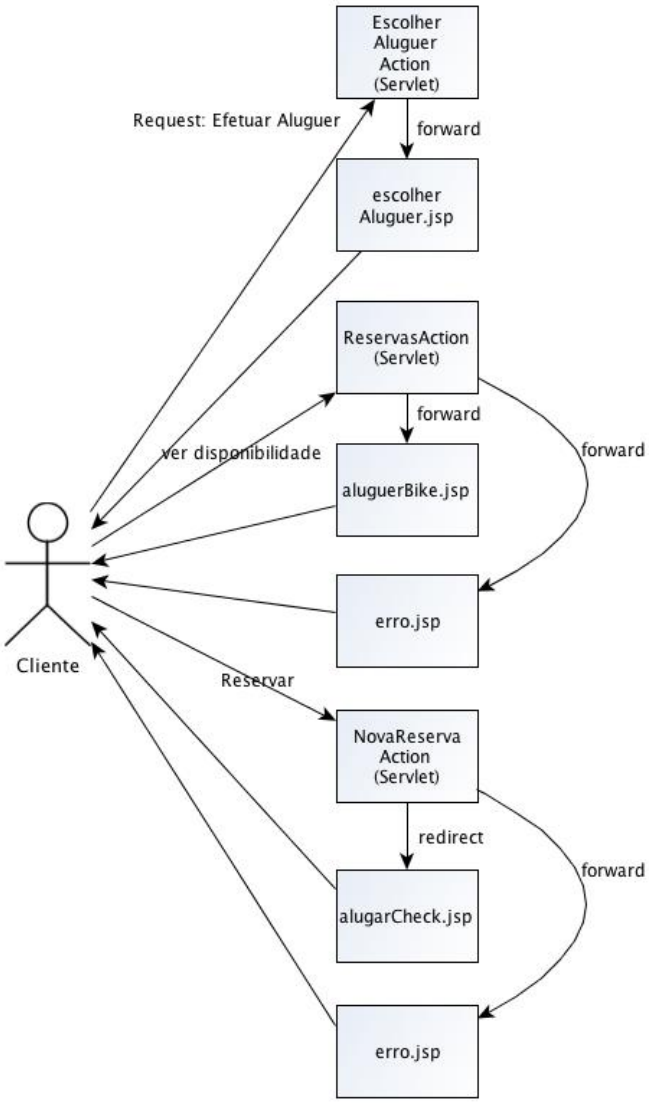

Figure 4. Fluxo de controlo Efetuar uma reserva

4.1.3 Registar Aluguer

Para registar um aluguer, o servlet ListaReservaAction encaminha a página da lista de reservas efetuadas. O funcionário define qual o aluguer que pretende oficializar (através da seleção da reserva que pretende). Por sua vez, o servlet GestãoReveraAluguerAction encaminha a página de confirmação do aluguer.

O servlet RegistarReservaAluguerAction é responsável por registar o aluguer na base de dados. De seguida reencaminha para o servlet que irá apresentar novamente a lista das reservas em curso. Caso não seja possível registar o aluguer, o erro é anunciado ao funcionário.

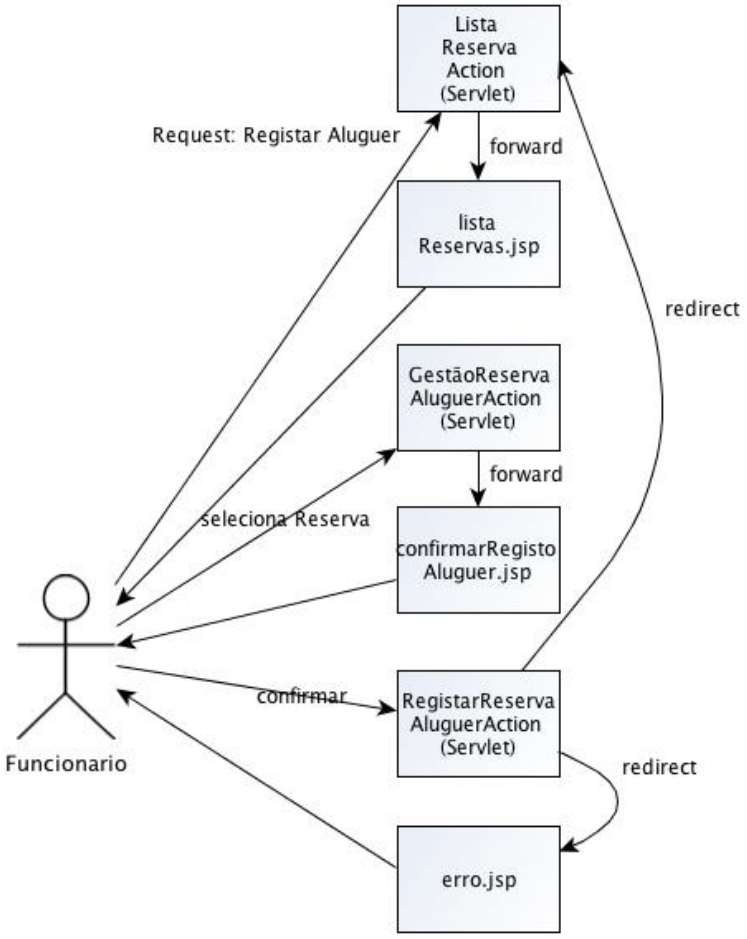

Figure 5. Fluxo de controlo Registar Aluguer

4.1.4 Registar Venda

Para proceder ao registo de uma venda, o servlet BicicletasArmazemAction reencaminha a página BikesArmazem.jsp, que contém todas as bicicletas que se encontram na loja.

O funcionário deverá escolher a bicicleta sobre a qual pretende efetuar o registo de venda. O servlet VenderBicicletaAction encaminha para a página de registo efetivo da venda, onde se define o cliente. O servlet RegistoVendaAction é responsável pelo registo da venda dessa bicicleta, a esse mesmo cliente. Por fim reencaminha novamente para o servlet que, por sua vez, apresenta a página das bicicletas que estão na loja.

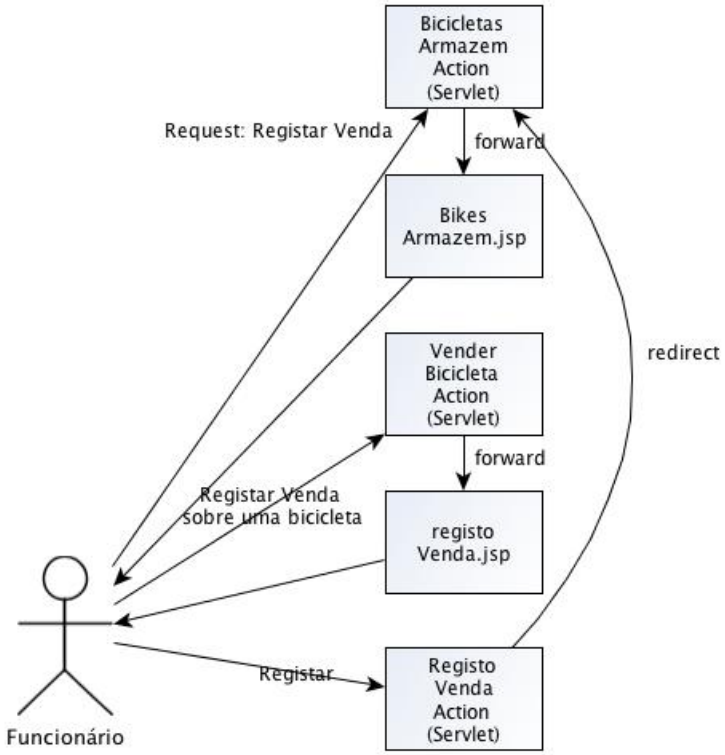

Figure 6. Fluxo de controlo Registar Venda

4.1.5 Registar início da Manutenção

Para assinalar o início de uma manutenção, o servlet BicicletasArmazemAction é responsável por apresentar a lista das bicicletas que se encontram na loja. O funcionário deverá escolher a bicicleta sobre a qual pretende efetuar o registo da manutenção.

O servlet FormManutençãoAction encaminha para a página de registo efetivo da manutenção, onde se descrevem os seus detalhes. O servlet RegistarManutençãoAction é responsável pelo registo da manutenção dessa bicicleta. O estado da bicicleta passa então a ser "em manutenção". Por fim reencaminha novamente para o servlet que, por sua vez, apresenta a página das bicicletas que estão na loja.

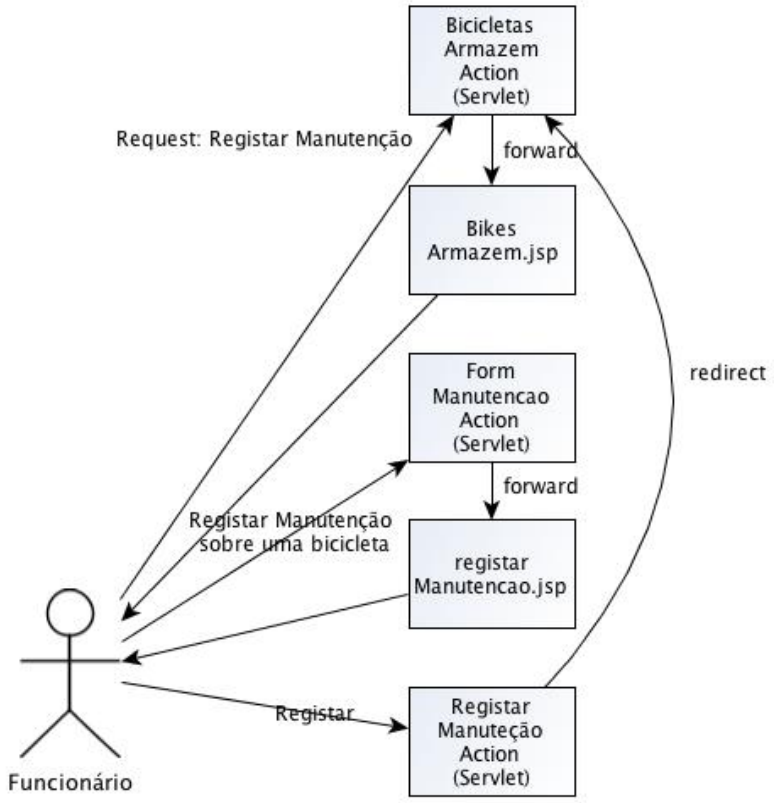

Figure 7. Fluxo de controlo Registar inicio da Manutenção

4.2 Arquitetura global da aplicação

Como já foi referido anteriormente, a nossa aplicação integra tecnologia de Servlets e JSP e, como é natural, a sua arquitetura reflete essa mesma prática.

O modelo MVC está presente na organização desta arquitetura, e como tal foi necessário ter em conta um conjunto de procedimentos aprendidos nas aulas deste módulo, para que o resultado final ficasse devidamente estruturado.

Deste modo, as páginas JSP são utilizadas na camada de apresentação, e por isso todos as *views* da aplicação foram implementadas somente utilizando esta tecnologia.

Por sua vez, os Servlets assumem o papel de controladores, sendo responsáveis por pelo tratamento de pedidos ao sistema. A implementação de Servlets está ausente de qualquer código HTML, assegurando a total separação entre as camadas de negócio e apresentação. É também da responsabilidade dos Servlets o encaminhamento das páginas JSP corretas para satisfazerem os pedidos que chegam ao sistema.

Por fim, os *models* consistem em classe JavaBeans, que são classes especiais que possuem o construtor (vazio), implementam a classe *serializable* e são permitem o acesso aos objetos usando os métodos *get* e *set*.

A figura seguinte ilustra a arquitetura da nossa aplicação:

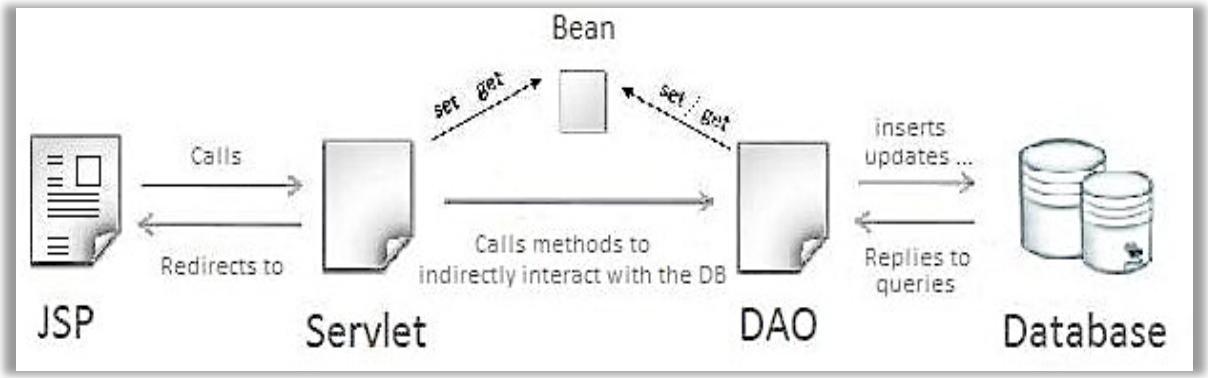

Figure 8. Arquitetura da aplicação

4.3 Considerações sobre a persistência de dados

Para conseguirmos um encapsulamento dos dados durante o acesso à base de dados, recorremos à utilização do *desing pattern* DAO, que nos oferece um mecanismo de acesso a dados para abstrair os detalhes da implementação de persistência numa aplicação.

A camada de DAOs impede que se realize uma comunicação direta entre a camada lógica e a base de dados, assumindo essa tarefa.

Cada DAO é responsável pelas operações CRUD de um objeto. O envio das queries SQL à base de dados é feito através de JBDC.

5 Conclusão

Após terminado este trabalho prático, há algumas considerações que podem ser mencionadas.

O primeiro passo para o seu desenvolvimento consistiu na identificação das principais operações envolvidas na aplicação. A criação do diagrama de use cases foi uma tarefa mais ou menos intuitiva, após a análise e interpretação atenta do enunciado, sucedendo a organização dos use cases em packages.

O diagrama de classes, por sua vez, gerou algumas discussões entre o grupo. Optamos assim por criar um esquema de classes que nos permitisse implementar as funcionalidades desejadas de forma eficaz.

Para a implementação dos Servlets, o desenho do controlo de fluxo das operações mais complexas foi bastante útil, na medida em que nos permitiu perceber quais os servlets envolvidos numa dada tarefa.

Durante a implementação do projeto, tivemos sempre em consideração a arquitetura definida para o sistema, respeitando sempre as regras de implementação do modelo MVC. O resultado final obtido consiste numa aplicação sólida, com uma arquitetura bem definida e uma forte independência entre as suas camadas estruturais.

Embora a aplicação desenvolvida desempenhe todas as funcionalidades exigidas, existem algumas falhas de implementação que é necessário referir, como é o caso de não recorrermos à utilização de Filters para controlar o acesso às páginas privadas e não existir a opção de escolha do idioma das páginas.