Alocação de equipamento em DIF2

Relatório do Trabalho Final de Curso

Aluno: Pedro da Costa Campos Correia

Orientador: Professor Sérgio Guerreiro.

Agradecimentos

Em primeiro lugar ao professor Sérgio Guerreiro pelo apoio e orientação prestada ao trabalho.

Ao Departamento de Sistemas de Informação pelo conhecimento e material de estudo disponibilizado para desenvolvimento do trabalho.

Por último gostaria de agradecer aos meus colegas e restantes professores do curso por todo o seu apoio ao longo destes meses de trabalho.

Resumo

Este relatório tem como objectivo descrever todo o processo envolvido no desenvolvimento deste projecto de Trabalho Final de Curso.

O seu objectivo consistiu no planeamento e desenvolvimento de uma aplicação de alocação de equipamento, fazendo uso da tecnologia DIF 2.

Este documento encontra-se dividido em 8 capítulos onde são descritas as várias fases de desenvolvimento e planeamento deste projecto.

O primeiro capítulo possui um glossário onde é explicado palavras e termos pouco comuns usados ao longo do relatório.

O segundo capítulo é a introdução onde, de forma sucinta, é explicado o objectivo deste projecto e o seu enquadramento actual. Este capítulo encontra-se dividido em 5 subcapítulos: propósito do documento, âmbito do projecto, situação actual, aplicação no âmbito da engenharia de software e tecnologias usadas.

O terceiro capítulo descreve os requisitos de forma detalhada da solução, desde os seus intervenientes a uma descrição mais detalhada dos requisitos.

O quarto capítulo é reservado à análise e modelação do sistema, é nesta fase onde os requisitos são formalizados via modelos visuais e analisados sob várias perspectivas.

O quinto capítulo descreve o processo de desenvolvimento. Desde uma descrição mais abrangente até a uma análise componente a componente, este capítulo descreve a parte onde mais tempo foi dispendido neste projecto. Divide-se em 5 subcapítulos: Descrição, arquitectura, componentes, testes e integração.

O sexto capítulo é dedicada à conclusão, aqui é descrito o resultado, as suas vantagens e desvantagens.

Nos 2 últimos capítulos, sétimo e oitavo, encontram-se a bibliografia e os anexos relevantes ao projecto.

Abstract

This report describes the whole process involved in the development of this Final Course Project.

The purpose of this project was the planning and development of an application of equipment reservation using the framework DIF 2.

This document is sectioned in 8 chapters, where each process of the development of this project is described in detail.

The first chapter contains a glossary with unusual terms that are shown throughout this document.

The second chapter is the introduction that, succinctly speaking, describes the purpose of this project and it's the scope. It is divided in 5 subchapters: the purpose of the document, the scope, the current situation, the project in the context of software engineering and the technologies that were used.

The third chapter contains a thorough description of the requirements, the stakeholders and a more detailed analysis of the requirements.

The fourth chapter is dedicated to the analysis and modeling of the system, in this phase, requirements are formalized through the use of diagrams.

The fifth chapter describes the process regarding development process. The description is made from a more comprehensive to a more detailed analysis of each component. The chapter is composed of 5 subchapters: Description, architecture, components, tests and integration.

The sixth chapter is dedicated to the conclusion of this work, here is explained the results, advantages and disadvantages.

The last 2 chapters, seventh and eighth, contain the bibliography and annexes relevant to this project.

Índice

Índice de ilustrações

1 Glossário

AJAX – Asynchronous Javascript And XML, conjunto de tecnologias que permite a troca de informação assíncrona entre cliente e servidor.

Bean – Componente de software reutilizável que respeita certas nomenclaturas e convenções de desenho.

DAO – Data Access Object, classes cujo propósito é fazer a ligação entre a base de dados e a aplicação.

Framework – Estrutura de suporte em que um projecto de software pode ser organizado e desenvolvido.

IDE – Integrated Development Environment, ambiente para o desenvolvimento de software.

J2EE – Java Enterprise Edition, ambiente de desenvolvimento Java para aplicações Web, composta por serviços, APIs e funcionalidades próprias.

JSP – Java Server Pages, extensão às servlets que combina processamento do lado do servidor com HTML.

MVC – Model View Controller, arquitectura de desenvolvimento que visa separar as camadas de lógica, apresentação e dados.

ORM – Object Relational Mapping, técnica usada para integrar o paradigma Object Oriented com um modelo relacional.

Packages – pacotes usados pelo java para a organização de classes.

Postback – Evento que ocorre quando uma página, após a submissão, é redireccionada para ela própria.

RAD – Rapid Application Development, modelo de desenvolvimento que enfatiza o período de desenvolvimento curto.

Rollback – Reverter as alterações feitas na Base de dados.

Servlet – Componente do lado do servidor, é uma classe Java com capacidade para processar pedidos e resposta de HTML e XML dinamicamente.

SoC – Separation of Concerns, separação de um programa por funcionalidades que não se sobreponham umas às outras.

XML – eXtensible Markup Language, linguagem de estruturação para dados.

2 Introdução

2.1 Propósito do documento

Este documento descreve, em detalhe, o trabalho realizado para o desenvolvimento da aplicação de alocação de reserva e o uso das funcionalidades da DIF 2. Este projecto foi concebido para responder às necessidades dos docentes.

2.2 Âmbito do projecto

Este projecto inseriu-se no âmbito da disciplina do Trabalho Final de Curso (TFC), sendo esta uma componente essencial do curso, onde o aluno demonstra os conhecimentos adquiridos ao longo da licenciatura.

2.3 Situação actual

A faculdade possui actualmente o equipe (aplicação de alocação de equipamento já existente) que, de acordo com a docência, possui pouca flexibilidade, nomeadamente:

- A ausência de alteração de reserva após esta ter sido criada.
- A ausência de desmarcação de uma reserva.
- Marcação de individual de reserva.

Aliado a estes problemas encontra-se o facto de esta solução não estar integrada no restante sistema, dificultando ainda mais a coerência dos dados entre ambos.

2.4 Engenharia de software

Este projecto foi concebido seguindo os princípios da engenharia de software. Para tal foi estruturada as etapas do desenvolvimento, os seus intervenientes, escolha e atribuição de metodologias usadas e a calendarização do trabalho.

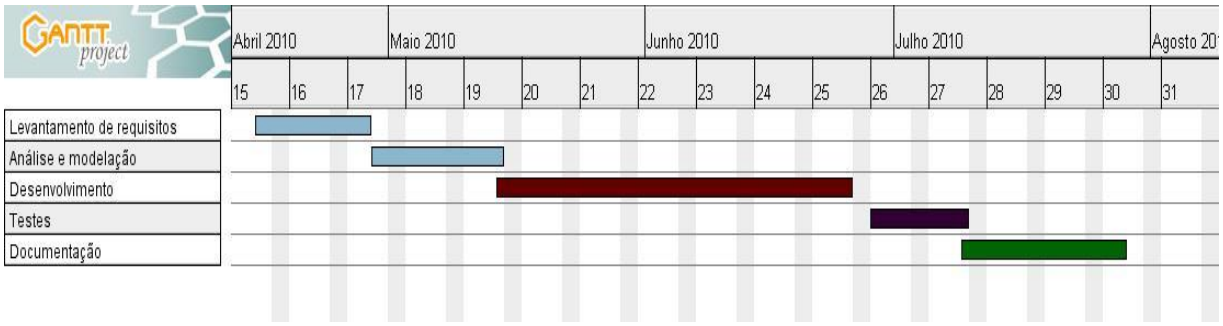

Ilustração 1 – Mapa de Gantt do projecto.

A calendarização do projecto permitiu uma melhor gestão do tempo dispendido em cada fase do trabalho, a percentagem do tempo é proporcional ao esforço requerido em cada uma das fases, a divisão escolhida foi a seguinte:

- O levantamento de requisitos teve a duração de 12 dias.
- A análise e modelação tiveram a duração de 15 dias.
- O desenvolvimento teve a duração de 28 dias.
- Os testes tiveram a duração de 10 dias.
- A documentação teve a duração de 14 dias.

Para este projecto foi utilizado o modelo de prototipagem. Este modelo permitiu o desenvolvimento de vários protótipos, sendo incrementalmente melhorados a cada versão. Este modelo permitiu também para fazer alterações ao desenho e até mesmo redefinir alguns requisitos.

2.5 Tecnologias usadas

2.5.1 DIF 2

Framework para J2EE orientada para RAD. A Framework é composta por inúmeros plugins assim como uma configuração via anotações em contraste aos ficheiros XML.

A Framework utiliza um modelo MVC, onde os JSP servem de View, os Stages, de controller e os Beans de Model. A hierarquia de uma aplicação na DIF é composta por: Provider, Application, Service e Stage.

2.5.2 PostgreSQL

Base de dados opensource usada para este projecto, a escolha deste deve-se ao facto de a base de dados existente na plataforma Netp@ ser a mesma.

2.5.3 Maven

Ferramenta de automação para projectos de Java. Permite a gestão de dependências associadas ao projecto.

2.5.4 Eclipse

IDE distribuído pela Digitalis já configurada para o uso da DIF.

2.5.5 StarUML

Aplicação usada para a modelação na fase da análise usando os diagramas UML.

2.5.6 Sybase PowerDesigner 15

Aplicação de modelação usada para desenhar o modelo de dados e a geração do código SQL do mesmo.

2.5.7 Jetty

Pequeno servidor Web que foi utilizado para o teste directo da aplicação, devido à sua rápida execução e configuração por parte do Maven foi possível assim um lançamento da aplicação em curto espaço de tempo.

3 Levantamento de requisitos

3.1 Stakeholders

Os stakeholders são o grupo de pessoas a quem a aplicação vai, directamente ou indirectamente influenciar. Cada membro dos stakeholders vai, de uma forma ou de outra, validar cada requisito estabelecido da aplicação.

3.2 Requisitos

Os requisitos constituem a parte mais crítica do sistema e estes encontram-se divididos em 2 categorias, funcionais e não funcionais, cada requisito foi analisado a fundo, permitindo desta forma justificar o seu aparecimento na solução.

Estes requisitos foram sempre revistos ao longo do processo de desenvolvimento para garantir que o produto que se estava a desenvolver correspondia ao pedido, assim como validar se os requisitos poderiam ser incorporados na aplicação em tempo útil.

3.2.1 Requisitos funcionais

Os requisitos nesta secção foram divididos em requisitos obrigatórios e requisitos opcionais. Os primeiros são aqueles cujo sistema obrigatoriamente teria que ter para ser aprovado, os opcionais são requisitos que, ao adicionados, adicionam mais valor à aplicação.

3.2.2 Requisitos não funcionais

Os requisitos não funcionais são aqueles influenciam a implementação do sistema, por outras palavras impõem restrições que o sistema tem que, forçosamente, cumprir.

3.2.3 Implementação de requisitos

Todos os requisitos de relevo ao sistema encontram-se presentes na aplicação. Este sistema permite reservas recursivas, mais concretamente reservar o mesmo equipamento para a mesma sala durante um período de N semanas. A marcação para múltiplas horas e locais no dado dia também encontra-se implementado assim como a alteração, desmarcação e listagem de reservas.

Dos requisitos opcionais encontra-se implementado a lista de reservas ao funcionário, tarefa possível graças ao facto esta tabela ser dinamicamente gerada e a organização e tipos de dados mostrados poderem ser alterados sem mudança por parte do servidor.

O segundo requisito opcional, possibilidade de marcação de reservas por parte do funcionário, acabou por não ser implementado, uma vez que o calendário estipulado no plano inicial ter sido seguido à risca, o tempo restante seria insuficiente para implementar esta funcionalidade, uma vez que teriam que ser feitas alterações ao próprio modelo de dados e à funcionalidade de criação de reserva existente.

4 Análise e modelação

4.1 Casos de uso

Diagrama que permite descrever a funcionalidade do ponto de vista do utilizador, tem como objectivo a validação dos requisitos funcionais.

4.2 Modelo de dados

Diagrama que mostra a estrutura de dados que será implementada em PostgreSQL.

4.3 Diagrama de classes

Diagrama que mostra a estrutura de classes usadas na aplicação, componente de uma análise estática.

4.4 Diagrama de sequência

Diagrama com o objectivo de mostrar sequência temporal da funcionalidade do sistema, assim como a sua relação com as restantes classes.

4.5 Diagrama de Componentes

Diagrama que mostra o sistema como um conjunto de componentes e como estes comunicam entre si.

5 Desenvolvimento

5.1 Descrição do projecto

Este projecto resultou numa aplicação Web onde as funcionalidades foram implementadas de acordo com a informação especificada no levantamento de requisitos.

A aplicação é usada por 2 tipos de utilizadores, docente e funcionário. O primeiro tem à sua disposição o menu de criação de reserva e, na página de listagem de reserva, pode ver as suas reservas, marcadas, canceladas e levantadas, assim como escolher editar uma em específica ou então desmarcar uma ou mais alocações. O segundo pela sua vez só tem acesso ao menu de listagem, contudo diferente do docente, onde vê apenas as reservas para o dia actual, podendo marcar uma reserva a um equipamento como levantado.

5.2 Arquitectura da aplicação

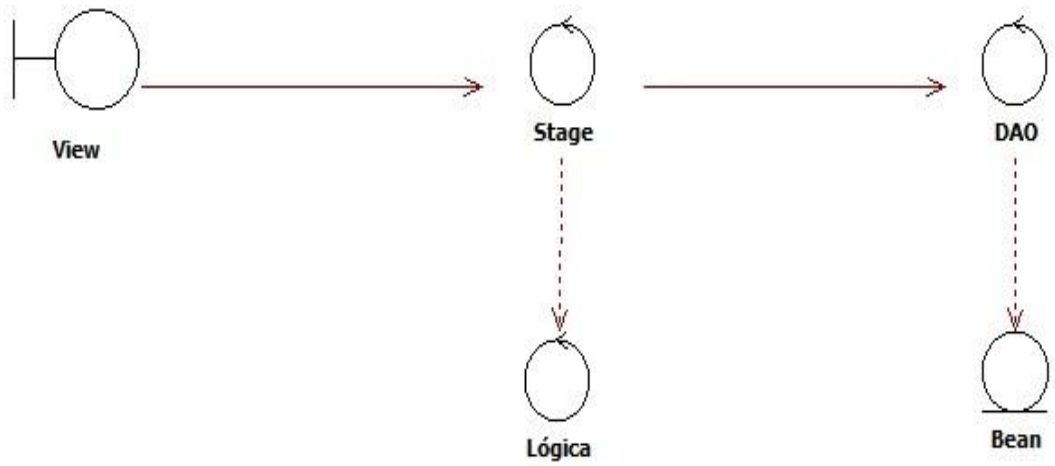

Ilustração 2 – Arquitectura MVC

A arquitectura utilizada para este projecto foi a MVC, este modelo permite uma melhor separação das várias camadas da aplicação. A estrutura dos projectos da DIF seguem este modelo com os JSP(views), stages(controllers) e beans com anotações (Model).

Na view são mostrados os dados da aplicação, cada view está associado a uma stage. A camada de controlo é composto por 2 tipos de classes, a primeira são as stages que recebem o comando da view e fazem todo o processamento dos dados e comunicação com a camada de dados. O segundo tipo de classes do controlador é as classes de lógica, estas validam a informação relacionada com o negócio propriamente dito, permitindo adoptar uma SoC.

Por último são as classes do modelo que incluem os beans e DAOs, que representam os dados e a sua persistência.

5.3 Componentes

O desenvolvimento foi feito com base em componentes, de forma a simplificar a sua organização, as várias componentes foram particionados em 7 packages distintos, sendo a organização escolhida a seguinte:

- AJAXServlets
- Lógica
- Beans
- Stages
- DAO
- Views
- Utils

Para uma exemplificação mais clara usou-se um diagrama de componentes, podendo assim analisar, não só a forma como os devidos componentes foram organizados mas também as suas dependências.

5.3.1 AJAXServlets

Neste package encontram-se as servlets que tratam dos pedidos assíncronos usando o XML como linguagem de transporte. A necessidade associada ao uso directo de servlets é justificada pela limitação das soluções AJAX providenciadas pela Framework, que acabou por ser esta a alternativa mais fiável.

As chamadas assíncronas são realizadas pelas classes AJAXServSala e AJAXServReserva, onde cada uma serve como servlet de informação assíncrona para os dados das respectivas entidades.

5.3.2 Lógica

Este package tem as interfaces e classes com as regras do negócio. O uso da interface permite que possam ser adicionados novos tipos de validações ou regras a uma marcação de reserva, o que permite assim uma maior modularidade. Para além das classes de validação encontra-se uma classe com constantes que são usadas tanto pelas Stages como pelos métodos de validação.

Nesta package pode ser encontrado 2 tipos de classes de validações, uma de validação de parâmetros, gere se os parâmetros estão de acordo com as regras estabelecidas, este tipo de validação é igualmente feita no lado do cliente via javascript.

O segundo tipo de validação consiste nas próprias regras do negócio, como as horas a que uma reserva pode ser feita (ex: horário de funcionamento da universidade).

5.3.3 Beans

Este package possui os modelos usados pelo Hibernate como referência às tabelas na base de dados, sendo estes modelos beans em java.

Outra propriedade nos modelos é o facto de estes possuírem anotações que permitem, em associação com os ficheiros de mapeamento, estabelecer ligação entre as classes e as devidas tabelas, assim como as suas relações.

Os ficheiros beans encontram-se associados a ficheiros de mapeamento com os mesmos nomes e extensão hbm.xml, estes últimos possuem as configurações e relações de entre atributos, colunas e tipos de dados com o modelo de dados.

Tantos os ficheiros de mapeamento, classes, relações e configurações foram gerados automaticamente a partir do modelo de dados já implementado em BD usando as funções de reverse engineering do IDE Eclipse com o plugin hibernate tools. Os ficheiros são automaticamente colocados na pasta de recursos onde a DIF guarda os seus ficheiros de configurações.

Os ID de todas as tabelas com a excepção de utilizadores, foram configuradas para usarem o sistema de auto-increment, alteração justificada uma vez que os ID possuem o único objectivo de distinguirem os registos uns dos outros.

5.3.4 DAO

Todas as classes DAO possuem o seu bean equivalente. Possuem os métodos que fazem o acesso à base de dados propriamente dita.

Sempre que um DAO é instanciado, este abre uma ligação à base de dados de forma concorrente, esta funcionalidade é possível devido à classe HibernateUtil, esta é chamada pelo construtor da classe, inicializando assim uma sessão JDBC.

Todas as classes DAO utilizam a Query By Criteria como linguagem de query à BD, esta escolha foi feita com base no facto de esta ser bastante mais flexível na adição que critérios de pesquisa, funcionalidade que torna-se vantajosa para queries dinâmicas, esta característica acaba por compensar o facto de sempre que requer-se somente uma coluna, o método Criteria carrega o registo todo.

Sempre que é feita uma alteração ou inserção de uma reserva e existe uma falha de ligação ou outro tipo que impossibilite a query de ser executada, o DAO executa um rollback, este método previne que haja informação não coerente na base de dados.

5.3.5 Stages

As stages representam o núcleo da aplicação, é aqui que todo o sistema é controlado, neste package encontram-se 6 stages, estas usam anotações de Java para notificar à Framework a qual view está associada, a que grupos de utilizadores está acessível ou mesmo definir os métodos pertencentes às servlets (init, destroy).

A stage Erro tem o único objectivo de servir de página de redireccionamento quando se tenta aceder a uma zona ilegal, esta stage é usada pelas restantes, com a excepção da HomeStage, sempre que existe algum problema ou excepção.

A stage de Sucesso é só utilizada em 2 ocasiões, durante a criação e alteração de reserva com êxito, nesse caso as stages CriaReserva e EditaReserva adicionam as devidas mensagens de sucesso à stage da mesma que é acedida via EL na View.

As restantes 4 stages possuem todo o processamento da solução, sendo detalhadas individualmente devido à sua complexidade.

5.3.5.1 Criação de reserva

A stage CriaReserva é a responsável pela marcação de uma nova reserva, quando esta é invocada é executada uma função de inicialização, a primeira coisa a verificar é a existência de um objecto DIFSession, objecto instanciado caso alguém esteja activo no sistema, se o utilizador existir e for docente pode prosseguir, caso contrário redirecciona para a stage de erro.

Com o acesso validado o utilizador passa a ter acesso à página de reserva, a criação de reserva envolve 2 passos distintos. O primeiro passo é o preenchimento dos detalhes gerais da reserva como o utilizador (automaticamente registado pela stage), data para a reserva, equipamento e a marcação recursiva, esta última permite ao utilizador a marcação para a mesma dia da semana, o mesmo equipamento e horas durante as N semanas que se seguem, sendo 1 o valor por defeito, ou seja será marcada somente uma vez. O segundo passo é a alocação, mais concretamente o período de tempo e o local, o utilizador pode escolher até no máximo 5 alocações, sendo adicionados os campos de alocação dinamicamente.

Quando os dados são submetidos os parâmetros dinâmicos são guardados em lista. Nesta fase a validação é feita em 2 ciclos, o primeiro é por cada alocação que foi criada, o segundo por cada semana, cada campo é validado individualmente e cada processo de validação é registado no caso de erro numa lista somente para essa função.

A validação é composta por 2 partes, a primeira valida se o valor no parâmetro é válido ou não, o segundo valida as próprias regras para marcar uma reserva, validações feitas usando as classes do package Lógica.

As validações vão desde possíveis colisões com outras reservas ao mesmo, equipamento, conflito entre alocações da mesma reserva, período mínimo para a requisição do equipamento entre outras que podem ser adicionadas.

Se existir alguma entrada na lista de erros, sinal que alguma validação falhou, então é carregada a página de novo, para evitar que as componentes dinâmicas e o seu conteúdo se percam durante o postback, todos os dados são armazenados em atributos do contexto. Quando a página é iniciada se detectar um atributo postback as componentes e os seus valores são repostos durante a inicialização da página, juntamente é adicionado a lista de erros de notificação para o utilizador.

Caso a validação tenha sido positiva, é criado uma reserva por cada alocação existente e estas são inseridas na base de dados, uma vez terminadas a stage é redireccionada para outra de sucesso.

5.3.5.2 Edição de reservas

O processo da stage EditaReserva assemelha-se à CriaReserva, contudo trabalha somente com os dados de equipamento, utilizador, data, local, hora de inicio e fim. Esta razão foi suficiente para que fosse criada uma stage à parte para as funcionalidades de alteração de reservas.

Quando a stage é chamada é verificado na inicialização o acesso à mesma e se existe algum ID de reserva válido passado por parâmetro, se ambas as condições forem válidas então os campos da reserva são preenchidos com os seus dados.

A submissão da reserva é feita à semelhança da stage CriaReserva, os parâmetros são validados, também com recurso às classes de validação, caso existe erros nos dados, os erros são guardados e mostrados no postback.

Os dados são armazenados via atributos, no inicio da stage, nos métodos da stage que são usadas pela EL na view, é feita uma verificação pelo respectivo atributo, se existir então retorna o seu valor, caso contrário retorna o valor existente na reserva original.

5.3.5.3 Listagem de reservas

A stage ListaReserva é a stage responsável pela listagem das reservas de um utilizador específico, no inicio de execução desta é verificado se um utilizador válido existe, o seu ID e função serão então armazenados tanto na view como na stage.

A sua função determina a forma como a listagem funciona. A listagem para docentes dá acesso ao menu de criação e edição mais a opção de desmarcação de reservas. A função de funcionário possui somente acesso à funcionalidade de confirmação do levantamento do equipamento.

A lista com os detalhes das reservas é criada via javascript, desta forma evita-se os constantes carregamentos e permite seleccionar a informação a mostrar. O javascript detecta na página qual o utilizador que está em sessão e chama as servlets AJAX que tratam de enviar a lista de reservas de um dado docente ou, no caso do funcionário, a lista com as reservas para o dia actual.

Sempre que a stage é invocada é executado um outro método, que altera o estado das reservas, cuja data for anterior à actual e o equipamento não tenha sido dado como levantado, para cancelado.

5.3.5.4 Homestage

Esta é a stage inicial da aplicação, possui somente a janela de login que, quando autenticado, redirecciona para a stage ListaReserva.

Nesta stage é criado um objecto DIFSession que incorpora um outro designado de DIFUserInSession, este objecto contem a propriedade profileID que assume 2 valores, "docente" ou "funcionário".

5.3.6 Views

As views são, na sua essência, JSPs que servem de *frontend* gráfico para as *Servlets* com o mesmo nome.

Estas recebem a sua informação da stage via EL, permitindo à stage ter o controlo sob o conteúdo que é mostrado ao utilizador, com a excepção de alguns dados que são carregados via AJAX, a informação desta última é utilizada em casos onde as componentes são geradas de forma dinâmica ou então usadas quando o volume de informação e validação são mínimos, não perdendo tempo com constantes carregamentos da página.

Para colmatar os erros apresentados nas componentes UI da Framework, optou-se por uma solução mais leve, estável e gratuita, a alternativa foi a biblioteca jQuery, para esta solução foi tida em conta vários aspectos, como:

- Componentes UI.
- Integração com os tipos de dados usados pela DIF.
- Compatibilidade entre os vários browsers.
- Peso na aplicação.
- Suporte para AJAX.
- Peso na comunidade.

Os ficheiros JSP foram codificados recorrendo ao uso das *tags* disponibilizadas pela Framework, isto permitiu criar templates de forma mais rápida e assim evitar repetição de código, este método do uso de *tags* foi somente excluída em casos onde o seu uso era deficiente, optando pela sua alternativa HTML e jQuery.

Por último é necessário referir que as views CriaReserva, ListaReserva e EditaReserva fazem uso das funcionalidades de javascript, seja para criação de componentes dinâmicas, AJAX, validação dos campos na parte do cliente e as componentes gráficas usadas. Estes ficheiros javascript possuem os mesmos nomes que as views, com a excepção da EditaReserva que partilha o mesmo ficheiro com a view de criação.

5.4 Testes

Os testes à aplicação foram feitas em 2 partes distintas, a primeira foi testes isolados, a segunda consistiu em testes a componentes conjuntas.

Sempre que uma componente era terminada, eram criadas classes de simulação que simulavam algumas funcionalidades de outras componentes (ex: DIFSession). Estes testes permitiam verificar as falhas que as componentes possuíam e o resultado gerado com base no input inserido.

A segunda parte começava quando uma componente era terminada, esta era testada juntamente com as já existentes, era criados casos de uso com utilizadores e reservas fictícias. Foi usada a funcionalidade de logging para registar as classes e os valores que estas estavam a gerar, isto para caso de o resultado não ser o esperado, poder-se examinar onde o processo falhou.

Durante a fase de desenvolvimento todos os valores eram mostrados na consola de output de forma a facilitar o processo de validação da lógica, sendo esta a parte mais complexa do sistema.

5.5 Integração

Um dos objectivos para a aplicação era a sua integração com a plataforma Netp@, que assenta sobre a mesma tecnologia, DIF 2.

O primeiro aspecto foi o acesso aos utilizadores, devido ao facto de as tabelas de utilizadores serem diferentes entre ambas as aplicações, escolheu-se utilizar a propriedade DIFUserInSession que guarda, de forma uniforme, os dados essenciais do utilizador numa sessão da DIF. Após um login bem sucedido este objecto é criado, após esse momento esta aplicação passa a utilizar unicamente esses dados.

O segundo aspecto é o uso da tecnologia Hibernate, este garante que, na eventualidade de uma mudança de BD, os dados mantenham-se consistentes e sem alteração no código.

Outro aspecto que garante uma melhor integração entre os sistemas é a adição de todas as componentes externas à aplicação, ser feita usando as funções próprias da Framework, permitindo esta gerar o seu próprio código, garantindo compatibilidade.

6 Conclusão

O desenvolvimento deste projecto foi concluído com o desenvolvimento de uma aplicação funcional que cumpriu os objectivos requeridos pelos intervenientes. O resultado desta foi uma aplicação Web que será integrada na plataforma Netp@.

Este projecto, para além do desenvolvimento, serviu para fazer uma análise à Framework. A sua resposta face ao problema proposto.

O desenvolvimento da aplicação foi, em muito, simplificado pelo uso da DIF, pequenos passos que teriam, repetidamente, de ser feitos manualmente, passaram a ser automatizados. Esta Framework permite uma grande integração com outras tecnologias que permitiram melhorar em muito a produtividade do projecto, como Hibernate, autenticação, validação de atributos e componentes Web UI 2.0.

Contudo a Framework não fica isenta de desvantagens que, de certa forma, prejudicam um projecto. O primeiro factor visível deve-se ao facto da documentação e material de apoio à ferramenta, implicando uma maior curva de aprendizagem que, em projectos de poucos meses, podem baixar a qualidade de um sistema final. Aliado a estes factos encontra-se a baixa versatilidade da Framework para suportar adições que não sejam modularmente integradas no seu núcleo, tarefa que exige tempo e conhecimento.

Concluo este relatório com o grande objectivo deste trabalho. A aprendizagem que foi feita com este projecto. Neste projecto adquiri conhecimento em várias tecnologias que são utilizadas no mercado de trabalho, não só do ponto de vista técnico, mas também a analisá-las consoante à produtividade, tempo de aprendizagem, possibilidades de reutilização de código e metodologias envolvidas neste processo.

A escolha de uma Framework é algo com bastante peso num projecto, existem factores a serem considerados, se esta consegue lidar com a carga da tarefa pedida, curva de aprendizagem, robustez e comunidade da mesma. Por último a concepção de uma aplicação Web requer um planeamento elaborado, caso esta falhe, o código resultante poderá estar bastante mal optimizado e a separação entre as camadas, algo que hoje em dia é imprescindível em aplicações empresarias (ex: J2EE) poderá ficar pouco clara, terminando com uma aplicação de baixa qualidade.

7 Bibliografia

Hunter, Jason; Crawford, William. Java Servlet Programming 2nd Edition. O'Reilly, 2001. ISBN: 0-596-00040-5

Olson, Steve Douglas. Ajax On Java. O'Reilly, 2007. ISBN: 0-596-10187-2

Design Patterns: Model View Controller – Model View Controller
http://java.sun.com/blueprints/patterns/MVC.html

Digitalis Development – Digitalis Development Website
http://development.digitalis.pt

GUJ – Maior fórum brasileiro sobre Java
http://www.guj.com.br

Hibernate and JBoss Community – Relational Persistence for Java & .NET
http://www.hibernate.org

J2EE at a Glance – Java Enterprise Edition
http://java.sun.com/javaee/

8 Anexos

Anexo I – Levantamento de requisites.

Anexo II – Análise e modelação.