



**Pedro Miguel
Oliveira Estima**

**Plataforma de apoio à aprendizagem de linguagens
de programação**

**Support Platform for learning programming
languages**



**Pedro Miguel
Oliveira Estima**

**Plataforma de apoio à aprendizagem de linguagens
de programação**

**Support Platform for learning programming
languages**

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Sistema de Informação, realizada sob a orientação científica do Doutor Diogo Nuno Pereira Gomes, Professor Auxiliar Convidado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro, e do Doutor João Paulo Barraca, Assistente Convidado do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

presidente / president

Prof. Doutor José Manuel Matos Moreira
Professor Auxiliar da Universidade de Aveiro

vogais / examiners committee

Prof. Doutor José Paulo Vilhena Geraldes Leal
Professor Auxiliar na Universidade do Porto

Prof. Doutor Diogo Nuno Pereira Gomes
Professor Auxiliar Convidado da Universidade de Aveiro

Prof. Doutor João Paulo Silva Barraca
Assistente Convidado da Universidade de Aveiro

agradecimentos / acknowledgements

Quero começar por agradecer aos meus pais pela possibilidade que, com muito esforço, me deram para que me fosse possível atingir esta meta e também pela educação dada para que me tornasse a pessoa que sou hoje. Obrigado pais.

Também quero agradecer aos meus orientadores, os professores Diogo Gomes e João Paulo Barraca por me terem proposto este trabalho, com o qual aprendi muito, por todas as sugestões, ajuda fornecida e o tempo que dispenderam comigo, obrigado a ambos. Quero ainda agradecer ao professor Miguel Oliveira e Silva pela sua ajuda e recomendações neste trabalho.

Por fim, quero agradecer a todos os meus amigos, aos mais presentes e aos menos presentes e que de alguma forma contribuíram para o sucesso deste trabalho, mas especialmente ao André Prata e ao Carlos Gonçalves pela ajuda fornecida.

Palavras Chave

E-learning, ensino de programação, avaliação de programas, SOA, tecnologias web, Serviços Web.

Resumo

Muito recentemente tem-se assistido a uma proliferação de ferramentas de apoio ao ensino à distância recorrendo a plataformas on-line. Estas ferramentas são demasiado focadas em determinados componentes mais relacionados com a gestão documental, a comunicação entre alunos e docentes, a administração de questionários e testes com base em perguntas de escolha múltipla ou de resposta numérica; pelo que as disciplinas de carisma mais técnico ou prático, como o caso do ensino de programação de computadores, não conseguem fazer uso de tais plataformas.

A promoção de concursos de programação por todo o mundo criou a necessidade do desenvolvimento de plataformas capazes de fazer a avaliação de código pela análise da informação à saída da execução do programa de acordo com a entrada de dados.

Neste trabalho é apresentada uma ferramenta on-line capaz de juntar o melhor dos dois géneros de ferramentas já referidos, obtendo assim uma ferramenta de análise e avaliação de código orientada ao ensino on-line.

Keywords

E-learning, learn programming, program evaluation, SOA, web technologies, web services.

Abstract

Lately, there has been an increase on the number of tools to support distance learning using online platforms. These tools are too focused on certain components mostly related to document management, communication between students and teachers, administration of questionnaires and multiple choice or numerical answer tests. Therefore more technical and practical subjects, like computer programming, can't make use of such platforms.

The promotion of programming contests throughout the world has created a need for the development of platforms capable of evaluating the code through the analysis of program output according to the input data.

In this work it is presented a system capable to combine the two kinds of tools mentioned above, resulting on a tool capable to assess and to analyse the code, aimed at e-learning.

Conteúdo

Conteúdo	i
Lista de Figuras	iii
Lista de Tabelas	v
1 Introdução	1
1.1 Enquadramento	1
1.2 Objetivos	2
1.3 Organização do Documento	2
2 Estado da Arte	5
2.1 Contextualização	5
2.2 Ferramentas Existentes	6
2.2.1 Moodle	6
2.2.2 Blackboard	7
2.2.3 PC ²	7
2.2.4 BOCA	8
2.2.5 SPOJ	9
2.2.6 KATTIS	9
2.2.7 Mooshak	9
2.2.8 CheckIO	10
2.2.9 Codeforces.com	11
2.2.10 TopCoder	11
2.3 Resumo	12
3 Solução Proposta	15
3.1 Requisitos Funcionais	15
3.2 Propostas Arquiteturais	16
3.3 Arquitetura Adotada	16
3.4 Tecnologias e Conceitos Arquiteturais	17
3.4.1 SOA	18

3.4.2	REST	18
3.4.3	Java & Java EE	18
3.4.4	JBoss AS & RESTeasy	19
3.4.5	Maven	19
3.4.6	<i>Single Page Application</i>	20
3.4.7	Backbone.js & jQuery	20
3.4.8	Twitter Bootstrap	21
3.4.9	MySQL, JPA & Hibernate	21
3.4.10	SAML	22
3.5	SAAL	23
3.5.1	Taxonomia	23
3.5.2	Casos de Utilização	24
3.5.3	Professor	25
3.5.4	Autenticação e Autorização	26
3.5.5	Base de Dados	28
3.5.6	Serviços	33
3.5.7	Aplicação WEB	36
3.6	PEPR	44
3.6.1	Contextualização	45
3.6.2	Criação do Ambiente Protegido	45
3.6.3	Arquitetura	48
3.6.4	Comunicação	49
3.6.5	Características Funcionais	51
4	Avaliação e Resultados	55
4.1	SAAL	55
4.2	PEPR	60
5	Conclusões	63
5.1	Contribuições	64
5.2	Trabalho Futuro	64
	Bibliografia	67
	Anexos	71
	Lista de Serviços	71

Lista de Figuras

3.1	Arquitetura do projeto	17
3.2	Diagrama de Casos de Utilização	24
3.3	Sequência de autenticação com SAML	27
3.4	Diagrama de tabelas da base de dados	29
3.5	Modelo relacional de gestão de permissões	30
3.6	Modelo relacional de gestão curricular	31
3.7	Modelo relacional de gestão de exercícios	32
3.8	Página de entrada	37
3.9	Vista de Administrador	38
3.10	Listagem de exercícios	39
3.11	Edição de exercício	40
3.12	Detalhes da execução de uma submissão	41
3.13	Execução na PEPR	42
3.14	Página de exercício do aluno	43
3.15	Versão em dispositivo móvel	44
3.16	Sistema de ficheiros	47
3.17	Arquitetura interna da PEPR	48
4.1	Design da página web	56
4.2	Linguagem utilizada	56
4.3	Navegabilidade da página web	57
4.4	Consistência da página web	57
4.5	Feedback após uma ação	58
4.6	Tempos de respostas	58
4.7	Prevenção de erros do utilizador	59
4.8	Feedback da submissão de trabalhos	59
4.9	Resultados do tempo de execução e memória ocupada por número de execuções	60

Lista de Tabelas

2.1	Resumo comparativo dos diferentes sistemas	12
1	Lista de permissões de serviços por grupo	71

Lista de Acrónimos

ACM	Association for Computing Machinery	P2P	Peer to Peer
AJAX	Asynchronous JavaScript and XML	PACO	Portal Académico
API	Application programming interface	PEPR	Plataforma de Execução Protegida
AST	Abstract Syntax Tree	POJO	Plain Old Java Object
CSS	Cascading Style Sheets	ORM	Object-Relational Mapping
CSV	Comma separated values	REST	Representational State Transfer
DETI	Departamento de Eletrónica, Telecomunicações e Informática	SAAL	Sistema de Apoio à Aprendizagem de Linguagens
HTML	HyperText Markup Language	SAML	Security Assertion Markup Language
HTTP	HyperText Transfer Protocol	SGBD	Sistema de Gestão de Base de Dados
ICPC	International Collegiate Programming Contest	SO	Sistema Operativo
IDE	Integrated Development Environment	SOA	Service Oriented Architecture
IdP	Identity Provider	SQL	Structured Query Language
JPA	Java Persistence API	SSL	Secure Sockets Layer
JPQL	Java Persistence Query Language	SSO	Single Sign-On
JSON	JavaScript Object Notation	UA	Universidade de Aveiro
JVM	Java Virtual Machine	UML	Unified Modeling Language
MVC	Model View Controller	VLE	Virtual Learning Environment
		WYSIWYG	What You See Is What You Get
		XML	Extensible Markup Language

Capítulo 1

Introdução

Nesta dissertação pretende-se desenvolver uma plataforma de *e-learning*, apoio ao ensino à distância, capaz de auxiliar o serviço docente do Departamento de Eletrónica, Telecomunicações e Informática (DETI) da instituição de ensino Universidade de Aveiro (UA) no desempenho das tarefas de avaliação de alunos das diversas disciplinas de ensino de programação de computadores.

1.1 Enquadramento

Nos últimos anos tem-se assistido a uma enorme proliferação no que diz respeito a ferramentas de *e-learning*. Estas caracterizam-se por um modelo de ensino à distância suportado eletronicamente num ambiente on-line com recurso a um *web browser*. São variadíssimas as ferramentas existentes (Moodle2.2.1, Blackboard2.2.2, etc.) e múltiplas as formas de apoio ao ensino (resolução de exercícios on-line, troca de mensagens com professores, etc.). Grande parte destas ferramentas suporta a adição de novas funcionalidades através de um sistema de plugins, cujas funções pedagógicas nem sempre se apresentam totalmente clarificadas.

O uso destes sistemas tornou-se generalizado por quase todas as instituições de ensino e com enorme aceitação por parte de professores e alunos [1]. Estas fornecem recursos de gestão documental, comunicação entre alunos e docentes, criação de questionários, resolução e classificação de exercícios, etc. No ensino da programação de computadores, além da utilização dos sistemas já referidos surgiu também a necessidade de ferramentas capazes de dar suporte à programação propriamente dita. Sistemas funcionais que permitam a execução de código fonte de várias linguagens de programação e análise do resultado. A promoção de concursos de programação a nível mundial criou também esta necessidade e consequentemente o desenvolvimento de ferramentas capazes de lidar com esse problema, execução e análise de código e comparação do resultado obtido com o resultado esperado.

A utilização de ferramentas capazes de auxiliar no ensino da programação de computadores é de grande interesse para os professores uma vez que auxilia na disseminação das boas práticas e do gosto pela programação aos alunos, permitindo que estes obtenham em tempo real a validação dos exercícios previamente definidos pelo professor. O ato de os alunos exercitarem as suas capacidades de programação com exercícios que vão sofrendo um incremento de dificuldade permite que a sua progressão e prática os torne melhores programadores.

Relativamente aos sistemas de gestão de concursos de programação, existem vários capazes de efetuar aceitação de código e execução do mesmo independentemente da linguagem de programação

usada, todos estes sistemas seguem uma vertente orientada à necessidade pela qual foram desenvolvidos, e desta forma, não são capazes de dar a melhor resposta à necessidade de um sistema de apoio ao ensino da programação para as instituições de ensino que permitam efetuar a gestão de alunos e professores, bem como o ambiente envolvente.

1.2 Objetivos

Para este trabalho foi proposto o desenvolvimento de um sistema capaz de dar resposta à necessidade criada com o ensino de disciplinas de programação de computadores. Este sistema tem de ser capaz de implementar os vários mecanismos que tornam as ferramentas de e-learning de fácil adoção ao mesmo tempo que permita efetuar a análise de código fonte bem como sua execução e respetiva análise do resultado de saída. Gestão de alunos, professores, turmas e disciplinas são algumas das principais características que este sistema terá de implementar e que se relacionam com os mecanismos presentes em ferramentas de e-learning existentes. Parte da essência desta aplicação prende-se com a gestão dos exercícios dentro de ambiente escolar, estes terão de ser catalogados por módulos e ter associados vários testes que funcionarão como validadores do código fonte submetido pelos alunos, ou seja, o código terá de passar todos os testes corretamente para ser considerado válido.

De acordo com as tendências tecnológicas mais recentes na web, verifica-se um grande crescimento na adoção de aplicações desenvolvidas com base em Service Oriented Architecture (SOA). Pretende-se então uma aplicação baseada nesta arquitetura que implemente fortes componentes de segurança e restrição de acesso, características que esta arquitetura obriga. É também necessário o desenvolvimento de uma aplicação web capaz de fazer uso dos serviços desenvolvidos. Tal aplicação terá de ser pensada de acordo com o que é pretendido pelos alunos, ou seja, consulta de exercícios e entrega das resoluções, pelo que terá de ser fácil interação e visualmente agradável.

É sabido pela experiência dos professores que neste género de sistemas existirão momentos mais ou menos bem definidos com um incremento de afluência dos alunos a fim de efetuarem as submissões dos seus trabalhos. O sistema terá de ser capaz de lidar com este problema distribuindo o processamento das várias submissões efetuadas por vários sistemas. À submissão de código fonte por parte dos alunos é exigida uma resposta do sistema no mínimo espaço de tempo acerca da validade da submissão, é necessário apresentar informação detalhada do processo de execução do código do aluno facilitando a deteção e correção de falhas se existirem.

A execução do código terá de acontecer num ambiente controlado onde a execução de código malicioso não comprometa a máquina onde está a acontecer a execução bem como todas as outras presentes na rede. O tempos de execução são uma peça fundamental a ter em consideração e que têm de ser constantemente monitorizados, não podendo existir processo nenhum em execução durante um tempo pré estabelecido e configurável. É necessário também garantir o arquivamento do código de todas as submissões efetuadas.

1.3 Organização do Documento

Este trabalho encontra-se organizado em sete capítulos, sendo que o primeiro é focado na contextualização do problema e explicação da necessidade que levou ao seu desenvolvimento bem como os objetivos propostos. No segundo capítulo será dado a conhecer o estado da arte, ou seja, serão apresentados vários sistemas que têm evoluído ao longo dos anos para colmatar as necessidades de sistemas

de apoio ao ensino à distância bem como de execução e análise de resultados da execução de código. Segue-se o terceiro capítulo onde será apresentada a aplicação desenvolvida neste trabalho. Este está seccionado na arquitetura de implementação, onde será apresentado de forma muito geral o trabalho e as várias relações de dependências existentes e na modelação e detalhes de implementação da solução desenvolvida, que consiste em duas aplicações.

Segue-se o terceiro capítulo onde será apresentada a aplicação desenvolvida neste trabalho seccionado com a arquitetura de implementação, onde será apresentado de forma muito geral o trabalho e as várias relações e dependências existentes e a modelação e detalhes de implementação da solução desenvolvida que consiste em duas aplicações. Foi também conduzida uma avaliação ao trabalho desenvolvido em forma de inquérito e com alguns testes de desempenho com os resultados a serem apresentados no capítulo seis. No último capítulo serão apresentadas as conclusões e o trabalho futuro.

Capítulo 2

Estado da Arte

Neste capítulo será contextualizado o estado da arte e algumas aproximações para a resolução do problema apresentado neste trabalho. Serão também focadas algumas das plataformas mais maduras que surgiram como forma de colmatar as várias necessidades existentes no âmbito das ferramentas de apoio ao ensino à distância e de execução e análise de código fonte.

2.1 Contextualização

Nas ferramentas que suportam compilação e execução de código fonte existe a problemática da definição da análise do resultado obtido. Existem várias vertentes que podem ser consideradas, a comparação do resultado obtido com um resultado previamente definido, a análise do código fonte em si e estruturas lógicas implementadas confirmando a implementação dos conceitos exigidos, etc. A comparação do resultado obtido com algo previamente definido será a forma mais fácil de garantir que a execução retorna o resultado esperado. Esta é também a forma mais fácil de enganar o sistema uma vez que é possível definir de forma estática o que a aplicação deve devolver. A criação de vários testes pelos quais a execução terá de responder corretamente ajuda a evitar a programação estática do resultado ao impor que o código submetido se comporte de diferente forma em cada um dos vários testes. Esta prática poderá ser usada em todas as soluções que se possam criar para verificação da execução de código, contudo pode também ser usada em simultâneo com a análise semântica do código. A análise semântica do código fonte permite verificar se este cumpre determinados padrões necessários, por exemplo, verificar se foi implementado usando recursividade ou se existem funções que criem determinadas iterações cíclicas, entre outras.

Uma outra alternativa às duas acima apresentadas seriam as Abstract Syntax Tree (AST), estas são representações abstratas em árvore da estrutura sintática do código fonte. Os nós na árvore criada representam ocorrências no código que podem ser identificadas como estruturas cíclicas ou de controlo de fluxo e que desta forma criam as ramificações e consequentemente novos nós. As AST são sobretudo usadas por compiladores na análise semântica do código fonte escrito e também com objetivo de validar a estrutura lógica do mesmo.

2.2 Ferramentas Existentes

Nesta secção serão apresentadas algumas ferramentas existentes que podem ser enquadradas no contexto de apoio ao ensino à distância ou na execução de código fonte. Independentemente do contexto onde se enquadrem, todas tiveram um longo tempo de maturação até atingirem o estado de desenvolvimento atual, algumas destas continuam ainda em permanente desenvolvimento, adicionando novas funcionalidades e definindo constantemente os objetivos para os quais foram desenvolvidas. De seguida são apresentadas duas ferramentas de apoio ao ensino à distância e oito capazes de efetuar a compilação de código fonte.

2.2.1 Moodle

O Moodle [2] é uma plataforma de e-learning, Virtual Learning Environment (VLE), desenvolvida a partir de princípios pedagógicos bem definidos e a pensar na criação de comunidades de aprendizagem. Com uma grande vertente orientada ao uso escolar facilita a disseminação de conteúdos de suporte para diferentes cursos bem como a comunicação interna entre utilizadores onde alunos e professores podem aceder a um conjunto de ferramentas e recursos que são partilhados com diferentes níveis de privilégios.

A palavra “Moodle” significa “Modular Object-Oriented Dynamic Learning Environment”, que é especialmente significativo para programadores e investigadores da área da educação. Em inglês a palavra “Moodle” é também um verbo que descreve a ação que, com frequência, conduz a resultados criativos.

Este sistema começou com a tese de Doutoramento de Martin Dougiamas relativa ao uso de ferramentas livres para suporte ao ensino em comunidades baseadas na internet, onde foi desenvolvida uma primeira versão do sistema. Este foi continuamente melhorado mantendo-se sempre como ferramenta de uso livre e já conta com suporte para 82 línguas diferentes. O seu sistema de plugins permite-lhe ser tão extensível quanto necessário e o facto de não ser necessário licenças permite que toda a comunidade desenvolva e ajude no seu crescimento.

Do ponto de vista pedagógico é uma ferramenta que dá grande ênfase ao utilizador Aluno, algo que não era tão visível até então. Tal característica juntamente com a facilidade de utilização, mesmo junto de alunos do ensino básico, contribuíram para que esta ferramenta se tornasse num grande sucesso principalmente entre comunidades escolares.

Algumas das principais características implementadas nativamente no sistema são a submissão de trabalhos, possibilidade de criar discussões dentro de fóruns categorizados por disciplinas, efetuar descarregamento de ficheiros disponibilizados pelos professores, sistema de troca de mensagens e envio de e-mails, calendário onde poderão ser marcados eventos dos quais os alunos serão informados (testes, etc.), sistema de notícias por estabelecimento de ensino ou por disciplina e finalmente questionários on-line de resposta rápida e direta com avaliação imediata. Do ponto de vista mais técnico, este sistema está preparado para suportar vários tipos de interoperabilidade, por exemplo, permite a integração com vários sistemas de autenticação padronizados (LDAP, Shibboleth, etc.), importação de exercícios da plataforma Blackboard 2.2.2 e integração com outros sistemas de gestão de conteúdos (Joomla, Drupal, Postnuke, etc).

Desenvolvido colaborativamente por uma comunidade virtual que reúne programadores, administradores de sistemas, professores, e designers de todo o mundo, está em constante evolução adequando-se às necessidades que estão constantemente a surgir.

Hoje em dia é utilizado maioritariamente por escolas e universidades, não só para cursos virtuais, mas também para aulas presenciais de alunos. Além destas atividades, também se enquadram formações de grupos e desenvolvimento de projetos. Nesta ferramenta não existe suporte para efetuar a análise e execução de código fonte.

2.2.2 Blackboard

A plataforma de apoio ao ensino Blackboard [3] é um sistema web que se foca em vários aspetos da experiência educacional para ambientes escolares universitários como de ensino secundário. À semelhança do Moodle 2.2.1, esta ferramenta é também uma VLE mas constituída por um diverso número de plataformas com diferentes especializações (ensino, colaborativa, móvel, etc) que quando instaladas, funcionam de forma integrada.

Esta ferramenta permite aos professores a disseminação de recursos on-line para os alunos nos formatos de texto, imagens, som e vídeo, gráficos, etc. bem como a possibilidade de criação de questionários de preparação para testes e exames. Permite ainda que os alunos se divirtam enquanto aprendem com jogos como o *Quem quer ser milionário ou Jeopardy!*¹.

Algumas das características mais interessantes do Blackboard são a possibilidade de consulta da informação das disciplinas nas quais o aluno se encontra inscrito. Esta informação passa por uma descrição da disciplinas, objetivos que o aluno deve atingir, políticas de avaliação, material necessário e até um calendário de datas importantes. Existem também ferramentas que permitem o trabalho colaborativo entre alunos com um quadro para discussão de ideias, troca de mensagens instantâneas e ficheiros e ajuda no controlo do tempo de atividades para melhor organização do trabalho. Existe ainda a possibilidade de simulação de exercícios específicos da vida real que podem ser repetidos e executados sem perigos envolvidos; este tipo de exercícios recorre a um conjunto de ferramentas externas para criação dos ambientes de simulação.

Apesar das diferentes plataformas existentes do Blackboard, não existe ainda a possibilidade em nenhuma destas para análise e execução de código fonte.

2.2.3 PC²

O PC² [4] é provavelmente a primeira ferramenta de organização e gestão de concursos de programação. Foi desenvolvido na universidade *California State University, Sacramento* para avaliação de concursos organizados pela Association for Computing Machinery (ACM). A versão 1.0 surgiu no ano de 1989 com suporte apenas para a linguagem de programação Turbo Pascal e até ao ano de 1994 esta ferramenta funcionava apenas em sistemas locais com o Sistema Operativo (SO) instalado MS-DOS. Em 1998 foram introduzidas novidades como suporte ao Java e possibilidade de executar em sistemas UNIX. Esta ferramenta foi mantida até ao ano de 2008 com a sua versão 9, altura que foi substituído pelo KATTIS 2.2.6.

Esta ferramenta, já antiquada e um pouco antiga para os dias de hoje, constitui um dos primeiros esforços no desenvolvimentos de aplicações cujo objetivo fosse ao encontro da execução de código fonte de forma autónoma e sem interação humana. Todo o controlo da aplicação, bem como procedimentos de execução do código submetido, é realizado com recurso de *scripts* cuja informação é persistida numa base de dados em Microsoft Excel. Nas versões mais recentes, a interface de utilizador para gestão dos concursos passou a ser desenvolvida em Java e desta forma multi-plataforma. Apesar desta

¹ JEOPARDY! - America's Favorite Quiz Show, <http://www.jeopardy.com/>

característica, todos os utilizadores administrativos são obrigados a manter uma cópia da aplicação por não existir nenhum sistema que permita a distribuição. Ainda assim é uma ferramenta com muitos anos de maturação e com um grande leque de funcionalidades, tais como:

- gestão de vários tipos de utilizadores, com possibilidade de definir permissões específicas para cada utilizador individualmente
- definição muito básica de algumas linguagens de programação pré-definidas
- sistema de envio de mensagens ou notificações por exercício
- visualização de relatórios com informação categorizada por utilizadores, submissões, tempos de execução, etc.
- sistema básico de avaliação da execução de uma aplicação
- suporte a múltiplas instâncias, simulando um ambiente distribuído

2.2.4 BOCA

O BOCA [5] é um sistema on-line desenvolvido para ser usado nas maratonas de programação organizadas pela Sociedade Brasileira de Computação. Estas competições, destinadas apenas a alunos do ensino superior, começaram em 1996. Devido ao seu sucesso, os números oficiais apontavam para mais de 1300 estudantes inscritos e cerca de 60 escolas em 2003. Devido a este crescente interesse dos alunos foi necessário desenvolver um sistema capaz de responder de forma rápida às submissões dos exercícios efetuados pelos alunos, e assim surgiu o BOCA.

Uma vez que os concursos em questão já eram destinados apenas a alunos do ensino superior, e combinando esta característica com a facilidade de instalação e manutenção do sistema, rapidamente esta passou a ser usada em algumas escolas como sistema de submissão e avaliação de trabalhos escolares.

Esta ferramenta é orientada ao ambiente web e desenvolvida em PHP com persistência de dados em PostgreSQL. As suas principais características são a portabilidade, controlo de concorrência em ambientes distribuídos e uma interface gráfica web. A utilização de ligações Secure Sockets Layer (SSL) permite um ambiente totalmente seguro e confiável para os utilizadores do sistema que podem aceder aos diferentes concursos configurados na aplicação.

Um dos principais problemas deste sistema prende-se com o facto de não ser automático e autónomo do ponto de vista da análise e avaliação dos problemas submetidos. Todas as submissões têm de passar por um agente, denominado pela aplicação como *juiz* e que tem a responsabilidade de avaliar o código e transmitir à equipa participante uma das seguintes respostas: certo, errado, saída mal formatada, erro de compilação ou erro em tempo de execução. Nenhuma informação mais é passada aos participantes que são assim obrigados a encontrar um erro que poderá nem acontecer nos seus computadores e que desta forma torna a correção mais demorada e difícil.

Os administradores têm a tarefa inicial mais complicada uma vez que têm de inserir manualmente todos os elementos para as provas a realizar; desses elementos entendam-se as linguagens de programação permitidas, as várias questões que os participantes têm de responder, os vários juizes envolvidos e ainda os próprios participantes. Após delinear todo o concurso, e após início deste, o administrador tem ainda a possibilidade de acompanhar todo o concurso e mesmo de o influenciar, desde efetuar correções a avaliações já efetuadas por juizes até eliminar submissões feitas pelos participantes.

2.2.5 SPOJ

O *SPhere Online Judge*, mais conhecido por SPOJ [6], é uma sistema web que permite a avaliação online e automática de programas submetidos por vários utilizadores de todo o mundo. Atualmente este sistema tem suporte para 45 linguagens de programação e conta já com mais de 13000 exercícios para resolver com suporte em várias línguas, uma das quais o português (do Brasil).

O que distingue esta plataforma de outras, é a possibilidade de serem os próprios utilizadores a criarem os seus próprios exercícios e submete-los para serem resolvidos por outros, tornando-o desta forma mais orientado para a prática e aprendizagem de programação, independentemente da linguagem usada.

Este sistema permite também a criação de concursos próprios para grupos de utilizadores restritos, tornando-o assim ideal para ser usado por aplicações web de terceiros para realização de concursos de programação. Um caso de sucesso que faz uso desta plataforma é o CodeChef² que já conta com mais de 210 concursos realizados em todo o mundo.

Outra característica inovadora é o facto de existirem categorias com diferentes formas de avaliação:

- Clássico** respostas validadas apenas com Correto ou Incorreto
- Desafio** problemas permitem que sejam submetidos vários tipos de soluções, umas mais corretas que outras
- Tutorial** igual ao clássico mas com problemas de mais fácil resolução (para uso educacional)
- Parcial** igual ao desafio, mas também com um grau de facilitismo mais elevado

2.2.6 KATTIS

KATTIS [7] é um sistema de gestão de concursos de programação on-line desenvolvido pela Royal Institute of Technology, na Suécia. Este foi usado nos mais conhecidos e prestigiados concursos organizados pela ACM nas finais mundiais dos International Collegiate Programming Contest (ICPC). O KATTIS é um sistema que se baseia numa arquitetura cliente-servidor e além da interface web, permite também a interação através da troca de mensagens por e-mail.

Uma grande vantagem deste sistema é a sua disponibilidade on-line para utilização por parte dos membros KTH Royal Institute of Technology, ou seja, não existe necessidade de efetuar transferências e posteriores instalações do sistema, basta apenas usar o sistema que já se encontra disponível on-line, e para isso só é necessário efetuar um registo na aplicação.

Na criação de concursos existe a possibilidade de os tornar públicos ou privados, pelo que os públicos podem ser consultados por qualquer pessoa sem nenhum registo prévio. Na própria página web desta aplicação é possível seguir alguns dos concursos públicos e ver mesmo informação acerca das várias submissões efetuadas.

2.2.7 Mooshak

Mooshak [8] é um sistema web para gestão e organização de concursos de programação baseado na arquitetura cliente-servidor desenvolvido pelo Prof. José Paulo Leal, do Departamento de Ciência de Computadores da Faculdade de Ciências da Universidade do Porto. Este sistema é usado em vários concursos [9] tais como nas Olimpíadas Nacionais de Informática, a Maratona Inter-Universitária de

²CodeChef, <http://www.codechef.com>

Programação e o concurso regional ACM do Sudoeste da Europa. Mas a sua utilização não se fica apenas por aqui, este sistema é também amplamente usado no ensino universitário em Portugal já que foi adotado por várias universidades para apoio às aulas de ensino de programação de computadores.

O sistema é composto por quatro tipos de utilizadores distintos, são eles o Administradores, o Júri, os Concorrentes e a Audiência. O primeiro tem a responsabilidade de criar e gerir novos concursos, efetuar também a gestão dos problemas bem como das linguagens de programação que se poderão usar, existe a necessidade de ser o administrador a gerir também as equipas e definir as datas e tempos de início e fim do concurso. O segundo, os juízes, têm ao seu encargo a validação das submissões efetuadas pelos concorrentes, ou seja, cada submissão é respondida automaticamente pelo sistema, mas existe ainda a possibilidade dos juízes alterarem a resposta dada pelo sistema ou até mesmo pedir nova avaliação. São os juízes que têm ainda de responder às dúvidas colocadas pelos concorrentes. Estes, os concorrentes, têm a possibilidade de envio de uma ou mais propostas de solução para um problema, visualização da resposta automatizada e resposta definida pelos revisores, ver o enunciado dos problemas, fazer perguntas ao júri, ver todas as submissões efetuadas (e respectivo código), ver a classificação atual e até visualizar uma pequena ajuda sobre o sistema. Por fim, a audiência, apenas pode visualizar as classificações e outras listagens, como equipas, problemas etc.

O sistema de ranking utilizado pelo Mooshak é o mesmo usado pelo ACM-ICPC e consiste na seguinte formula: apenas contam o número de problemas resolvidos. No caso de existirem empates, então verifica-se o tempo, os concorrentes que demoraram menos tempo a resolver os problemas ficam à frente. Existe ainda outra condicionante, por cada submissão errada efetuada pelos concorrentes, existe um acréscimo de mais 20 minutos ao tempo total.

O Mooshak é também um sistema escalável que permite que vários servidores ligados entre si, todos a executar o Mooshak, disponibilizem um serviço que pode ser usado tanto por concursos de pequena como de grande dimensão e ao mesmo tempo garantem fiabilidade ao implementar sistemas de replicação e recuperação de informação.

2.2.8 CheckIO

CheckIO [10] é uma aplicação on-line que surgiu no ano de 2011 e é *open-source*. Esta é apresentada em forma de um jogo no browser de cariz interativo e educacional e com o único objetivo de ensinar programação na linguagem *Python*. Este jogo está desenvolvido para que tanto programadores iniciados como avançados se possam divertir enquanto aprendem uma vez que todos os exercícios são apresentados mediante uma história com vários personagens. Também existe um mecanismo de recompensa que por cada exercício resolvido corretamente os utilizadores ganham pontos que podem ser usados para inúmeras coisas, tais como construção de um avião para viajar para uma ilha à procura de novos desafios.

Neste jogo o único requisito do utilizador é um *browser* e ligação à internet. A resolução de exercícios acontece no browser é feita num editor de texto com *syntax highlight*, uma consola para apresentação das saídas dos programas e/ou erros de execução e a possibilidade de escolher qual das versões da linguagem *Python* se pretende usar. Os pedidos de execução enviam o código para o servidor que o executa e devolve a resposta.

2.2.9 Codeforces.com

Codeforces [11] é um sistema desenvolvido pela *Saratov State University*, na Rússia, com objetivo de criar uma plataforma para organizar e executar concursos de programação. Este sistema conta também com uma componente social bastante forte e ao mesmo tempo porta-se como um portal de notícias.

O web site é responsável pela organização de concursos de programação mas ao mesmo tempo permite que qualquer utilizador possa também gerir e organizar os seus próprios concursos, com a possibilidade de os tornar públicos ou privados. A forma de avaliação é uma mais valia já que é inspirada no método de ELO [12], método também usado nos concursos de xadrez que permite classificar jogadores de acordo com o ranking atual.

Num nível mais técnico, este sistema de gestão de concursos tem suporte para as linguagens de programação mais conhecidas, tais como: C/C++, PHP, Python, C#, Java, MS C++, Perl, Pascal, Haskell, etc. Durante os concursos os vários utilizadores são divididos em grupos de elementos de acordo com a sua classificação, desta forma as competições tornam-se mais justas uma vez que diminuí a disparidade de conhecimento entre estes.

Este sistema permite a criação e manutenção de concursos de programação bem como toda a gestão envolvida nestes. Relativamente às linguagens de programação suportadas, é ainda possível efetuar alterações no processo de compilação ao passar argumentos de compilação.

2.2.10 TopCoder

TopCoder [13] é a comunidade de desenvolvimento de software com maior número de membros no mundo, cerca de 480 mil pessoas divididas entre programadores, especialistas em algoritmos e designers digitais que representam mais de 200 países e criam uma rede global de profissionais de software [14].

Esta plataforma on-line consegue juntar muitos participantes pois envolve uma lógica de negócio muito interessante e inovadora ao servir de centro de recrutamento de programadores para empresas ao mesmo tempo que permite que estas submetam projetos ao TopCoder. Para a comunidade de desenvolvimento on-line é apenas mais um exercício numa competição onde as equipas participantes tentam ficar em primeiro lugar no sistema de avaliação. O trabalho realizado por esta comunidade passa pela concepção, análise e implementação de cada parte do projeto inicial, por fim a equipa vencedora de cada uma das partes poderá receber um prémio definido pela empresa.

São também organizados concursos em que todos os participantes competem individualmente com as provas a acontecerem durante um período de tempo definido. No final poderá ser consultada a lista de posições dos melhores programadores e para fomentar ainda mais a competição são distribuídos prémios monetários patrocinados por empresas como Google³ ou Yahoo⁴. De forma a garantir a igualdade entre utilizadores, estes são dispersos por grupos de até 20 elementos com classificações semelhantes.

Para contribuir ainda mais para o sucesso desta plataforma, esta dispõe ainda da possibilidade dos utilizadores poderem organizar e gerir os seus próprios concursos.

³Google, <http://www.google.com>

⁴Yahoo, <http://www.yahoo.com>

2.3 Resumo

As várias ferramentas acima apresentadas representam dois conjuntos de sistemas, um de apoio ao ensino à distância e outro de execução e análise de código fonte. Os sistemas de apoio ao ensino são muito semelhantes do ponto de vista das funcionalidades disponíveis e diferem principalmente na usabilidade destas, em relação aos sistemas de execução de código, são um pouco mais distintos, tanto nas funcionalidades como na usabilidade, principalmente devido ao objetivo pelo qual foram desenvolvidos.

Em relação às ferramentas de execução de código, todas estas cumprem os objetivos principais para os quais foram desenvolvidas, ainda assim a forma como conseguem atingir esses objetivos varia consoante as necessidades e objetivos apresentadas de cada uma. Deste conjunto mais restrito de ferramentas só a CheckIO 2.2.8 foi direcionada especificamente para o ensino da programação em Python e não permite a submissão de código fonte nem definição de exercícios, pelo que existe um enorme entrave à sua aplicação em ambientes escolares. Do restante conjunto existente de ferramentas, apenas duas serão capazes de fornecer alguma ajuda em ambientes escolares no ensino da programação de computadores, são elas o BOCA 2.2.4 e o Mooshak 2.2.7. As suas características e facilidade de instalação, manutenção e suporte, podem permitir a sua adoção tanto por concursos de programação como por escolas secundárias e faculdades.

Apesar de melhor facilidade na adaptabilidade de algumas ferramentas de execução de código a um ambiente escolar, estas apresentam limitações e problemas que este trabalho pretende resolver ao combinar as suas funcionalidades com as funcionalidades de gestão curricular apresentadas pelas ferramentas de apoio ao ensino. Um resumo das funcionalidades principais das várias ferramentas apresentadas e que vão de encontro aos objetivos deste trabalho, podem ser consultadas na tabela 2.1.

	Utilizadores / Turmas	Executa Código	Exercícios Customizados	Sistema de Comunicação	Permite Download	Multi. Ling. Prog.
Moodle	Sim/Sim	Não	Não	Sim	Sim	-
Blackboard	Sim/Sim	Não	Não	Sim	Sim	-
PC2	Sim/Não	Sim	Sim	Sim	Sim	Sim
BOCA	Sim/Não	Sim	Sim	Sim	Sim	Sim
SPOJ	Sim/Não	Sim	Sim	Sim	Não	Sim
KATTIS	Sim/Não	Sim	Sim	Sim	Não	Sim
Mooshak	Sim/Não	Sim	Sim	Sim	Sim	Sim
CheckIO	Sim/Não	Sim	Não	Não	Não	Não
Codesforces	Sim/Não	Sim	Sim	Sim	Não	Sim
TopCoder	Sim/Não	Sim	Sim	Sim	Não	Sim

Tabela 2.1: Resumo comparativo dos diferentes sistemas

Como seria de esperar, todas as ferramentas permitem efetuar a gestão de utilizadores mas só as ferramentas de e-learning têm a possibilidade de gestão de turmas e professores. Esta característica é essencial num sistema que efetue a execução de código a ser adotado em ambientes de ensino de programação de computadores. Como se pode observar pela consulta da tabela, os dois sistemas de apoio ao ensino são muito semelhantes e apresentam características idênticas, da mesma forma que os sistemas de execução de código também se comportam de forma idêntica entre si, com a exceção do CheckIO que não permite a customização de exercícios, não tem um sistema de comunicação de

utilizadores e não permite a execução de diferentes linguagens de programação. Ainda relativamente aos sistemas de execução de código, a única diferença que estes apresentam entre si relaciona-se com a possibilidade de poder ser feito *download* do sistema para configuração própria ou se apenas pode ser usado no domínio on-line do próprio sistema.

Capítulo 3

Solução Proposta

Neste capítulo serão dados a conhecer alguns modelos arquiteturais nos quais a aplicação a desenvolver se poderá apoiar e proposta uma arquitetura da aplicação a desenvolver que será validada pela aplicação final. A proposta arquitetural é de extrema importância no desenvolvimento de software por poder ser responsável pelo sucesso da solução final. A fusão da arquitetura com as tecnologias usadas no desenvolvimento poderão ditar o rumo do desenvolvimento e o seu sucesso. Serão também clarificados os vários detalhes da aplicação desenvolvida que consiste em duas soluções que trabalham em conjunto.

3.1 Requisitos Funcionais

A análise de requisitos consiste na especificação das funcionalidades que o sistema terá de suportar, estas funcionalidades poderão ser traduzidas em requisitos funcionais e em casos de utilização. Em relação aos requisitos do sistema a desenvolver, é necessário que este suporte autenticação de utilizadores e diferentes funcionalidades de acordo com as permissões definidas a cada um destes (administradores, professores e alunos). Será também necessário que a aplicação a desenvolver suporte um sistema de gestão de alunos e respetivas turmas por disciplinas bem como implemente a capacidade de efetuar gestão de exercícios e que estes sejam categorizados em grupos. Ainda relativamente à gestão de alunos, terá de dar suporte à importação automática de alunos de sistemas externos. No contexto deste trabalho, esse sistema externo será o Portal Académico (PACO), responsável pela gestão de turmas na UA.

Uma vez que a aplicação poderá dar suporte a múltiplas disciplinas e irá tratar informação relativamente sensível, como exercícios de testes que poderão ser realizados no sistema, é fundamental implementar um sistema de controlo de acesso que tenha em consideração a informação que cada utilizador poderá aceder.

Por último, e um dos requisitos que levou ao desenvolvimento deste trabalho, é a possibilidade que a aplicação terá de suportar para efetuar a execução de código fonte e análise de resultados obtidos das execuções. Tal como acontece com algumas das ferramentas de execução de código fonte apresentadas na secção Ferramentas Existentes 2.2, é estritamente necessário não limitar este sistema à execução de apenas uma linguagem de programação, tornando o sistema modular e capaz de executar qualquer código fonte, independentemente da sua linguagem.

Estes são os requisitos funcionais de alto nível e serão detalhados e desenvolvidos em casos de

utilização durante a apresentação do sistema implementado.

3.2 Propostas Arquiteturais

Existem várias propostas arquiteturais para projeção de software, entre as mais conhecidas encontram-se as aplicações monolíticas, cliente-servidor, Peer to Peer (P2P), SOA.

As aplicações monolíticas são talvez das arquiteturas mais usuais e antigas de desenvolvimento de software. Neste género de aplicações a interface do utilizador é combinada no mesmo programa junto com a camada de acesso aos dados e completamente dependente da plataforma para a qual foi compilada. Assume-se que estas aplicações são independentes de outras e que são também responsáveis por todas e quaisquer funcionalidades necessárias para a execução de determinada tarefa. Uma aplicação monolítica, em geral, não faz uso da modularidade para reuso de funcionalidades de outros sistemas.

O crescimento que se tem assistido nos últimos anos na web levou ao desenvolvimento das aplicações baseadas na arquitetura cliente-servidor. Estas caracterizam-se por serem modelos de aplicações distribuídas onde os recursos são fornecidos por servidores e consumidos por clientes. O exemplo mais típico de uma aplicação deste género é a consulta de uma página web, o browser do utilizador é o cliente de um serviço disponibilizado por uma outra entidade. Uma das principais vantagens relaciona-se com a facilidade existente na administração por existirem um número reduzido de servidores e que desta forma também permitem que os recursos sejam centralizados (bases de dados, ficheiros, etc) evitando problemas de redundância de informação. Existe também maior segurança por existirem menos pontos de acesso aos serviços, mas por outro lado, se a segurança deste for comprometida, também os clientes poderão ser comprometidos.

Uma aplicação P2P é frequentemente usada em redes de computadores onde cada um destes atua como sendo um cliente e um servidor em simultâneo. Os vários computadores da rede são denominados de “nós” e dispõem de funcionalidades de partilha e consumo de serviços comuns sem a necessidade de servidor central capaz de efetuar tal gestão. Estas aplicações têm a vantagem de, por estarem todas ligadas entre si, terem uma elevada taxa de disponibilidade, uma vez que se um nó ficar incomunicável, existirão outros capazes de continuar a comunicação e fornecimento de serviços.

As aplicações que seguem uma arquitetura orientada aos serviços, SOA, presam-se pelo desenvolvimento de funcionalidades que são expostas em forma de serviços. Estes são muitas vezes encontrados associados a um barramento de serviços que permitem a conectividade de serviços de múltiplas entidades bem como disponibilidade baseada em contratos. Também esta arquitetura é baseada num modelo distribuído. Uma das suas principais vantagens relaciona-se com a integração de serviços de diferentes departamentos cujas funcionalidades são necessárias entre ambos.

3.3 Arquitetura Adotada

Neste trabalho foi pensada uma solução que integra as arquiteturas cliente-servidor e SOA. A primeira foi usada no desenvolvimento de uma aplicação web e a segunda para o desenvolvimento de um conjunto de serviços que dão suporte às várias páginas web. Desta forma o trabalho desenvolvido é fundamentalmente constituído por dois módulos que apresentam funcionalidades completamente distintas, fez-se então uma separação lógica de ambos que originou assim duas aplicações.

Na Figura 3.1 é apresentada a esquematização dos módulos que constituem este trabalho bem como as ligações entre estes.

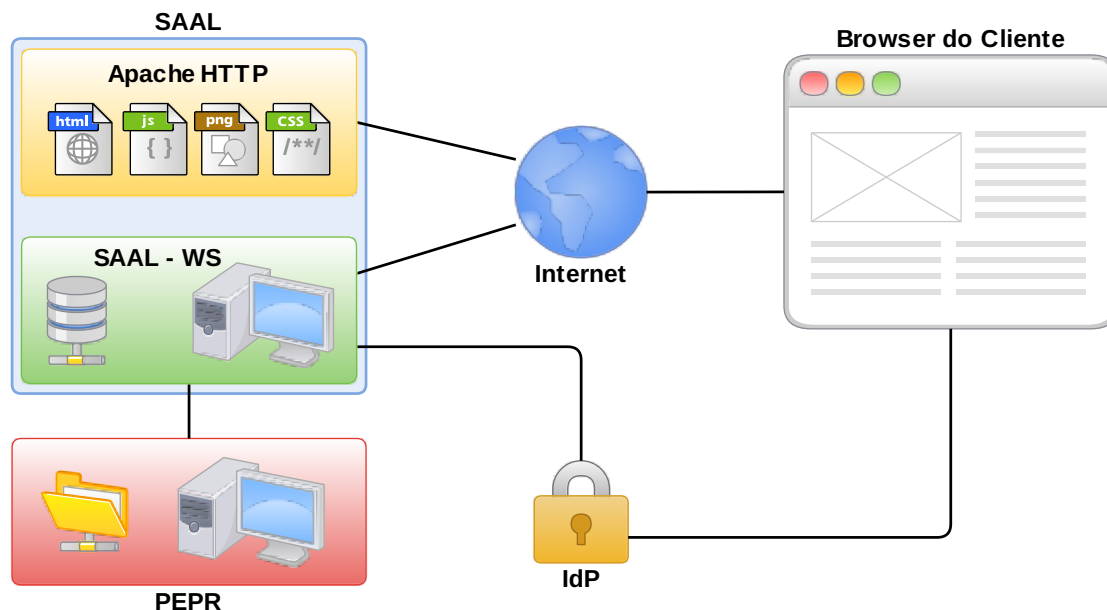


Figura 3.1: Arquitetura do projeto

O Sistema de Apoio à Aprendizagem de Linguagens (SAAL) é constituído por um conjunto de serviços web que foram desenvolvidos para dar suporte a um web site e como é visível na Figura 3.1 são considerados como um sistema único. O web site desenvolvido atua como um front-end para professores e alunos de todos os serviços implementados. A Plataforma de Execução Protegida (PEPR) é a segunda aplicação desenvolvida e tem como principais funcionalidades a criação de ambientes virtuais protegidos, execução de código fonte e análise do resultado da execução.

O trabalho aqui apresentado é constituído pelos dois blocos do lado esquerdo da Figura 3.1, nela realça-se a divisão do projeto SAAL nos dois sub projetos, a aplicação web e os serviços. É também visível o componente **IdP** que funcionará como um provedor de identidades genérico que tem a finalidade de autenticar os vários utilizadores do sistema.

A comunicação entre o vários componentes é iniciada pelo browser cliente que comunica diretamente com o SAAL, em particular com cada um dos dois sub projetos desenvolvidos. O pedido para execução de código terá também de ser feito ao SAAL que comunicará esse pedido à PEPR. O browser do cliente comunica ainda com o Identity Provider (IdP), que por sua vez comunicará com o SAAL com o objetivo de efetuar a autenticação do utilizador no sistema.

3.4 Tecnologias e Conceitos Arquiteturais

O desenvolvimento de uma nova ferramenta envolve a utilização de várias tecnologias já existentes e também a adoção de alguns conceitos arquiteturais para apoio ao desenvolvimento ou mesmo como forma de implementação de determinadas escolhas tecnológicas. Neste trabalho não foram feitas nenhuma restrições na escolha de que tecnologias utilizar, ainda assim foram dadas várias sugestões de algumas aproximações à resolução. Seguidamente serão apresentadas algumas das tecnologias usadas bem como conceitos arquiteturais que servirão para dar suporte ao desenvolvimento da aplicação e que irão de encontro à arquitetura apresentada 3.3.

3.4.1 SOA

SOA é uma arquitetura que essencialmente é uma evolução natural das tecnologias de sistemas distribuídos, como por exemplo, o conceito de cliente-servidor ou *peer-to-peer* onde existiam dependências das plataformas, protocolos e formatos proprietários dos dados, falta de flexibilidade etc. SOA, ou em português, Arquitetura Orientada a Serviços, apresenta uma visão de um recurso bem definido e que pode ser usado por terceiros independentemente das tecnologias usadas, tanto para desenvolvimento dos próprios serviços, como da aplicação que irá consumir esses serviços.

Esta arquitetura é principalmente implementada por um tipo de tecnologia, serviços web, e o facto destes serviços fazerem uso do HyperText Transfer Protocol (HTTP) que permitiu a sua adoção em massa. O uso de serviços web permite que estes possam ser combinados com outros serviços para composição de um novo serviço com novas funcionalidades. É também possível ainda o uso de um Enterprise Service Bus (barramento de serviços) para integração de vários serviços web, desta forma é possível obter uma grande variedade de serviços num único ponto de acesso.

3.4.2 REST

É muito usual encontrar implementações desta arquitetura, SOA, baseadas no protocolo de comunicação Representational State Transfer (REST). Este, assenta numa comunicação entre cliente e servidor sem persistência de estado, onde cada novo pedido de informação não está de forma nenhuma relacionado com algum pedido efetuado anteriormente. O REST é constituído por endereços únicos e cada um destes representa um recurso; uma página web é um recurso. Existem quatro operações que se podem efetuar com este protocolo: GET, POST, UPDATE e DELETE. O primeiro permite a consulta de informação, seja uma página web ou outro recurso. O POST permite a criação de informação no servidor e é comum ser usado no preenchimento de formulários. O UPDATE efetua a atualização de informação previamente criada e o DELETE permite remover informação do servidor.

Este protocolo é comum ser usado com HTTP e a sua facilidade de utilização torna-o num dos protocolos mais utilizados para comunicação entre redes de computadores.

3.4.3 Java & Java EE

Java [15] é uma linguagem de programação de alto nível desenvolvida para a Sun Microsystems em 1995 e neste momento propriedade da Oracle Corporation¹. Esta segue os princípios da orientação a objetos e devido à sua simplicidade e portabilidade é hoje das linguagens mais populares em todo o mundo.

A portabilidade que esta linguagem apresenta é conseguida devido à Java Virtual Machine (JVM) que permite que qualquer programa escrito em Java possa ser executado em qualquer SO. Esta estratégia é também responsável por possíveis perdas de performance em algumas operações em comparação com linguagens nativas, ao mesmo tempo permite que certos aspetos sejam otimizados tornando outras operações mais eficientes.

A plataforma Java Enterprise Edition, Java EE, é uma coleção de especificações para o desenvolvimento e implementação de aplicações empresariais, como aplicações que são executadas em servidores web. As especificações descrevem serviços, Application programming interface (API)s e protocolos e são constituídas por 13 núcleos tecnológicos. Algumas das funcionalidades que mais se destacam

¹Oracle, <http://www.oracle.com>

relacionam-se com o acesso a bases de dados, comunicação entre várias aplicações cliente Java, implementação de *contentores* de serviços web, etc. O Java EE na forma de produto é tipicamente providenciado por empresas de desenvolvimento de software na forma de uma aplicação-servidor que fornece as classes que implementam as normas definidas na especificação.

A adoção do Java como linguagem de programação deste trabalho em detrimento de outras apenas se relaciona com a experiência pessoal e níveis de conhecimentos já adquiridos durante todo o percurso universitário, tornando esta uma escolha óbvia a usar.

3.4.4 JBoss AS & RESTeasy

JBoss² é uma divisão de desenvolvimento de software de uso livre e código fonte aberto da RedHat³ e responsáveis pelo desenvolvimento do JBoss Application Server, mais conhecido por JBoss AS, e pelo RESTeasy. JBoss AS é um servidor web compatível com a especificação Java EE 3.4.3, RESTeasy é uma implementação certificada e portátil da especificação JAX-RS.

JBoss AS é baseado no Tomcat⁴ mas desenhado para aplicações de média a grande dimensão devido às suas capacidades arquiteturais modulares. Este conta também com uma das maiores comunidades de desenvolvimento em aplicações do género.

A especificação JAX-RS define um conjunto de funcionalidades que permitem o desenvolvimento de aplicações web, entre elas, a composição de serviços web. Esta especificação é necessária durante a compilação, mas como apenas define as funcionalidades, necessita de algo que implemente tais funcionalidades durante a execução. O RESTeasy⁵, Apache CXF⁶ e Jersey⁷ são algumas das frameworks capazes de implementar esta especificação.

Outras razões que levaram à escolha destas duas tecnologias para implementação deste trabalho relacionam-se com a performance que estas apresentam [16], facilidade de instalação e configuração e também à possibilidade de seleção de quais as funcionalidades que são necessárias para execução de acordo com customização de perfis [17].

3.4.5 Maven

Maven [18] é uma ferramenta *open-source* de gestão de projetos, principalmente para Java, mas também com suporte a outras linguagens de programação. Projetado para diminuir o trabalho necessário para configuração dos projetos, esta ferramenta é capaz de gerir a construção do projeto, criar a documentação e gerir dependências uma vez que trabalha de forma centralizada. Assim, os vários elementos de uma equipa de desenvolvimento podem todos trabalhar com as mesmas versões de plugins ou bibliotecas necessárias. Também a necessidade de descarregar novas bibliotecas é deixada ao encargo desta ferramenta que usa repositórios próprios de onde são transferidos.

As capacidades de resolução de problemas desta ferramenta em grandes projetos ou com vários elementos de equipa são notórias e uma vez que é possível a sua integração com vários ambientes de desenvolvimento (Integrated Development Environment (IDE)) adotou-se a sua utilização para gestão do desenvolvimento deste trabalho, permitindo que, se existir uma continuação deste por outros

²JBoss, <http://www.jboss.com>

³RedHat, <http://www.redhat.com>

⁴Servidor web em Java que não implementa a norma Java EE na totalidade.

⁵RESTeasy, <http://www.jboss.org/resteasy>

⁶Apache CXF, <http://www.cxf.apache.org>

⁷Jersey - Java, <http://www.jersey.java.net>

membros, não existirão quaisquer problemas de compatibilidades ou com dependências de bibliotecas ou mesmo do SO utilizado.

3.4.6 *Single Page Application*

A arquitetura Single Page Application baseia-se em páginas web compostas unicamente por uma interface gráfica no qual todo o conteúdo apresentado é carregado dinamicamente. Este tipo de websites surgiram para oferecer uma experiência de utilização mais próxima da que se obtém com aplicações nativas e mais fluída devido à capacidade que estas páginas possuem de atualizarem partes específicas sem efetuar um carregamento completo; isto só é possível usando uma técnica de desenvolvimento denominada por Asynchronous JavaScript and XML (AJAX), que permite efetuar pedidos de informação em segundo plano e apresentar essa informação na página web sem existir a necessidade de uma completa atualização [19].

O desenvolvimento de páginas web baseadas nesta arquitetura é sempre mais complexo pois envolve um completo entendimento do website como um todo, caso contrário será escrito mais código e mais complexo. Muitas vezes, para se chegar uma boa aplicação web, só mesmo depois de resolver o mesmo problema várias vezes e efetuar várias correções ao código já escrito, isto acontece em casos onde seja necessário substituir pedaços de código que resolvem casos específicos para funcionarem de forma mais geral e de forma mais consistente.

3.4.7 *Backbone.js & jQuery*

Backbone.js e jQuery são duas frameworks desenvolvidas em Javascript, a primeira tem o objetivo de fornecer uma estrutura modular às aplicações web enquanto o jQuery permite o desenvolvimento mais rápido mas não estruturado.

O Backbone.js cria uma abstração entre a programação e a interface do utilizador ao aplicar alguns dos conceitos da arquitetura de desenvolvimento Model View Controller (MVC) que se divide em três componentes, os modelos, as vistas e os controladores. Do ponto de vista do programador, os conceitos utilizados pelo Backbone.js da arquitetura MVC são apenas os modelos e as vistas. Internamente os modelos fazem uso dos controladores para execução várias funções, tais como consulta de informação. No Backbone.js os modelos são a representação dos dados em objetos que podem ser criados, alterados, validados ou destruídos. Como referido estas operações só são possíveis devido à incorporação do conceito de controlador dentro dos próprios modelos, permitindo que estes executem ações. As vistas são representações das interfaces de utilizador na página web e que normalmente espelham a informação contida num modelo, os seja, o modelo contem a informação enquanto a vista transforma essa informação e apresenta-a numa interface gráfica: Tais interfaces têm também a capacidade de manipulação dos dados do modelos e de definição de eventos no próprio modelo.

Relativamente ao jQuery, esta é uma framework da qual o Backbone.js depende para uso interno. Esta permite ao programador a seleção de controlos HyperText Markup Language (HTML) de forma mais rápida e simples. Fornece também funcionalidades que permitem criar simples animações nas páginas, criação de eventos, manipulação de atributos dos vários elementos da página, etc. Além de todas as funcionalidades que permite, existem ainda outros projetos associados com objetivos de alargar o leque de funcionalidades, como por exemplo, a criação de bibliotecas com objetos gráficos como seletores de datas, etc.

3.4.8 Twitter Bootstrap

O Twitter Bootstrap [20] é uma coleção de acessórios para a criação de páginas web que vão desde templates em Cascading Style Sheets (CSS) para formulários, botões, gráficos, a extensões desenvolvidas em Javascript que permitem animações ou interações com a página web. Desenvolvido pela equipa do Twitter com objetivo inicial de criar uma consistência maior entre as várias ferramentas desenvolvidas por terceiros e até mesmo internas, tornando-se público em 2011 e ganhando assim uma enorme popularidade por toda a comunidade de programadores web devido à sua facilidade de utilização e conjunto de funcionalidades.

Esta ferramenta permite a criação de web sites automaticamente adaptáveis aos vários browsers dos diferentes dispositivos de tal forma que o programador não necessita de se preocupar com problemas de incompatibilidade entre estes. Suporta também um vasto conjunto de tabelas com várias propriedades, desde ajuste de tamanhos dinâmicos, diferentes cores para linhas ímpares e pares, etc. é também oferecido um conjunto muito grande de componentes gráficos e de fácil customização como um sistema de notificações ou de navegação, paginação, entre outros.

Devido ao grande sucesso que esta ferramenta conseguiu, é agora um dos projetos mais populares no GitHub [21] e conta com um suporte muito bom e desenvolvimento de terceiros muito ativo.

3.4.9 MySQL, JPA & Hibernate

MySQL [22] é o Sistema de Gestão de Base de Dados (SGBD) open-source existente mais popular e é desenvolvido e distribuído pela Oracle [23]. Este sistema é de tal forma popular que existem alguns *forks* para outros projetos, um dos mais conhecidos é o MariaDB [24] que é desenvolvido por alguns dos criadores originais do MySQL. A linguagem Structured Query Language (SQL) faz também parte do MySQL e tem como principais funções a manipulação de dados da base de dados, ou seja, consulta, inserção, atualização e remoção. O uso deste tipo de tecnologias para gestão de persistência de informação deve-se ao facto de se tratarem de aplicações altamente especializadas que conseguem lidar com a informação de uma forma extremamente rápida e eficiente, permitindo que, por exemplo, consultas de informação mediante determinados filtros, seja feita numa pequena parte de segundo.

O Java Persistence API (JPA) é uma API do Java definida para suportar persistência de objetos Plain Old Java Object (POJO), objetos simples do Java sem lógica programática. Esta API é apenas uma definição de métodos e classes que são implementados durante a execução por *frameworks* Object-Relational Mapping (ORM). Algumas destas frameworks mais conhecidas são o Hibernate [25], EclipseLink [26] ou OpenJPA [27] e a sua utilização permite efetuar a mudança de uma framework para outra sem que para isso se tenha de alterar o código fonte.

O principal objetivo desta API passa por libertar do programador a gestão das base de dados, isto não implica que não necessite ter conhecimentos acerca de base de dados relacionais, mas sim que poderá usar objetos em Java que são mapeados diretamente para tabelas de uma base de dados. Este padrão resulta do esforço de várias entidades para se criar uma norma que foi essencialmente inspirado no Hibernate mas que também contou com outras *frameworks* que passaram mais tarde a ser uma implementação do JPA. Foi também desenvolvendo juntamente com este projeto uma linguagem denominada de Java Persistence Query Language (JPQL) cujas funcionalidades permitem a consulta, inserção, atualização e remoção de dados.

Relativamente ao Hibernate[25], este é uma implementação do JPA, mas não se deixa ficar por aqui pois este pode ser usado independentemente e sem seguir os padrões do JPA, mas criando dependências na aplicação desta framework.

3.4.10 SAML

Security Assertion Markup Language (SAML) [28] é uma norma em Extensible Markup Language (XML) que permite que domínios seguros troquem informação de autenticação e autorização de utilizadores. Com base nesta troca de informação, é possível a um sistema aferir informação acerca da identidade, características e direitos de um utilizador associado num outro sistema independente mas confiável.

O SAML é composto por quatro blocos: alegações, protocolos, ligações e perfis. Por alegação entende-se um conjunto de informações, atributos ou decisão de autorização fornecidas por uma entidade acerca de um determinado sujeito a outra entidade. No bloco de protocolo, as mensagens trocadas definidas no SAML permitem uma série de pedidos e respostas que permitem efetuar todas as operações previstas: pedir a uma entidade alegações de um utilizador, pedir uma autenticação, pedir um fecho de sessão, etc. O bloco de ligação define como se podem ligar as mensagens protocoladas e normas de comunicação entre os diferentes sistemas uma vez que os vários sistemas podem não suportar os mesmos protocolos ou normas e é necessário encontrar uma forma de comunicar entendida entre ambos. Por fim, o bloco perfil do SAML define um conjunto de restrições e extensões para se poder atingir um determinado fim, ou seja, neste bloco estão presentes todos os dados e metadados que dão a conhecer ao sistema onde o utilizador se pretende autenticar, por exemplo, as cifras usadas para encriptação das mensagens trocadas, qual o tipo de chave assimétrica e respetiva chave pública para encriptação dos dados pessoais do utilizador, etc.

Com o SAML é também possível usar o método de autenticação Single Sign-On (SSO). Este permite que após uma autenticação válida num provedor de identidades, não seja necessário efetuar novamente um pedido de autenticação na consulta de recursos de outras entidades que confiem no mesmo provedor de identidades.

Resumo

Foram dadas a conhecer de forma muito geral as ferramentas e conceitos arquiteturais mais relevantes utilizados durante o processo de desenvolvimento deste trabalho. Várias outras ferramentas foram utilizadas, ainda assim a sua escolha em detrimento de outras não afetaria a condução do sistema como um todo e desta forma não foram referidas, por exemplo, a framework usada para criação de ficheiros de *logs*.

A sobreposição de tecnologias, como o caso da arquitetura SOA implementada com o protocolo REST, permite que sejam desenvolvidas, por exemplo, aplicações móveis baseadas nesta aplicação, uma vez que esta opção de implementação torna simples a comunicação neste género de ambientes. Também o uso do servidor JBoss AS em conjunto com o RESTeasy, permitem a continuidade de desenvolvimento deste projeto usando duas das aplicações com maior comunidade de utilizadores. A adoção do SAML para proceder à autenticação de utilizadores permite que a qualquer momento seja possível a adição ou remoção de provedores de identidades, tornando este sistema disponível a utilizadores de outros sistemas. Todas estas aplicações referidas anteriormente permitiram criar a aplicação que será seguidamente apresentada.

3.5 SAAL

A solução SAAL segue uma arquitetura SOA que lhe incute características de modularidade e permite que os vários serviços web desenvolvidos sejam combinados com serviços de terceiros. A aplicação web criada faz também parte desta solução que irá aqui ser apresentada com mais detalhe e profundidade. Inicialmente será feita uma introdução a alguns termos usados no contexto desta aplicação e que terão de ser clarificados. Seguidamente pretende-se focar o conjunto de casos de utilização que os serviços desenvolvidos estão vocacionados a dar resposta. Será também justificada a forma como é processada a autenticação dos serviços juntamente com o gestor de identidades da UA, bem como o processo em que decorre a autorização de acesso aos mesmos. Será ainda descrita e categorizada a base de dados e detalhadas algumas das características mais importantes dos serviços desenvolvidos bem como respetiva função. Por fim serão apresentadas algumas imagens do web site implementado.

O projeto de desenvolvimento dos serviços web exige o uso de um gestor de projetos capaz de lidar com todos os problemas existentes que passam desde a própria configuração do projeto, integração com um IDE ou até mesmo a gestão de dependências de bibliotecas. Tal necessidade levou à adoção da ferramenta Maven 3.4.5; relativamente ao projeto da aplicação web não foi adotado nenhum gestor de projeto.

3.5.1 Taxonomia

Dentro deste sistema existem alguns conceitos próprios usados apenas no âmbito deste trabalho para uso académico. De seguida serão apresentados alguns conceitos que necessitam ser contextualizados.

Edição Representação de uma disciplina num determinado ano letivo e num determinado semestre.

A edição é constituída por módulos de exercícios e turmas de alunos.

Regente Professor responsável por uma Edição de uma disciplina e que envolve funções de gestão da disciplina.

Exercício Obrigatório existe a necessidade de uma submissão correta por parte dos alunos de forma a ser considerado como aprovado para uma nova fase de avaliação, como a ida a um exame.

Exercício Individual não existe a possibilidade de efetuar submissões em grupos de vários alunos.

Grupo o grupo é uma representação de pelo menos um aluno. Poderão ser criados grupos de vários alunos para submissão de trabalhos.

Módulo Os conteúdos a lecionar pelas várias disciplinas estão divididos em módulos.

Exercícios Conjunto de exercícios que envolvem os conteúdos lecionados de terminado módulo.

Teste de Exercício Os testes de um exercício são compostos por ações com dados de entrada e de saída. A ação são comandos que permitem a compilação e execução do código fonte submetido mas pode também representar uma ação da linha de comandos dos sistemas UNIX.

Submissão A submissão é realizada pelos alunos e é composta pelo código fonte da resolução do exercício e em caso de necessidade por ficheiros extra.

3.5.2 Casos de Utilização

O diagrama de casos de utilização é apresentado pela Figura 3.2 sob a notação Unified Modeling Language (UML) [29]. Nele encontram-se os vários utilizadores do sistema, o Aluno, Professor, Regente e Administrador do site. Ainda nos utilizadores, é possível observar que existe uma generalização de Professor para Regente bem como de Regente para Administrador, isto é, um regente tem as suas tarefas específicas acrescidas das tarefas de professor, da mesma forma um administrador do sistema também tem as tarefas do regente e por sua vez do professor. Relativamente ao aluno, não existe nenhum tipo de sobreposição entre o seu papel e o papel desempenhado pelos outros utilizadores, nem mesmo com o Administrador, só desta forma o desempenho do aluno será considerado válido uma vez que se garante a não intervenção do professor no seu percurso.

Desde o início do desenvolvimento deste trabalho que se adotou o uso de nomenclatura em inglês para todos os termos, desta forma o diagrama seguinte é também apresentado em língua inglesa.

Uma observação mais atenta do diagrama mostra que este está dividido também em diagrama de pacotes referenciados pelas cores de cada caso de utilização, sendo que cada pacote está relacionado diretamente com apenas um dos vários utilizadores existentes.

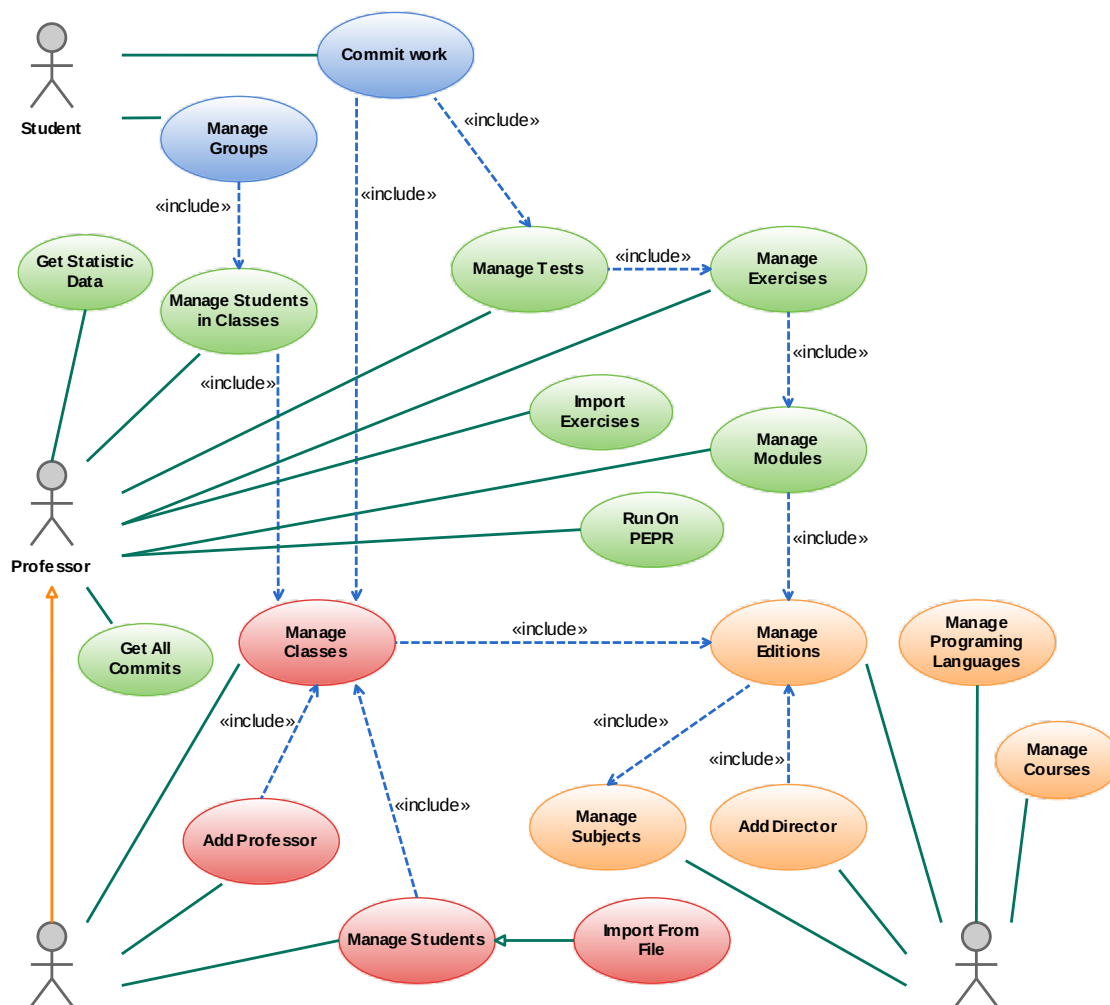


Figura 3.2: Diagrama de Casos de Utilização

De acordo com a linguagem de modelação UML, as setas no diagrama com a descrição «*include*» dão a conhecer que determinado caso de utilização necessita de outro de forma a ser concluído com sucesso, ou seja, para se efetuar a gestão de alunos é necessário criar primeiro as turmas. Desta forma será mais fácil perceber os casos de utilização apresentados. Também de salientar que alguns casos de utilização podem ser decompostos noutros casos, por exemplo, o caso de utilização de gestão de turmas, inclui consultar, criar, editar e apagar turmas.

Administrador

Como seria de esperar numa aplicação deste género, o Administrador irá sempre ter algum trabalho inicial de forma inserir alguma informação necessária à aplicação e sem a qual nada será possível efetuar.

Gerir Linguagens de Programação Gestão das várias linguagens de programação que a aplicação terá suporte, isto inclui a configuração de um SO onde irá ser executado o código fonte correspondente.

Gerir Cursos A gestão de cursos não é obrigatória, mas deve ser tida em conta uma vez que permite aos alunos selecionarem o seu curso permitindo uma melhor categorização destes.

Gerir Disciplinas Como o nome indica, permite efetuar toda a gestão de disciplinas

Gerir Edições As edições representam uma disciplina num determinado ano e num determinado semestre que é regida por determinados professores.

Adicionar Regente A adição de regentes passa pela gestão de edições, desta forma exige-se que a sua adição implique criar ou editar uma Edição.

Regente

O Regente é o utilizador responsável por realizar a gestão da edição pela qual se encontra responsável, as suas ações passam por:

Gerir Turmas A gestão de turmas passa pela criação destas.

Adicionar Professores Esta ação envolve a gestão das turmas visto que cada turma tem pelo menos um professor associado.

Gerir Alunos É o regente que tem a responsabilidade de adicionar ou remover os alunos da disciplina.

Importação por Ficheiro A adição de alunos para a disciplina pode ser efetuada pela importação de ficheiros exportados da plataforma PACO.

3.5.3 Professor

Este terá um leque mais alargado de responsabilidades que passam pela gestão das suas turmas e de exercícios.

Gerir Alunos na Turma Esta gestão passa por adicionar alunos às suas turmas já previamente adicionados à edição pelo regente e remover alunos das suas turmas colocando-os numa turma “alunos sem turma”.

Gerir Módulos Gestão de módulos (conjuntos de exercícios) relativos à edição na qual o professor está inserido.

Gerir Exercícios A cada módulo são associados exercícios que terão de ser resolvidos pelos alunos.

Importar Exercícios Os exercícios de determinada edição podem ser adicionados fazendo uso da funcionalidade de importação de exercícios de outras edições. Esta funcionalidade importa módulos, exercícios, testes e ficheiros auxiliares.

Gerir Testes Os testes são parte integrante de cada exercício.

Consultar Informação Estatística Fazer *download* de um ficheiro no formato *Comma separated values (CSV)* com toda a informação estatística dos alunos da edição.

Consultar Submissões Consulta das submissões efetuadas pelos alunos e análise dos resultados das execuções.

Executar no PEPR Os professores têm também a possibilidade de executar código no PEPR que poderá servir para confirmar o resultado esperado para os vários exercícios.

Aluno

O aluno será a entidade com menos funcionalidades, estas passam apenas pela gestão de grupos e submissão da resolução de exercícios.

Gestão de Grupos Os alunos têm a possibilidade de trabalhar em conjunto e desta forma efetuar submissões em conjunto, desde que o professor assim o permita para cada exercício.

Submissão de trabalhos É o ato de entregar a resolução de um exercício para ser analisado e verificar se passa corretamente todos os testes definidos.

3.5.4 Autenticação e Autorização

Todos os utilizadores do sistema necessitarão de uma autenticação para poderem usufruir deste, visto esta ser uma aplicação para ser usada dentro da instituição UA no âmbito de disciplinas de programação de computadores, foi óbvia a escolha de usar o sistema provedor de identidades desta instituição para autenticação dos utilizadores, já para autorização foi necessário implementar um sistema de regras baseado em listas de acesso e grupos.

Autenticação

A figura 3.3 demonstra a sequência envolvida durante um processo de autenticação entre um utilizador e o IdP. Esta troca de mensagens entre ambas entidades é realizada fazendo uso do protocolo SAML 3.4.10. A criação do sistema de autorização envolveu uma forma capaz de efetuar a autorização apenas dos serviços web, uma vez que a aplicação web serve apenas para apresentação de informação e que toda a gestão da infra-estrutura é realizada através dos serviços. Desta forma também se garante que independentemente da forma de acesso (aplicação móvel, web site, etc.) o utilizador é sempre obrigado a efetuar primeiro a autenticação junto da UA.

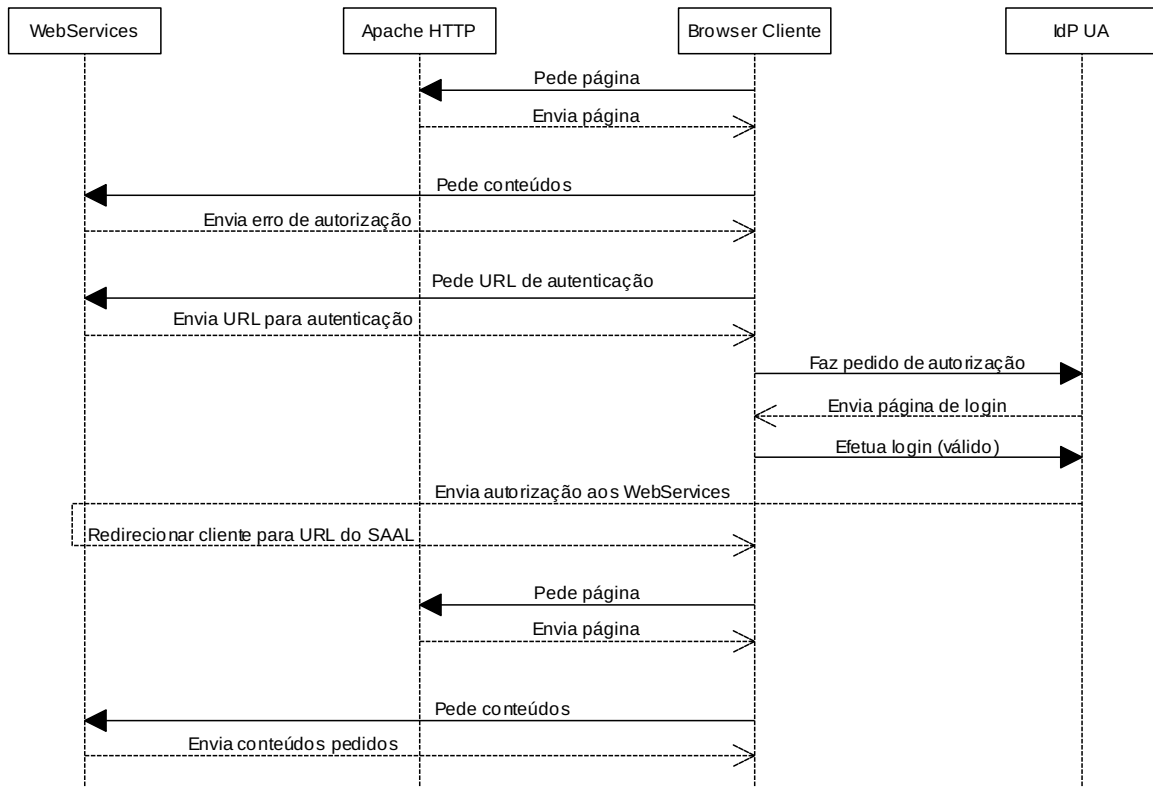


Figura 3.3: Sequência de autenticação com SAML

A interação começa com o utilizador a aceder ao sistema através do seu browser com os seguintes passos:

1. O utilizador acede ao endereço web do sistema e recebe a página juntamente com os temas e o código Javascript
2. São efetuados pedidos AJAX aos serviços desenvolvidos requerindo informação do utilizador
3. Os serviços respondem com erro de autenticação. O browser faz então um pedido para obtenção do URL de autenticação e redireciona o utilizador para esse mesmo endereço.
4. Uma vez na página do IdP o utilizador terá de inserir as suas credenciais e após autenticação válida, o browser é reencaminhado para os serviços, onde será criada uma sessão válida.
5. De seguida o browser é novamente reencaminhado para o endereço web do sistema e o processo é novamente repetido do ponto 1. Desta vez já não existem as falhas de autenticação e como tal será devolvida a informação para preenchimento da página.

Ao ocorrer a exceção de falta de autenticação nos pedidos efetuados ao SAAL, este deveria efetuar automaticamente o redirecionamento do browser do utilizador para o IdP, onde o utilizador procederia à sua autenticação. Este procedimento não pode acontecer desta forma uma vez que vai limitar o uso dos serviços para autenticação noutras plataformas que não façam uso de um browser. Devido a este problema optou-se pela forma acima referida: efetuar um pedido para obtenção do endereço

eletrónico para o qual se deverá reencaminhar o utilizador. Este endereço é construído tendo em consideração a hora do servidor, um identificador único para cada pedido e a alguma informação necessária requisitada pelo IdP.

É ainda necessário referir que esta implementação faz uso de um par de chaves assimétricas, em que uma das chaves é privada e a outra de acesso público. Esta é passada sempre que está em curso um novo pedido de autenticação. Existe um serviço desenvolvido exclusivamente para fornecer toda a informação necessária ao IdP da UA.

Autorização

Como já referido acima, o sistema de autenticação deste trabalho é providenciada pelo IdP da instituição UA. Uma vez que a informação dos utilizador devolvida por este é escassa, não existe forma de criar um sistema de autorização capaz de responder às necessidades exigidas pelo SAAL. Desta forma foi necessário a implementação de um sistema que seja capaz de dar resposta às necessidades exigidas.

A solução encontrada passa por um sistema bastante versátil baseado em listas de acesso individuais e de grupos capaz de garantir quais os recursos que cada utilizador tem permissões de consulta. Por cada pedido efetuado será consultada uma lista de permissões que relaciona o grupo ao qual o utilizador pertence (Aluno, Professor ou Administrador) com o serviço em causa e no caso do grupo não ter acesso ao recurso especificado será lançada uma exceção de falta de privilégios. Ainda assim, e por exemplo para um aluno, o administrador poderá definir que este em particular tem permissões para uma tarefa específica dos Professores e desta forma também este poderia executar tal tarefa. A lista de permissões encontra-se armazenada em base de dados, mas é carregada para memória durante a inicialização dos serviços, permitindo assim maior desempenho uma vez que não necessita do acesso constante à base de dados. Não são previstas alterações de permissões aos grupos, pelo que no caso de acontecer, será necessário efetuar a reinicialização da aplicação. Relativamente às permissões individuais, estas são sempre consultadas quando o utilizador efetuar autenticação no sistema.

Ainda relativamente ao sistema de autorização, como o IdP não fornece informação acerca das funções do utilizador dentro da universidade, estes são todos considerados alunos numa primeira fase. Para um professor, que no sistema é considerado um aluno, passar a desempenhar as suas funções como professor, terá que ser adicionado como tal numa turma pelo Regente de uma edição de uma disciplina. Só após esta fase que o utilizador deixa o grupo dos estudantes e passa a fazer parte do grupo de professores.

3.5.5 Base de Dados

Para o desenvolvimento de uma aplicação com esta complexidade foi necessário explorar as potencialidades do uso de bases de dados relacionais para persistência da informação dos utilizadores, disciplinas, turmas, exercícios, etc. Como referido na secção MySQL, JPA & Hibernate 3.4.9, adotou-se a tecnologia JPA do Java EE 3.4.3 com a implementação em Hibernate e gestor de base de dados MySQL. A utilização do JPA permite que a qualquer momento seja possível migrar de Hibernate para outra implementação sem problemas de compatibilidades ou dependências.

A base de dados desenvolvida é apresentada na Figura 3.4 e baseia-se no modelo de dados relacional, este por sua vez é assente em modelos matemáticos sobre lógica dos predicados e teoria dos

conjuntos [30]. Na implementação da base de dados todos os nomes utilizados, tabelas e campos, foram escritos na língua inglesa.

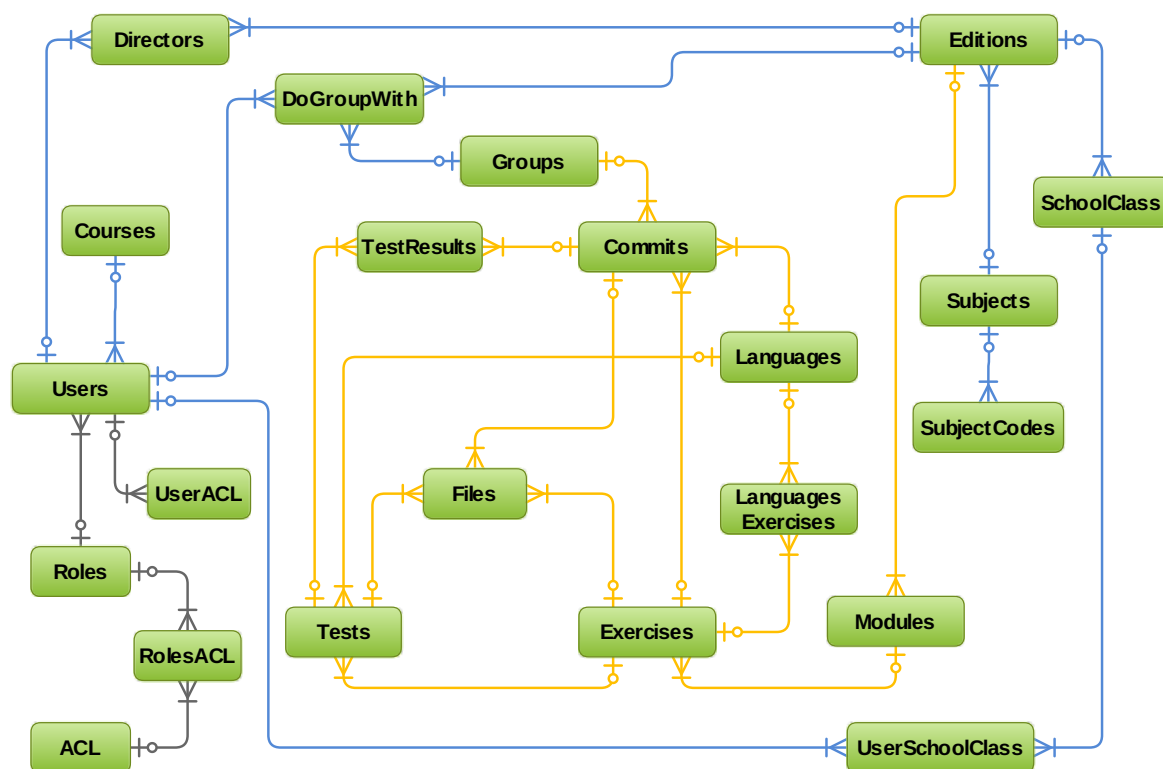


Figura 3.4: Diagrama de tabelas da base de dados

Durante todo o processo de modelação da base de dados foram tidas em consideração as restrições a colocar nos vários campos das tabelas de forma a alcançar uma base de dados sem problemas de consistência ou falta de eficiência, ao mesmo tempo que se encontre munida de uma flexibilidade que lhe permita uma adaptação, a outras realidades ou necessidades.

No diagrama de base de dados apresentado acima, apenas os nomes e relações das tabelas criadas estão visíveis. É também possível observar que as relações estão divididas por três cores que representam três componentes com diferentes funcionalidades a que este modelo dá suporte. As relações a cinzento refere-se ao sistema de gestão de permissões, a cor azul ao sistema de gestão curricular e por fim a cor laranja à gestão de exercícios e submissões.

Gestão de Permissões

Esta componente da base de dados é a que permite suportar as funcionalidades apresentadas na secção Autorização 3.5.4, cujo modelo da base de dados é apresentado na Figura 3.5.

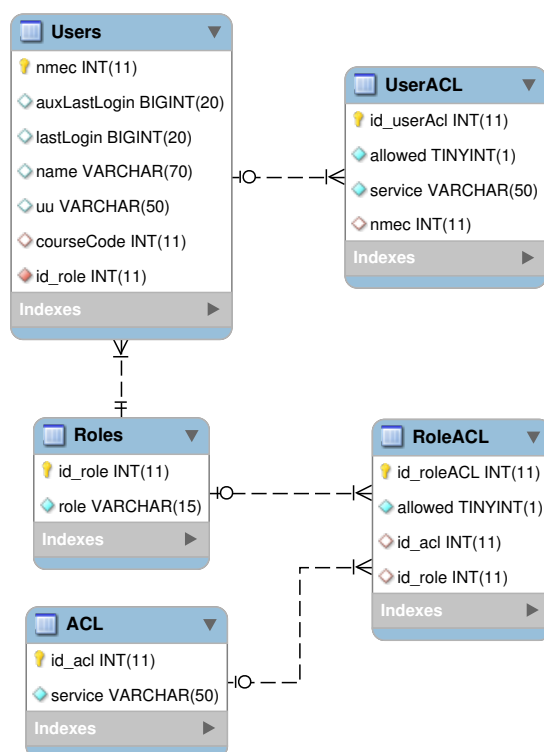


Figura 3.5: Modelo relacional de gestão de permissões

Como é visível neste diagrama, esta componente da base de dados é constituída por cinco tabelas. Destacam-se a tabela de *Roles* cuja funcionalidade passa por associar cada utilizador em grupos (Estudante, Professor ou Administrador). A tabela *ACL* contém todos os identificadores únicos de cada serviço desenvolvido e a relação desta tabela com a tabela *Roles* permite criar uma lista de permissões que define para cada grupo quais os serviços que podem ser acedidos e cuja informação é persistida na tabela *RoleACL*. Finalmente a tabela *UserACL*, que por omissão se encontra vazia para todos os utilizadores, também relaciona os vários serviços com cada um dos utilizadores existentes e permite alterar as permissões do grupo de um serviço para um determinado utilizador.

Gestão Curricular

Esta secção da base de dados, a gestão curricular, concretiza toda a gestão que existe num ambiente escolar, que envolve a gestão de disciplinas, edições, turmas, alunos, etc.

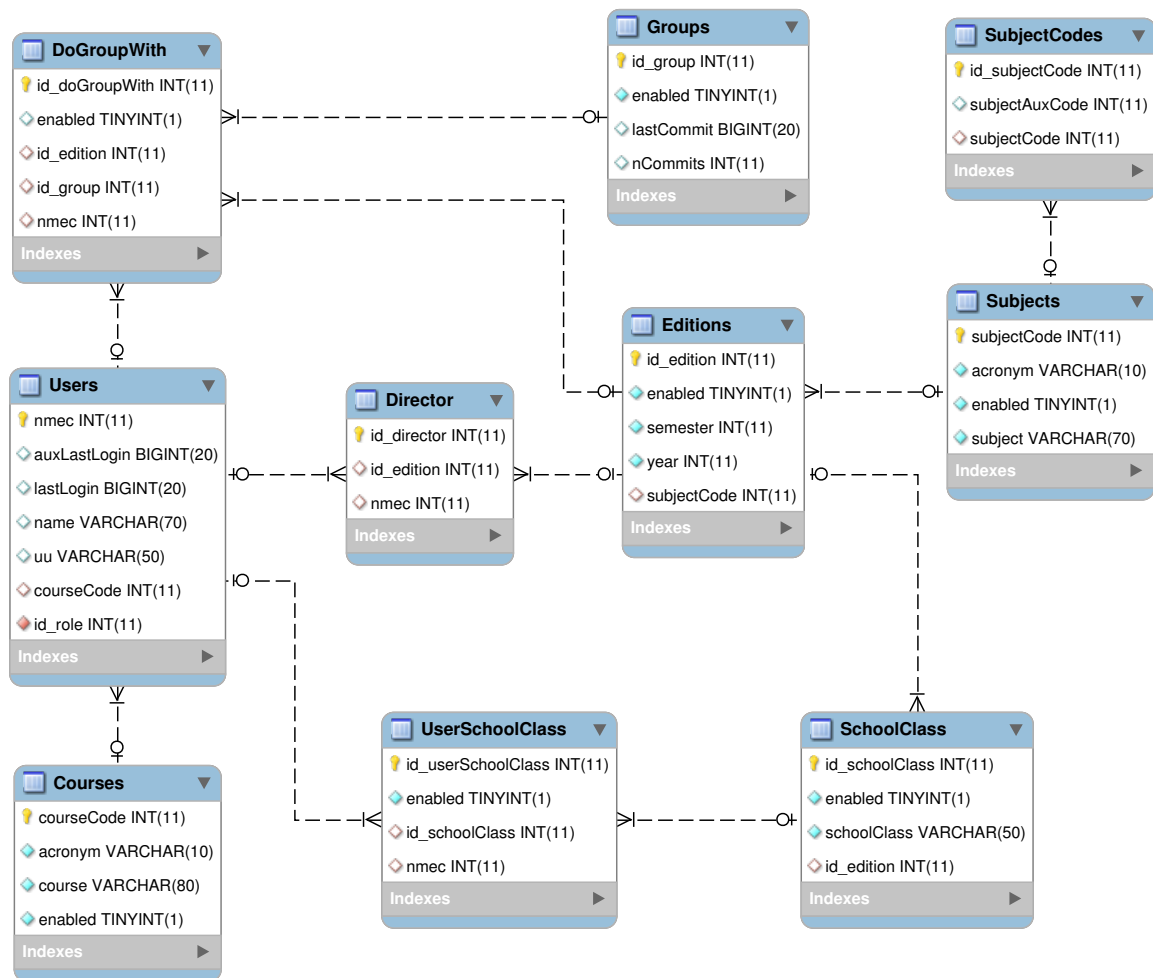


Figura 3.6: Modelo relacional de gestão curricular

Na Figura 3.6 são apresentadas as várias tabelas que suportam as funcionalidades implementadas na gestão curricular. Começando do lado direito do diagrama, a tabela *Subjects* (disciplinas) relaciona-se com a tabela *SubjectCodes* pois acontece com alguma frequência uma disciplina ser constituída por mais que um código de identificação; estas, as disciplinas, contêm várias *Editions* (edições) e para cada uma destas edições tem de existir pelo menos um *Director* (Regente). As edições também são constituídas por várias turmas, *SchoolClass*, que por sua vez são constituídas por vários alunos e professores, *UserSchoolClass*. Ainda associado às edições, existe a tabela *DoGroupWith* que em conjunto com a tabela *Groups* originam a possibilidade da criação de grupos de alunos para submissão de trabalhos. Para finalizar, existem ainda os cursos aos quais os alunos pertencem.

Gestão de Exercícios

A gestão de exercícios, Figura 3.7, permite a categorização de exercícios por módulos bem como definição de testes para cada exercício. Tais testes serão usados durante a execução dos exercícios submetidos pelos alunos.

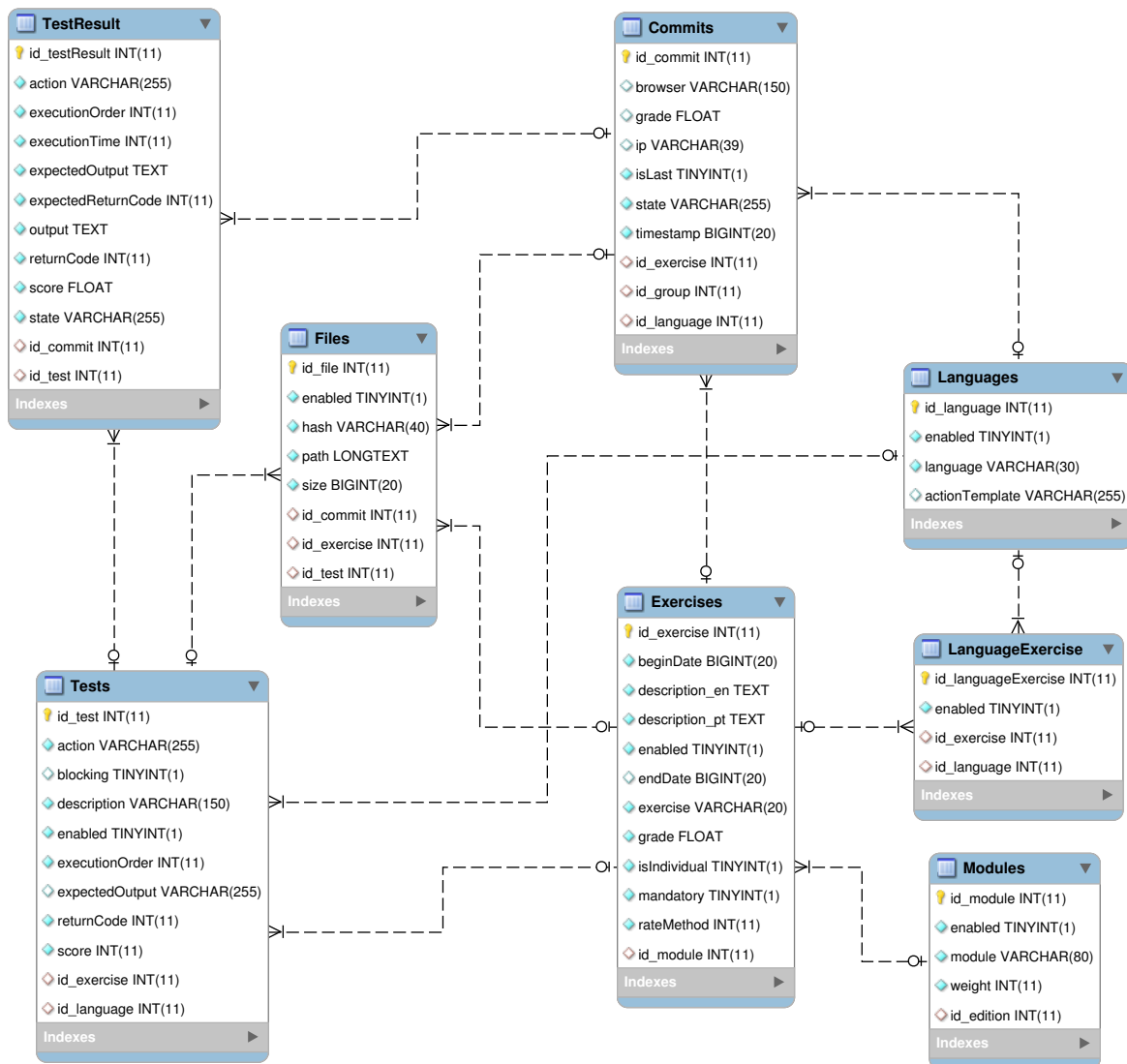


Figura 3.7: Modelo relacional de gestão de exercícios

Esta componente é constituída por oito tabelas. Os exercícios são parte integrante dos módulos e têm uma lista de linguagens de programação das quais pelo menos uma terá de ser usada para resolução do exercício em causa; existem também ficheiros auxiliares que podem ser associados aos exercícios para ajuda da resolução, por exemplo, um ficheiro de texto que deve ser lido e trabalhado pelo programa do aluno. Ainda nos exercícios, estes também são constituídos por testes, da tabela *Tests*, que contem toda a informação relativa aos testes a executar após cada submissão dos alunos. A tabela *Commits* representa todas as submissões efetuadas pelos alunos, uma vez que um exercício poderá ser resolvido em várias linguagens de programação, também as submissões têm de referir qual a linguagem escolhida. Associado às submissões existe ainda a tabela *TestResult* com toda a informação da execução dos vários testes definidos para o exercício em causa e linguagem de programação em questão.

De referir que os exercícios e testes podem ter ficheiros associados na tabela *Files*, já as submissões têm obrigatoriamente um ficheiro submetido pelo aluno.

Stored Procedures & Views

Em alguns dos serviços desenvolvidos, como a consulta de edições ou de turmas, é exigido um grande esforço de processamento por parte da base dados. Este esforço de processamento resulta da apresentação de informação estatística que estes serviços devolvem. A adoção de Hibernate criou uma abstração da base de dados demasiado elevada e criou alguns problemas de desempenho em algumas consultas mais específicas. De forma a resolver esta adversidade foi necessário empregar o uso direto de SQL e criar dois procedimentos e duas vistas na base de dados que são depois consultados usando novamente Hibernate e sem perda de performance.

Os procedimentos desenvolvidos permitem a consulta de informação estatística de uma edição e de uma turma, as vistas aplicam-se à informação obtida de forma a trabalhar esse conjunto de dados adquiridos.

Seria de esperar que para consulta da informação estatística de uma edição consumisse o procedimento de consulta de informação estatística de uma turma. Esta reutilização de procedimentos passaria pela consulta de informação por cada uma das turmas dessa edição e sumariação da mesma. Esta forma permitiria a reutilização de código, uma das boas práticas aplicadas à programação, mas do ponto de vista da performance este procedimento seria muito pesado computacionalmente. Para minimizar o problema da complexidade computacional, o procedimento de consulta de informação estatística de uma edição foi desenvolvido apenas com esse objetivo, obtendo desta forma um ganho de performance.

3.5.6 Serviços

Nesta seção pretende-se efetuar uma contextualização do SAAL relativamente aos vários serviços desenvolvidos e cujas funcionalidades ou características internas dos mesmo exijam uma explicação mais detalhada. Esta clarificação vai desde detalhes relacionados com a segurança de acesso aos serviços como aos problemas de desempenho resultantes da complexidade que alguns envolvem.

Já foi referido anteriormente na secção Autenticação e Autorização 3.5.4, a cada serviço está associado uma regra para cada um dos diferentes grupos de utilizadores. A lista completa de todos os serviços criados bem como as regras por cada grupo de utilizadores podem ser consultadas na tabela em anexo na secção Lista de Serviços 5.2.

No total foram desenvolvidos oitenta e quatro serviços REST dos quais quatro estão relacionados com o sistema de autenticação e fim de sessão e cinco promovem a interação com a PEPR. Estes nove serviços são de acesso público, desta forma não necessitam de qualquer tipo de autorização e também não são apresentados na lista em anexo dos serviços desenvolvidos. Dos restantes setenta e cinco serviços, todos eles são compostos por um identificador único que permite relacionar o serviço em causa com o grupo ao qual o utilizador está inserido, concedendo desta forma acesso, ou não, à utilização do recurso.

Houve a necessidade de em todos os serviços ter cuidado com a informação a que o utilizador pode aceder, desta forma apesar de ser possível qualquer aluno efetuar um pedido de exercícios de disciplinas às quais não se encontra inscrito, irá receber como resposta um erro de falta de privilégios. Assim sendo, existem três níveis de segurança pelos quais todos os pedidos efetuados têm de passar, o primeiro é relacionado com o grupo de utilizadores, o segundo com as regras específicas por cada utilizador (no caso de existirem) e finalmente a implementação interna de cada serviço que monitoriza constantemente a origem dos pedidos e o contexto de informação a consultar, avaliando se existe ou não privilégios do utilizador a esta.

Ainda relativamente aos setenta e cinco serviços, todos eles foram implementados de forma a responderem nos formatos JavaScript Object Notation (JSON) ou XML. A diferenciação do formato que se pretende receber os dados é passada no endereço do pedido acrescentando o sufixo “json” ou “xml”. Por omissão todos os pedidos são respondidos no formato XML. Os dados devolvidos pelos serviços são objetos criados exclusivamente com o propósito de serem usados para transmissão de informação, isto é, foram criados modelos de dados que são traduzidos pelo Hibernate para tabelas de base de dados e que poderiam também ser usados para retorno de informação dos serviços, desta forma estaria-se a expor diretamente o modelo de base de dados utilizado. Seguidamente é apresentado o pedido a um serviço e respetiva informação do utilizador (independentemente se é professor ou aluno) com o número mecanográfico *12345* no formato JSON:

`http://saal.aws.atnog.av.it.pt/v1/curricula/userByNMec.json?nmec=12345`

```
{
  "uu": "utilizador@ua.pt",
  "nmec": 12345,
  "name": "Utilizador_de_Teste",
  "course": {
    "courseCode": 1000,
    "course": "Disciplina_Isolada",
    "acronym": "DI",
    "nStudents": 0,
    "enabled": true
  },
  "role": "Admin"
}
```

É também visível no endereço eletrónico a divisão deste em vários elementos, entre os quais o “v1” e o “curricula”. O primeiro exprime a versão do recurso a consultar, desta forma é possível criar novas versões destes serviços e manter a compatibilidade com aplicações antigas que não tenham acompanhado a evolução deste sistema. Relativamente ao segundo elemento, “curricula”, representa o tipo de recursos a consultar, neste caso será um serviço relacionado com a gestão curricular. De forma a garantir um registo de acesso de todos os utilizadores, foi implementado um sistema de registo que permite guardar todos os acessos efetuados bem como o respetivo utilizador na data e hora específicas de consulta de um qualquer serviço.

Como já referido nos Casos de Utilização 3.5.2, foi implementado um serviço exclusivamente para importação de exercícios de edições anteriores. Esta funcionalidade faz importação dos vários módulos existentes da edição selecionada bem como respetivos exercícios. Para cada um dos exercícios define também as linguagens de programação que este aceita, bem como todos os ficheiros auxiliares que possam existir e os testes que validarão as submissões dos alunos. Nenhuma informação de submissões anteriores será importada.

Durante o desenvolvimento dos vários serviços foram tidos em consideração vários aspetos relacionados com o desempenho destes, chegando-se mesmo a efetuar várias correções durante a fase de testes, mitigando o tempo de resposta de alguns destes. Este problema aconteceu essencialmente devido às configurações usadas no Hibernate e após alguns ajustes, afinações e refinamento nas consultas efetuadas se conseguiu diminuir estes valores para pequenas partes de segundo. Mas não só com a base de dados foram tidos cuidados especiais, por exemplo, na consulta de informação estatística

de uma edição, é criado dinamicamente um ficheiro no formato CSV com a informação de todos os alunos categorizados por turmas e de todos os exercícios definidos bem como números de submissões efetuadas corretamente e total de número de submissões. A construção deste ficheiro é um processo de extrema complexidade computacional e que pode chegar a demorar vários segundos (dependendo do número de alunos, exercícios e submissões existentes), desta forma usaram-se objetos da linguagem Java desenvolvidos para a otimização de determinadas tarefas, tal como a concatenação de texto[31].

Os serviços de criação de turmas ou edições obrigam à associação de pelo menos um professor para a sua criação. Isto pode levantar problemas no momento de se criar uma nova edição de uma disciplina ou no momento de criar as turmas de uma edição se o professor que se pretende associar ainda não tiver efetuado uma primeira autenticação no sistema. Nestas condições não existirá informações relativamente a este pelo que não é possível efetuar uma pesquisa dos professores pelo seu e-mail, mais propriamente Utilizador Universal. Para contornar o problema foi implementada a possibilidade de serem adicionadas as edições ou turmas associando o professor pelo seu número mecanográfico, permitindo que no momento que o professor efetuar a primeira autenticação válida no sistema fique automaticamente associado às edições ou turmas nas quais tinha sido referenciado previamente.

Por cada nova edição criada é necessário efetuar a definição das várias turmas e respetivos professores. À criação de turmas prende-se a necessidade de associar alunos. Como existem disciplinas que em cada nova edição contam com um número substancial de alunos, é importante que exista uma funcionalidade de importação de alunos. O procedimento de associação de alunos às turmas ocorre na seguinte ordem:

1. O Regente obtém a lista de todos os alunos do PACO e adiciona-os a uma turma denominada de *Pool* à qual pertencem todos os alunos que não têm uma turma definida.
2. Cada um dos professores obtém também do PACO a lista de alunos das suas turmas e importa os alunos da *Pool* para a sua turma.

A lista de alunos por edição e por turma são exportados pelo PACO num ficheiro formatado na forma CSV. A motivação destes dois passos para associação dos alunos às turmas relaciona-se com o facto de apenas os regentes terem autorização para adicionar alunos às edições. Após os alunos estarem associados, é então possível aos professores moverem esses alunos da turma *Pool* para as suas turmas com uso da lista de alunos que podem exportar no PACO, minimizando a tarefa árdua de mover os vários alunos individualmente (opção que também está implementada). Apesar de na interface web os professores selecionarem um ficheiro que contem a lista dos alunos, a sua inserção passa por efetuar tantos pedidos ao serviço de associação de alunos quanto o número de alunos que estejam presentes no ficheiro, desta forma o serviço não necessita de ter conhecimento algum da estrutura dos ficheiros usados. No caso de um professor pretender remover um aluno da sua turma, este é automaticamente movido para a *Pool*, apenas os regentes têm permissões para desassociar um aluno de uma edição.

Relativamente à submissão de ficheiros, sejam eles da resolução de exercícios e submetidos pelos alunos ou ficheiros auxiliares à descrição dos exercícios ou dos testes dos exercícios, todos eles são alocados diretamente no sistema de ficheiros do disco rígido e categorizados pelo seu tipo (submissão, exercício ou teste), seguidamente sub categorizados mediante o identificador da categoria anterior.

A PEPR será apenas detalhada no capítulo 3.6, ainda assim, foi já referido durante a Arquitetura Proposta 3.1 que apesar desta possuir uma interface pública de comunicação REST para troca de mensagens, toda a interação será feita pelo SAAL. Esta opção de implementação está diretamente relacionada com a possibilidade de tornar este um sistema escalável ao permitir múltiplas instanciações da PEPR em diferentes máquinas e assim permitir controlar todos os pedidos de execução de código

que chegam à plataforma. Segundo a teoria das probabilidades [32], a escolha de um valor aleatório de um conjunto de valores pré estabelecido terá uma probabilidade igual de acontecer entre os vários valores. O sistema de escalabilidade implementado baseia-se nesta teoria e na escolha aleatória de um de vários endereços onde a PEPR se encontre em execução. Suponha-se que existem dois sistemas a executar a PEPR, será necessário configurar no SAAL a localização de ambas as máquinas e sempre que chegar um pedido para execução de algo, será escolhido uma das duas máquinas para executar o pedido. Apesar de muito simples, este sistema pode prevenir tempos de espera longos em períodos de muita afluência dos alunos, como em tempo de aulas ou mesmo durante um exame.

Uma aplicação desta envergadura necessita obviamente de um sistema de configurações. Uma das possibilidades passaria pela utilização da base de dados para persistência das várias opções configuráveis, apesar desta forma ser talvez mais segura optou-se pelo uso de um ficheiro com extensão “.properties” que guarda pares de chave e valor para cada propriedade. Algumas das configurações que se podem encontrar no ficheiro são os endereços de destino do IdP, o respetivo par de chaves assimétricas usadas na autenticação e até a localização das várias instâncias em execução da PEPR. Também a configuração das credenciais de acesso à base de dados é persistida num ficheiro, mas este no formato de XML, por ser este o formato lido pelo Hibernate. O uso de ficheiros como alternativa às bases de dados torna mais fácil a alteração de opções por apenas ser necessário um simples editor de texto.

3.5.7 Aplicação WEB

Como referido no início deste capítulo, os serviços por si só não resolviam o problema apresentado neste trabalho, desta forma foi também necessário o desenvolvimento de uma aplicação capaz de usar esses serviços e que pudesse ser acedida por todos os utilizadores que dela pretendam fazer uso. Para tal optou-se por uma aplicação web que é facilmente acedida através de qualquer dispositivo que tenha um browser instalado, inclusive nos dispositivos móveis (smartphones).

O desenvolvimento do web site envolveu o uso de várias ferramentas diferentes com diferentes propósitos. Para o desenvolvimento de toda a parte gráfica foi usado o Twitter Bootstrap [20] que, como já apresentado na secção 3.4.8, permite criar uma aplicação visualmente agradável de forma muito rápida e relativamente simples. Após a definição das várias páginas e formas de navegação entre estas, é ainda necessário implementar as funcionalidades que vão permitir controlar as ações do utilizador, para tal usaram-se as frameworks escritas em Javascript Backbone.js e jQuery 3.4.7. Foi ainda necessário recorrer a outras ferramentas Javascript como complemento a estas, o Alertify.js foi usado como sistema de notificação capaz de dar feedback ao utilizador das suas ações e ainda foi usada outra capaz de criar um editor de texto What You See Is What You Get (WYSIWYG)⁸. Ainda de referir que este web site foi desenvolvido de acordo com a arquitetura *Single Page Application* apresentada em 3.4.6.

Um requisito necessário à implementação desta aplicação web foi o facto de existirem alunos do programa *Erasmus* na UA que poderão fazer uso desta ferramenta. Desta forma a aplicação tem suporte à língua internacional Inglês, dando a possibilidade de o utilizador escolher em que linguagem pretende que lhe seja apresentado a informação.

Para os utilizadores que tenham efetuado um registo no web site Gravatar [33] com o seu e-mail da UA, terão a possibilidade de ver a foto definida no seu perfil também no SAAL.

⁸Editor de texto capaz de apresentar a edição do texto num formato muito próximo da forma como este será impresso.

Seguidamente serão apresentadas imagens que representam capturas de ecrãs de algumas vistas mais relevantes do web site.

Página de entrada

Na Figura 3.8 é apresentada a página inicial, esta é a página para a qual todos os utilizadores são reencaminhados após efetuarem autenticação válida. Apesar de ser apenas uma página de transição para o utilizador, é nesta que é apresentada a possibilidade de seleção do curso, visto esta informação não ser fornecida pelo IdP.

The screenshot shows the SAAL system interface. At the top, there's a header with the system name, language tabs (EN, PT), and a Logout button. On the left, a sidebar displays the user's name, profile picture, and navigation links (Home, Curricula, PEPR). The main content area shows a 'User Information' form with fields for Universal User, Name, N° Mec, and Course. Below the form is a 'Save' button. To the right of the form, there are three boxes showing 'MSI' (Course), '0' (N° Subjects), and '16:06 27-05-2013' (Last Login). At the bottom, a footer contains copyright information.

Sistema de Apoio à Aprendizagem de Linguagens

EN PT Logout

Pedro Miguel de Oliveira Estima
Admin
Paco
Moodle

Home

Curricula

PEPR

Register on
http://gravatar.com to see
your gravatar image on
SAAL website. Remember to
use your Universal User e-
mail during your
registration.

Home

MSI
Course

0
N° Subjects

16:06 27-05-2013
Last Login

User Information

Universal User: pedroestima@ua.pt

Name: Pedro Miguel de Oliveira Estima

N° Mec: 43602

Course: Mestrado em Sistema de Informação

Save

© 2013 Pedro Estima - Instituto de Telecomunicações - Universidade de Aveiro

Figura 3.8: Página de entrada

É possível ainda observar no canto superior direito a possibilidade de escolha das duas línguas disponíveis, o Inglês e o Português e ainda a opção de terminar sessão. Logo abaixo do lado esquerdo é apresentado o nome do utilizador, seguida da sua imagem de utilizador, se registado no web site Gravatar [33]. Do lado esquerdo são apresentados os menus laterais de navegação e que permitem a consulta de disciplinas, não visíveis nesta imagem, bem como a consulta da página de gestão do sistema. No centro será então apresentada toda a informação da página, são apresentados três colunas com informação do utilizador e logo abaixo a sua informação pessoal.

Vista de Administrador

Os Administradores do sistema têm uma secção própria onde podem efetuar a gestão das linguagens de programação, dos cursos e das disciplinas e bem como da respetiva gestão que engloba as edições, ou seja, podem efetuar qualquer uma das ações definidas no diagrama de casos de utilização 3.5.2 para

os utilizadores Professor e Regente. A Figura 3.9 mostra a página de disciplinas e respetivas edições da disciplina Programação II.

The screenshot shows the 'Sistema de Apoio à Aprendizagem de Linguagens' (SAAL) administrator interface. The user is logged in as Pedro Miguel de Oliveira Estima. The main menu includes 'Curricula', 'Home', and 'PEPR'. The 'Curricula' section is active, showing 'Courses Information' with tabs for 'Subjects', 'Courses', and 'Languages'. A 'New Subject' button is present. Below, a table lists subjects and their editions. The subject 'Programação II' (ID 43257) is selected, showing its details in a sub-table.

#	Subject	Acronym	Nº Editions	State	
16846	Programação III	P3	0	Enabled	
43257	Programação II	P2	1	Enabled	

Year	Semester	Director	Students	Approvals	Avg Grade	Status	
2013	2	1181 - Miguel Augusto Mendes Oliveira e Silva 3781 - Diogo Nuno Pereira Gomes	44	0 (0%)	0	Enabled	
46816		Programação Concurrente Orientada por Objetos		0		Enabled	

Figura 3.9: Vista de Administrador

Nesta imagem são visíveis três disciplinas das quais a *Programação II* encontra-se num estado aberto que permite a consulta das várias edições. Nesta caso apenas existe uma edição da disciplina selecionada com dois regentes e 44 alunos. É também ainda visível o conjunto de mais dois separadores que permitem a gestão dos cursos e das linguagens de programação.

Listagem de exercícios

A Figura 3.10 exibe a vista do professor ao selecionar uma edição. Neste caso pode-se observar que a imagem se refere à disciplina de Programação II e existem três separadores de informação; o primeiro para gestão de turmas, o segundo, e que se encontra selecionado, apresenta os módulos e respetivos exercícios e por fim o ranking de todos os alunos inscritos.

Sistema de Apoio à Aprendizagem de Linguagens

ENPTLogout

Pedro Miguel de Oliveira Estima

AdminPacoMoodle

Home

Curricula

PEPR

P2 (2º / 2013)

Register on <http://gravatar.com> to see your gravatar image on SAAL website. Remember to use your Universal User e-mail during your

Programação II

Classes

Exercises

Ranking

New Module

Import Exercises

Statistic Info

Module	Weight	Nº Exercises	State
APF	-	3	Enabled

Exercise	Begin Date	End Date	Rate Method	Grade	Enabled
E1	23-05-2013 00h:00m	-	#####	-	Enabled
E2	23-05-2013 00h:00m	-	#####	-	Enabled
E3	23-05-2013 00h:00m	23-05-2013 00h:00m	#####	-	Enabled

Figura 3.10: Listagem de exercícios

Nesta figura pode-se constatar a presença de uma disciplina no menu lateral esquerdo relativa ao segundo semestre do ano de 2013. O conteúdo apresentado na página é também referente a esta disciplina onde são apresentados os módulos existentes, neste caso denominado por APF e respetivos três exercícios. A seleção de um módulo permite abrir o seu conteúdo para apresentação da lista de exercícios.

Edição de Exercício

A página da edição de exercícios pode ser consultada na Figura 3.11, esta é útil por permitir a criação e edição de exercícios e ao mesmo tempo definir e consultar toda a informação relativa a estes.

Sistema de Apoio à Aprendizagem de Linguagens

ENPTLogout

Pedro Miguel de Oliveira Estima

AdminPacoMoodle

Home

Curricula

PEPR

P2 (2º / 2013)

Register on <http://gravatar.com> to see your gravatar image on SAAL website. Remember to use your Universal User e-mail during your registration.

Programação II

Edition - 2013 / 2º Semester

Module: APF

7

Nº Commits

0 (0.00%)

Nº Success

No

Mandatory

Yes

Individual

ExerciseTestsFilesCommits

Module: APF

Exercise: E1

PT Description:

File Edit Insert View Format Table Tools

Formats

B I

Exercício E1

Crie um módulo LeakyQueue, baseado na estrutura de dados fila, de forma a que o programa ProgX funcione devidamente1.

Uma fila "rota" (leaky queue) é uma estrutura de dados baseada numa fila, mas em que só ficam armazenados, no máximo, os últimos N números inseridos. Quando a fila está preenchida (N elementos) a inserção de um novo número implica a saída do primeiro (que deixa de existir).

p

EN Description:

File Edit Insert View Format Table Tools

Formats

B I

p

Programming Langs:

☒ Java

Begin Date:

23-5-2013

0h

0m

X

Leave date empty if you do not need a begin date

End Date:

0h

0m

X

Leave date empty if you do not need an end date

Mandatory: ☐

Individual: ☒

Enabled: ☒

Save

Figura 3.11: Edição de exercício

Como se pode constatar, é apresentado no início da página alguma informação estatística que poderá ser útil ao professor, como o número total de submissões ou o número de submissões aceites com valores em percentagem. Existem duas descrições para o exercício, uma será em português e é de preenchimento obrigatório ao passo que a segunda será em inglês e não existe a necessidade de a preencher. Será apresentada uma lista de linguagens de programação disponíveis, e o intervalo de datas durante o qual é possível aos alunos efetuarem a submissão dos exercícios. No final é ainda possível definir se o exercício é de resolução obrigatória, se é de submissão individual ou poderá ser em grupo e ainda se está disponível ou não ao aluno.

Nesta página encontram-se ainda os separadores de testes, onde são definidos os testes do exercício

em causa, os ficheiros auxiliares ao exercício e por fim o separador das submissões. Este último será apresentado pela Figura seguinte 3.12.

Detalhes da execução de código

A Figura 3.12 apresenta a lista de submissões e respetivos resultados de todos os alunos inscritos por edição.

Home

Curricula

PEPR

P2 (2º / 2013)

Register on
http://gravatar.com to see
your gravatar image on
SAAL website. Remember to
use your Universal User e-
mail during your
registration.

Module: APF

7

Nº Commits

0 (0.00%)

NºSuccess

No

Mandatory

Yes

Individual

Exercise

Tests

Files

Commits

Filters

Filter:

State: All

Filter

#	Group	Date	File	Score	Language	State
7	Danilo Alexandre Estima Saraiva	23-05-2013 18h:33m	ProgX.zip	-	java	With Errors
6	Danilo Alexandre Estima Saraiva	23-05-2013 17h:48m	ProgX.zip	-	java	With Errors
5	Danilo Alexandre Estima Saraiva	23-05-2013 17h:47m	ProgX.zip	-	java	With Errors
4	Danilo Alexandre Estima Saraiva	23-05-2013 17h:46m	ProgX.jar	-	java	With Errors
3	Danilo Alexandre Estima Saraiva	23-05-2013 17h:44m	ProgX.zip	-	java	With Errors
2	Danilo Alexandre Estima Saraiva	23-05-2013 17h:42m	ProgX.zip	-	java	With Errors
1	Danilo Alexandre Estima Saraiva	23-05-2013 17h:40m	ProgX.zip	-	java	With Errors

7 - Danilo Alexandre Estima Saraiva

Test1

Test2

Test3

Test4

Your Output

```
$ javac ProgX.java
$ java ProgX 1 2 3 4 5 6

i = 0 1.0 (Min = 0.0) «ENTER»
i = 1 1.0 2.0 (Min = 0.0) «ENTER»
i = 2 1.0 2.0 3.0 (Min = 0.0) «ENTER»
i = 3 2.0 3.0 4.0 (Min = 0.0) «ENTER»
i = 4 3.0 4.0 5.0 (Min = 0.0) «ENTER»
i = 5 4.0 5.0 6.0 (Min = 0.0) «ENTER»

Return Code: 0
```

Expected Output

```
i = 0 1.0 (Min = 1.0) «ENTER»
i = 1 1.0 2.0 (Min = 1.0) «ENTER»
i = 2 1.0 2.0 3.0 (Min = 1.0) «ENTER»
i = 3 2.0 3.0 4.0 (Min = 2.0) «ENTER»
i = 4 3.0 4.0 5.0 (Min = 3.0) «ENTER»
i = 5 4.0 5.0 6.0 (Min = 4.0) «ENTER»

Return Code: 0
```

Figura 3.12: Detalhes da execução de uma submissão

A informação está repartida em dois blocos distintos, no primeiro, e à semelhança da figura 3.11, é apresentada alguma informação estatística seguida da listagem de todas as submissões efetuadas.

Relativamente ao segundo bloco, este apenas é apresentado quando existir uma seleção de alguma submissão, ou seja, se o professor desejar consultar informação mais detalhada de determinada submissão, será apresentado este bloco com a informação relativa a todos os testes juntamente com os resultados da execução do código submetido pelo aluno.

O professor tem ainda a possibilidade de efetuar uma atualização da lista de submissões sempre que necessário possibilitando a consulta das submissões mais recentes. Esta opção é útil, sobretudo durante o período de aulas, e fornece a possibilidade de o professor consultar as submissões mais recentes sem ter de efetuar uma atualização completa da página. Esta funcionalidade poderia ser implementada de forma totalmente transparente para o professor, mas tal implicava o pedido completo da lista de submissões de forma recursiva ao servidor e em curtos intervalos de tempo. Por si só esta prática envolvia uma carga extra no servidor que foi evitada ao apresentar um botão que permite ao professor efetuar a atualização das submissões. Outro fator que contribuiu para esta implementação foi o facto que mesmo durante as aulas, o professor não se encontra constantemente em frente ao seu computador, pelo que não é necessário a aplicação se estar constantemente a atualizar.

Execução na PEPR

Todos os professores têm a possibilidade de criar exercícios e adicionar os respetivos testes. De forma a auxiliar o professor no preenchimento do resultado esperado para cada teste, existe uma página onde é possível ao professor executar código no sistema e ver qual o resultado devolvido, desta forma poderá criar os testes de acordo com a informação resultante desta execução. A Figura 3.13 mostra essa página com o resultado devolvido da execução de um pequeno exemplo de código submetido.

The screenshot displays the PEPR interface within a Moodle environment. The top navigation bar includes 'EN', 'PT', and 'Logout' buttons. The left sidebar identifies the user as 'Pedro Miguel de Oliveira Estima' with roles 'Admin', 'Paco', and 'Moodle'. The main content area is titled 'Plataforma de Execução PRotegida' and contains a 'Tests Execution' section. In this section, the 'Language' is set to 'Java'. The 'Action' field contains the commands: `javac -encoding UTF-8 $SUBMITTED_FILE.java` and `java -Dfile.encoding=UTF8 $SUBMITTED_FILE 10 5`. The 'Source File' is 'ex.java' and the 'Other File' is 'No file chosen'. A 'Run on PEPR' button is present. Below this, the 'Result' section shows 'resultado: 2' and 'Return Code: 1'. A green notification bar at the bottom right states 'The results of your submission are ready'.

Figura 3.13: Execução na PEPR

A lista das linguagens que o professor poderá escolher está dependente das linguagens de programação definidas pelo administrador do sistema e que a PEPR suporta. Já relativamente ao bloco onde é apresentado o resultado da execução, este é dinâmico e só é apresentado após chegada dos resultados desta. É também ainda possível observar nesta página uma notificação no canto inferior direito como forma de *feedback* dado ao utilizador, neste caso específico alertando para a atualização da página com o resultado da execução.

Vista do exercício do aluno

Por fim é apresentada a vista do exercício mas na visão do aluno. Além da descrição do exercício em si, é também possível consultar outras informações relevantes como os ficheiros auxiliares no caso de existirem, as datas entre as quais é possível submeter as resoluções deste exercício, informação acerca da obrigatoriedade da resolução do exercício e ainda a forma como este deve ser entregue, individualmente ou se poderá ser resolvido e entregue por grupos de alunos.

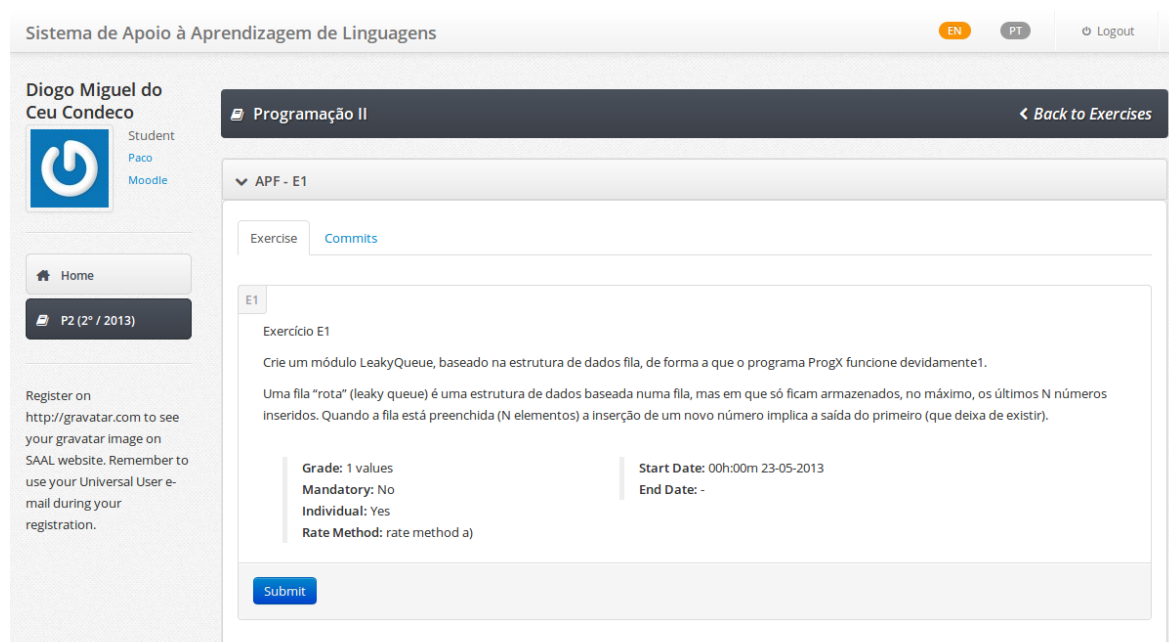


Figura 3.14: Página de exercício do aluno

Considerações finais

De forma a proporcionar uma interação fluída e o mais consistente entre as várias páginas juntamente com uma interface agradável, foram tidas em consideração as várias heurísticas de usabilidade de Jakob Nielsen [34]. Estas permitiram a criação de um web site de relativamente fácil navegação para os utilizadores que deste irão fazer uso durante a aprendizagem de linguagens de programação.

A Figura 3.15 apresenta uma possível vista da aplicação web em dispositivos móveis, *smartphones* e *tablets*. Nestes casos a visualização irá depender da resolução do dispositivo em questão e como é possível constatar, existem algumas diferenças no aspeto do web site em relação à vista num browser. Estas alterações ocorreram devido ao redimensionamento dos vários elementos da página de forma a serem renderizados para a resolução do dispositivo móvel. Tal acontece uma vez que o web site é

completamente dinâmico e capaz de se adaptar a todas as resoluções sem que para isso necessite de diferentes estilos consoante a resolução do dispositivo. Como já referido nas tecnologias usadas, estas características devem-se ao Twitter Bootstrap.

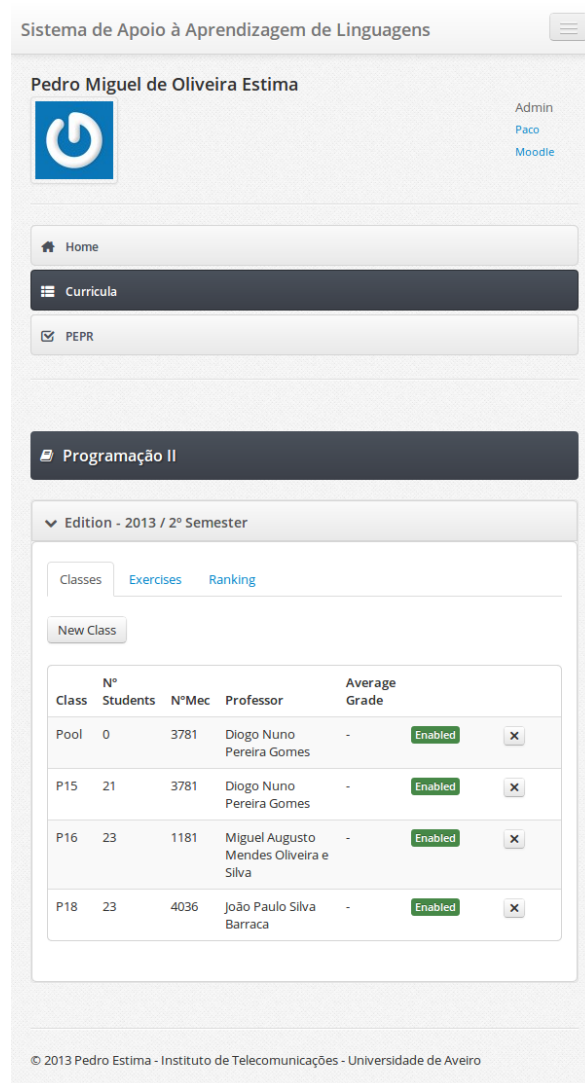


Figura 3.15: Versão em dispositivo móvel

Como é possível observar pela Figura 3.15, existiu uma modificação no topo da página, agora as diferentes línguas da aplicação web bem como a opção de terminar sessão encontram dentro do botão no canto superior direito. Também a informação do aluno passou a ocupar o espaço disponível todo e os menus laterais esquerdos mudaram a sua localização para cima do conteúdo da página.

3.6 PEPR

Esta secção é dedicada à contextualização da PEPR e serão apresentados os principais motivos que levaram ao seu desenvolvimento. Serão também apresentadas algumas formas de criação de um ambiente de execução protegido e qual a escolhida para este trabalho. Será também apresentado

como comunicar com a PEPR, a arquitetura interna desta e por fim algumas das suas características funcionais.

3.6.1 Contextualização

A PEPR é uma plataforma que permite a execução de código num ambiente protegido. No contexto deste trabalho, o foco principal passa pela execução do código submetido pelos alunos e respetiva avaliação do resultado com indicação de correto ou não. Visto ser necessariamente obrigatória a execução do código submetido pelos alunos, código este que pode vir acompanhado de vários tipos de ficheiros e os quais podem ter fins maliciosos, é estritamente necessário que esta execução aconteça num ambiente de tal forma protegido e isolado que não seja possível comprometer a segurança do SO nem de futuras execuções.

Como acontece no SAAL, a interação com a PEPR também acontece via web com pedidos REST e dados formatados exclusivamente segundo a notação JSON. Assim sendo, esta aplicação pode ser usada por terceiros em contextos independentes deste trabalho e poderão ser desenvolvidas interfaces gráficas que fazem uso da PEPR para execução do código, como concursos de programação que poderão adotar o uso desta plataforma para dar resposta às suas necessidades de execução de código.

Antes de qualquer execução efetuada pela plataforma existe a necessidade de efetuar uma preparação prévia do sistema. Esta preparação implica a criação de um ambiente virtual dedicado apenas para uma execução, denominado de *jail*. É também necessário efetuar a cópia dos ficheiros necessários à execução para esse ambiente, inicializar alguns *scripts* para a execução, executar e por fim recolher os resultados da execução. A inicialização dos referidos *scripts* é um passo intermédio para configuração do ambiente protegido, é através destes *scripts* que serão criadas algumas variáveis de ambiente dentro deste ambiente, que se comporta como um novo SO, e são eles também que permitem efetuar uma conversão de codificação dos ficheiros.

De uma forma resumida, esta plataforma permite adicionar uma camada de segurança e isolamento durante a execução de código, é ela também responsável por dar início à execução e efetuar a análise do resultado.

3.6.2 Criação do Ambiente Protegido

A criação de um ambiente protegido para execução de código justifica-se com a possibilidade dos utilizadores poderem tentar corromper o sistema e apoderar-se do mesmo. Com a utilização das *jails*, para execução de código, mesmo que uma destas fique comprometida, não irá comprometer de forma nenhuma futuras execuções nem outros sistemas que possam estar a ser executados no mesmo servidor ou na mesma rede. Esta estrutura de segurança que permite executar código num ambiente diferente do SO chama-se *Sandbox*, e existem várias formas de criação:

Chroot Jail este método cria uma sandbox com funcionalidades limitadas e previamente definidas dentro do sistema de ficheiros ao alterar a raiz do sistema de ficheiros para outra localização que desta forma cria um novo ambiente apenas com as funcionalidades adquiridas nessa nova localização. Este método é extremamente rápido e apesar de já existirem formas de quebrar a sua segurança, estas exigem determinados requisitos que normalmente não são atribuídos ao utilizador comum, como por exemplo o acesso *root* ao sistema.

HTML 5 uma forma de tornar os browsers mais seguros passa pelo uso de sistemas de sandbox para execução de *plugins* das várias páginas web. O exemplo mais comum e conhecido são as

animações em Flash Player⁹ ou os Applets em Java¹⁰. Com o aparecimento do HTML 5 esse problema ficou mais fácil de resolver ao colocar a possibilidade de importar conteúdos externos dentro das tags `<iframe>` com a opção de serem executadas dentro de uma sandbox. Desta forma é acrescentada mais uma camada de segurança à já existente dos browsers.

OpenVZ esta aplicação, apenas disponível para sistemas Linux, é capaz de criar contentores virtuais totalmente isolados com possibilidade de executar até outros sistemas operativos baseados também em Linux. Apesar de ser um sistema bastante seguro tem algumas desvantagens face ao *Chroot Jail* pois necessita de uma atualização especial no SO anfitrião e existe uma perda de performance que varia entre 1% e 3%.

KVM esta solução, também apenas para sistemas baseados em Linux, está muito perto do software tradicional de virtualização e necessita de alterações ao sistema de núcleo do SO onde estiver instalado. Ao contrário do *OpenVZ* agora já é possível a criação de máquinas virtuais de outros SOs. Em vez de criar um ambiente simulado, o KVM expõe o hardware ao vários sistemas instalados para que estes façam uso quase direto do mesmo.

SELinux é um mecanismo de atribuição de políticas de controlo de acessos que permite definir a que recursos as aplicações e os utilizadores podem aceder. O SELinux não se trata de uma aplicação mas sim de um conjunto de atualizações ao núcleo do Linux, que a maioria das distribuições já trazem, e que desta forma não necessita da instalação de aplicações extra. A sua configuração requer pessoal especializado e com muitos conhecimentos de sistemas Linux.

LXC é uma aplicação descrita como sendo um Chroot com funcionalidades avançadas por oferece o mesmo que o *Chroot* mas com um leque de opções adicionais. A sua segurança é obtida devido ao uso de políticas de controlo de acesso ao nível dos grupos de utilizadores ao uso de *namespaces* para definição de ambientes de novos ambientes de execução.

AppArmor é também um módulo para o núcleo do Linux que permite a associação de aplicações a perfis de segurança definidos pelo administrador do sistema. É visto como uma alternativa ao *SELinux* por conter algumas melhorias ao nível interno e forma de funcionamento. Este módulo foi adotado pelo sistema Ubuntu [35] e amplamente desenvolvido desde a sua adoção.

Das várias possibilidades para criação de uma *sandbox* acima referidas foi necessário escolher uma que melhor se enquadrasse no contexto e na resolução dos problemas deste trabalho. A criação de máquinas virtuais é sem dúvida o método mais seguro a seguir, mas ao mesmo tempo é também o mais dispendioso ao nível de recursos usados, tempo de preparação do ambiente, etc. Por sua vez, o *Chroot* é extremamente rápido apesar de poder não ser completamente fiável como acontece com as máquinas virtuais, ainda assim permite a criação, configuração e a destruição de um ambiente de execução numa pequena parte de segundo. Uma vez que irá sempre existir um compromisso de segurança - recursos utilizados (tempo, memória, CPU, etc.) a escolha incidiu sobre o *Chroot Jail*.

⁹Plataforma multimédia normalmente usada para reprodução de músicas ou vídeos, animações vetoriais e jogos. Estes são executadas no browser dentro do Adobe Flash Player.

¹⁰Aplicação desenvolvida em Java e em execução num processo diferente do browser dentro da JVM.

A escolha deste método para a criação da sandbox justifica-se essencialmente pelo tempo que estas demoram a ser criadas e funcionalidades obtidas. Não se espera que a aplicação desenvolvida tenha um uso mais ou menos constante ao longo do tempo pelos alunos, em vez disso esta será essencialmente usada em períodos de tempo mais ou menos conhecidos. Sabe-se pela experiência dos professores que a aplicação será quase exclusivamente usada em vésperas de períodos de entregas obrigatórias de trabalhos e durante as aulas ou testes, desta forma espera-se uma utilização algo abusiva durante estes períodos e os tempos que a PEPR necessitará para dar uma resposta terão de ser os mais baixos possíveis de forma a não criar listas de espera para execução de trabalhos.

Pela Figura 3.16 observa-se uma estrutura de ficheiros de como configurar as *jails*. Na pasta raiz do SO foi criada a pasta “jails” onde serão criadas e configuradas três diferentes *jails*. A grande vantagem deste método é que se pode usar o próprio sistema de ficheiros para ajudar a criar uma separação lógica de acordo com os requisitos pretendidos, ou seja, uma máquina que necessite de executar código Java provavelmente não necessita de executar código Python. É ainda possível efetuar subdivisões, por exemplo, existem várias versões de Python com requisitos e bibliotecas diferentes pelo que se podem configurar todas dentro da pasta respetiva do Python. De forma a dar a conhecer à PEPR quais as linguagens suportadas, deverá ser editado um ficheiro de configurações que contem toda a informação necessária à execução juntamente com as localizações de onde se encontram as várias *Jails*, tempos máximos de execução por jail, etc.

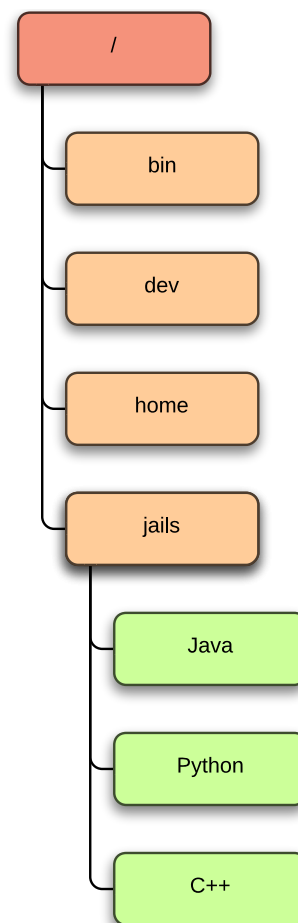


Figura 3.16: Sistema de ficheiros

A criação do ambiente virtual para a execução de código acontece em vários passos e existem algumas imposições que são necessárias satisfazer. A execução de alguns sistemas operativos baseados em Linux necessitam do diretório `/proc`, esta não é uma localização física no sistema de ficheiros mas sim uma localização virtual relacionada com os processos em execução e informação do SO, como carga do processador, memória, etc. É então necessário usar o comando *mount* para criar esta localização virtual para o novo ambiente virtual. Este comando tem a funcionalidade de permitir montar diretórios ou unidades de armazenamento externas, como unidades USB ou drives de CD, e associa-los a uma localização no sistema de ficheiros. É ainda necessário usar este comando para associar a localização base do sistema operativo previamente configurado na localização desejada onde as execuções irão acontecer. Neste caso o comando vai também fazer uso do sistema de ficheiros AuFS¹¹ que desta forma vai permitir que toda e qualquer alteração a ocorrer no sistema de ficheiros não seja refletida na localização base do SO mas que sejam mantidas noutra localização.

Uma vez preparados o sistema de ficheiros virtual, é necessário proceder à cópia de todos os ficheiros necessários à execução para a localização onde esta irá ocorrer. Após a cópia é então necessário

¹¹Sistema de ficheiros que permite a unificação de várias diretorias numa única localização.

efetuar a mudança de contexto para dentro do ambiente de execução. A alteração do contexto para a localização no sistema de ficheiros onde irá acontecer a execução é realizada pelo comando *chroot* que ao mesmo tempo permite definir uma ação a executar.

3.6.3 Arquitetura

Ao nível arquitetural, a aplicação divide-se em três blocos estruturais apresentados pela Figura 3.17. À parte destes blocos existem também ficheiros necessários ao funcionamento da PEPR e que também são apresentados.

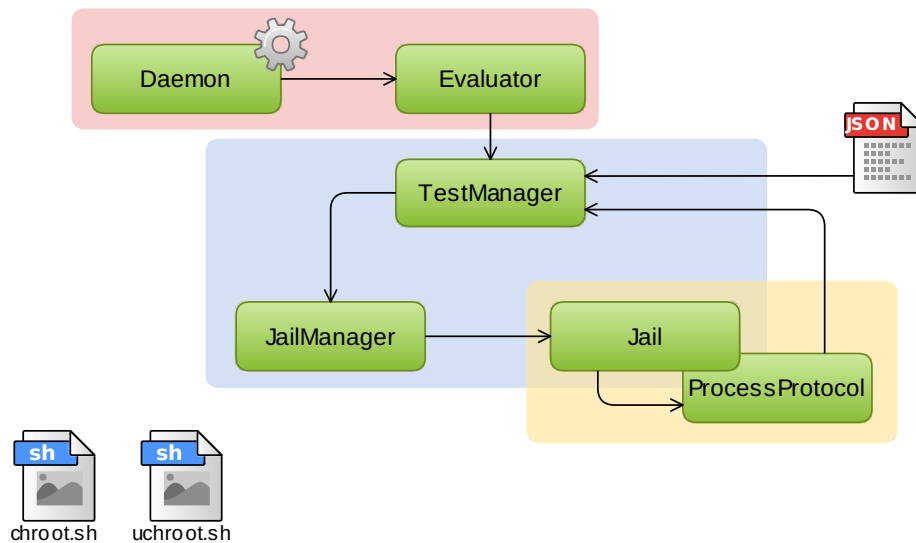


Figura 3.17: Arquitetura interna da PEPR

De acordo com o diagrama arquitetural e de cima para baixo, o primeiro dbloco é responsável pelo tratamento de novos pedidos para execução de código e é constituído por duas classes. A classe *Daemon* está constantemente à escuta da chegada desses novos pedidos REST e cria uma nova instância da classe *Evaluator* por cada nova chegada. Esta tem como principal funcionalidade a recolha de cookies do pedido efetuado bem como a extração dos dados recebidos no pedido e que contêm toda a informação necessária à execução.

O segundo bloco é dedicado à gestão dos testes e *jails* as classes *TestManager*, *JailManager* e *Jail*. A primeira recebe os dados extraídos anteriormente e prepara os vários testes para o exercício fornecido. Seguidamente por cada um dos testes será criada uma *jail* no bloco *JailManager* que será inicializada com o código fonte, ficheiros necessários à execução, preparação dos scripts de inicialização, etc. O terceiro bloco está sobreposto com o segundo na classe *Jail* devido à dependência que existe entre esta e a classe *ProcessProtocol*. A dependência ocorre apenas durante o intervalo de tempo da execução do código submetido e é nesta classe, *ProcessProtocol*, que ocorre a avaliação do resultado da execução do código fonte que será enviado para a *TestManager* que responderá ao SAAL e tratará de arquivar as várias *jails* criadas .

Existem ainda três ficheiros, um deles responsável por fornecer a configuração da plataforma e que se encontra em notação JSON e será passado em todos os pedidos ao *TestManager*. Os outros dois ficheiros são responsáveis por iniciar (*chroot.sh*) e terminar (*uchroot.sh*) o processo de ambiente

protegido. Seguidamente será apresentado um exemplo do ficheiro de configuração e respectivos parâmetros disponíveis:

```
{
  "spawnPath": "/opt/playground",
  "archivePath": "/opt/archives",
  "initScript": "/home/user/init.sh",
  "jails": [
    {
      "name": "Java-openjdk",
      "basePath": "/jails/Java/openjdk",
      "runPath": "/home/user",
      "user": "user",
      "executionTimeout": 10
    },
    {
      "name": "Java",
      "basePath": "/jails/Java/java_oracle",
      "runPath": "/home/user",
      "user": "user",
      "executionTimeout": 10
    }
  ]
}
```

spawnPath localização no sistema de ficheiros de onde serão criadas os vários ambientes virtuais para execução de código

archivePath localização no sistema de ficheiros onde serão mantidos os arquivos criados após destruição das várias instâncias

initScrip localização no ambiente virtual criado do script que irá dar início à execução do código submetido e onde se encontra a ação definida pelo professor

jails conjunto das várias configurações disponíveis para criação de ambientes virtuais, no exemplo acima são apresentadas duas configurações diferentes para Java

name nome da jail

basePath localização no sistema de ficheiros do sistema base pré configurado

runPath localização dentro do ambiente virtual para onde serão copiados os ficheiros submetidos e onde irá ocorrer a execução

user nome do utilizador dentro do ambiente virtual

executionTimeout tempo de execução máximo permitido para cada execução

3.6.4 Comunicação

A comunicação com a PEPR acontece com a troca de mensagens formatadas em JSON como exemplificado pelo seguinte excerto de código. A informação apresentada representa um novo pedido para uma nova execução.

Foi necessário delinear como seriam transmitidos à plataforma os vários testes a executar por cada submissão. Poderia ser efetuado um pedido de execução por cada teste existente ou seriam todos enviados num único pedido. Ambas as práticas têm vantagens e desvantagens pelo que a primeira obrigaria a um maior número de pedidos bem como mais informação trocada no total de todos os testes, pois por cada teste seria necessário enviar também o código fonte submetido pelo aluno. A segunda prática impõe a passagem de todos os testes de uma só vez e por isso uma grande quantidade de informação a ser trocada num único pedido, mas desta forma a PEPR ficaria munida para executar todos os testes com um só pedido. Optou-se pela segunda forma por permitir que apenas com um único pedido a plataforma fique munida de toda a informação necessária às várias execuções e minimizar o número de pedidos efetuados.

```
{
  "commit_id": 1,
  "exercise_id": 5,
  "callback": "http://localhost:8080/saal/v1/processing/processResult",
  "language": "Java",
  "about": {
    "name": "Pedro Estima",
    "nmec": "12345",
    "uu": "estima@ua.pt"
  },
  "source": {
    "value": "Ficheiro em Base64",
    "filename": "ex.java"
  },
  "tests": [
    {
      "files": [
        {
          "value": "Ficheiro em Base64",
          "filename": "file2"
        }
      ],
      "action": "javac exercicio.java; java exercicio argumento1 argumento2",
      "blocking": "true",
      "returnCode": 5,
      "expectedOutput": "compile02",
      "executionOrder": 3,
      "id": 2
    },
    {
      ...
    }
  ]
}
```

Apesar desta plataforma ter sido implementada de forma genérica não estando por isso exclusivamente dependente do SAAL, alguns dos termos utilizados foram baseados no contexto do trabalho desenvolvido, essa é a razão de existirem nomes como *commit_id* ou *exercise_id*. Seguidamente serão

apresentados todos os campos presentes e clarificadas as suas funcionalidades bem como a obrigatoriedade destes.

commit_id todas as execuções são arquivadas e este valor é usado como identificador único no nome do arquivo, serve também para responder ao SAAL a qual submissão corresponde a execução em causa.

exercise_id este campo é opcional e cuja única funcionalidade passa pela categorização dos arquivos em pastas de acordo com o exercício.

callback é o endereço eletrónico para o qual a PEPR deve enviar o resultado de todas as execuções efetuadas dos testes fornecidos.

language define qual o ambiente protegido que deve ser utilizado e que deve estar previamente configurado para suportar essa linguagem de programação.

about é um campo opcional que contém informação específica fornecida pelo SAAL e que única funcionalidade passa pelo arquivamento, permitindo aos professores a consulta de informação das execuções dos seus alunos se assim for necessário. Este campo é definido pelo nome do aluno, o seu número mecanográfico e e-mail.

source conteúdo do código fonte submetido pelo aluno. Este campo é definido pelo nome do ficheiro e a codificação do seu binário em Base64.

tests é a definição dos vários testes que têm de ser executados. Cada um destes testes é composto por vários campos: **files** é um campo opcional que representa uma lista de ficheiros extra e específicos à execução de cada teste, à semelhança do campo **source** também este é composto pelo nome do ficheiro e pela codificação do seu binário em Base64. O campo **action** é um comando em *bash script* que define o teste, este tanto poderá ser uma compilação seguido de uma execução como apenas uma contagem das linhas do código fonte. **blocking** é o campo que referencia o teste como sendo ou não bloqueante ¹². **returnCode** define qual o código de retorno com que a execução deve terminar. O campo **expectedOutput** apresenta o resultado que se espera que a execução devolva, se forem diferentes o teste é considerado errado, tal como comportamento acontece também com o **returnCode**. O campo **executionOrder** indica a ordem pela qual os testes devem ser ordenados para execução e finalmente o **id** representa o identificador do teste em questão, apesar de ser obrigatório pode conter qualquer valor visto que também só é usado para referência no SAAL.

3.6.5 Características Funcionais

O desenvolvimento de uma aplicação deste género envolve que esta seja munida de determinadas funcionalidades que permitam a execução de um grande leque de linguagens de programação, não estando assim apenas restrito a determinados detalhes de determinadas linguagens. Uma característica implementada e que é de alguma forma muito relevante prende-se com as configurações da plataforma, antes de começar qualquer processo para executar algo é sempre lido o ficheiro de configurações, como já referido no formato JSON. Desta forma é possível alterar as configurações a meio da execução sem necessidade de voltar a iniciar novamente a plataforma.

¹²Durante a execução de um teste bloqueante, se o resultado da execução não for o esperado, os seguintes testes serão ignorados e não irão ser executados.

Para decretar a validade de determinada execução, foi implementada uma forma de análise cega ao resultado obtido da execução. Quer isto dizer que, no pedido de uma execução está presente um campo de preenchimento obrigatório que define qual o resultado esperado por esta. No caso de se obter um resultado diferente do esperado, então o exercício submetido não é considerado como corretamente implementado pelo aluno.

Algumas das características funcionais implementadas na PEPR passam pela conversão da codificação de ficheiros de texto e de código fonte, isto porque diferentes alunos usam diferentes SOs e diferentes editores de texto que influenciam a codificação dos ficheiros. Desta forma, todos os ficheiros que não respeitem a formatação UTF-8 serão automaticamente codificados para esta, tentando assim minimizar a causa de problemas durante os processos de compilação ou execução.

Como apresentado na Arquitetura 3.6.3 desta plataforma, na configuração das *jails*, existe um parâmetro responsável por definir o nome do utilizador com o qual as ações irão ser executadas. Desta forma é criada mais uma camada de segurança neste sistema ao definir restrições de execução e acesso a este utilizador, e como só o administrador do sistema pode alterar este utilizador, os alunos deixam de ter privilégios a funcionalidades que poderiam de alguma forma colocar em causa a segurança do sistema.

A PEPR, como tem sido referido, é apenas uma plataforma que cria ambientes protegidos para execução de código, como tal, a plataforma terá de estar em execução num SO baseado em Linux com capacidades de execução da linguagem de programação Python 2.7 [36]. A dependência do Linux deve-se ao facto de apenas este tipo de SO ter uma funcionalidade necessária para a criação dos ambientes de execução protegidos. A criação destes ambientes depende da instanciação de um SO por cada nova execução, isto é, existe a necessidade de instalar e configurar um SO numa pasta no sistema de ficheiros. A configuração passa pela instalação dos compiladores necessários à execução de código e no caso do Java, pela instalação da JVM. Como tal processo seria extremamente moroso e pesado ao nível de processamento e tempo necessário, fez-se uso de outra funcionalidade presente nos sistemas Linux, esta funcionalidade permite que o processo de instalação e configuração de um novo SO seja feita apenas uma vez e que este possa ser usado em modo de leitura quantas vezes necessárias, ou seja, após ter o SO completamente configurado, este pode ser usado em todas as execuções de código submetidas pelos utilizadores e qualquer alteração que aconteça ao nível do sistema de ficheiros não será refletida nesse SO base, mas será mantida numa outra localização denominada por *scratch*, permitindo a consulta de todas as alterações efetuadas.

Foi já referido durante a contextualização desta plataforma que é permitida a cópia de ficheiros para dentro das várias *jails* criadas. À exceção dos ficheiros de texto que sofrem uma conversão da sua codificação e dos ficheiros arquivados que serão extraídos, não existe mais nenhum tipo de tratamento especial. Relativamente à extração de ficheiros arquivados, foi dado suporte a alguns formatos mais conhecidos zip, rar, tar, tar.gz, pelo que qualquer ficheiro com esta extensão será automaticamente extraído. Estes ficheiros podem conter qualquer conteúdo, desde o código a compilar e executar como a ficheiros auxiliares à execução, desta forma também estes ficheiros poderão ser alvos de uma conversão da sua codificação. Caso o código fonte seja submetido num ficheiro arquivado num dos formatos já referidos, o nome do arquivo (não contando a extensão do ficheiro) terá de ser igual ao nome do ficheiro do código fonte responsável por iniciar a execução, ou seja, o nome do arquivo terá de ser igual ao nome do ficheiro onde se encontra a função *main()* do programa principal.

Não existe forma da PEPR saber qual o ficheiro pelo qual irá dar início à execução do programa submetido. Foi então necessário fazer uso de variáveis de ambiente que são passadas para dentro do ambiente protegido durante a sua criação com o nome do ficheiro submetido pelo aluno. Desta forma,

durante o processo de criação dos testes no SAAL, e uma vez que o professor também não vai ter acesso aos nomes com os quais os alunos irão efetuar a submissão, o professor terá de ter o cuidado de usar *\$SUBMITTED_FILE* sempre que pretender fazer referencia ao ficheiro submetido pelos alunos. Esta variável representa o nome do ficheiro submetido até ao último ponto final encontrado, removendo assim a extensão do ficheiro, no caso de existir. A extensão será dada pela definição da linguagem de programação escolhida, “.java” para Java, “.py” para Python, “.cpp” para C++, etc.

Após a execução na *jail*, a plataforma fica responsável por arquivar todos os ficheiros alterados durante este processo. O arquivamento é depois guardado numa pasta pré definida nas configurações e categorizado por dia, identificador do exercício e depois pelo número mecanográfico do aluno. Se alguma desta informação não se encontrar presente para a PEPR, esta preenche os campos em falta com valores por omissão.

Capítulo 4

Avaliação e Resultados

Durante o desenvolvimento deste trabalho, especialmente no desenvolvimento da interface gráfica do web site, foi necessário efetuar vários testes de avaliação de usabilidade. Grande parte dos teste foram realizados com colegas que se submetiam a uma pequena avaliação de funcionalidades permitindo obter *feedback* do melhor formato para implementação de determinadas funcionalidades.

Foi também realizada uma reunião informal com um professor responsável pela organização de várias disciplinas de programação de computadores no DETI, com objetivo fundamental de conduzir o desenvolvimento do sistema para uma aplicação final capaz de dar resposta às necessidades experimentadas no ensino destas disciplinas.

Na fase final do desenvolvimento deste trabalho foram ainda realizadas algumas experiências que contaram com alunos de várias turmas da disciplina Programação II com objetivo de testar em ambiente de produção o funcionamento do sistema. Foi também pedido a todos os alunos que fizeram uso do sistema para preencherem um pequeno inquérito on-line. O facto de se encontrar on-line, permitiu aos alunos que não fizeram uso do sistema durante o período de aulas, tivesse também a oportunidade de testar o sistema e contribuir com mais respostas ao inquérito.

4.1 SAAL

Do trabalho desenvolvido, este sistema foi o que mais atenção recebeu durante o seu desenvolvimento com as questões de performance dos serviços e usabilidade do web site sempre presentes. Desta forma foram efetuadas várias medições dos tempos de resposta médio dos serviços e verificou-se que estes variam entre os 40ms (milissegundos) para a consulta de serviços mais básicos, como é o caso da consulta da lista de cursos, a 275ms na consulta do serviço que fornece informação de edições e respetiva informação estatística. Relativamente à última medição, este valor poderá ser facilmente influenciável pelo número de alunos da edição que conduzem a um crescimento do número de submissões efetuadas e desta forma levam a um conjunto de dados maior que por sua vez obrigam a efetuar mais cálculos para aferição da informação estatística.

O inquérito respondido pelos aluno também se baseia quase exclusivamente nesta solução. Seguidamente serão apresentados vários gráficos criados de acordo com as respostas obtidas pelo inquérito respondido pelos alunos e que conta com 170 respostas. Espera-se que este valor seja suficientemente significativo.

Visualmente acha o site

Esta questão, de um âmbito muito subjetivo, relaciona-se diretamente com o aspeto da aplicação web com objetivo de tentar perceber a opinião dos alunos.

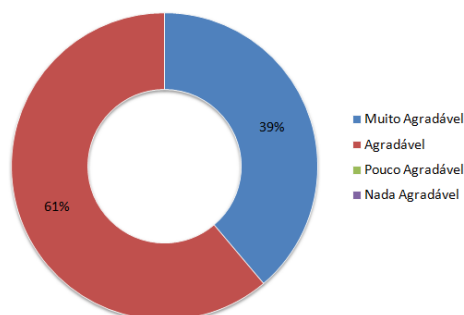


Figura 4.1: Design da página web

Pela Figura 4.1, cerca de 60% dos utilizadores acharam o web site agradável e mais de 40% consideraram-o mesmo como sendo muito agradável visualmente.

Linguagem utilizada

É pretendido também avaliar como se relacionam os termos utilizados na aplicação web com o contexto em que esta se enquadra.

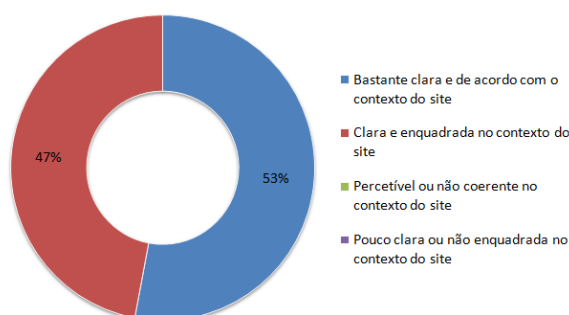


Figura 4.2: Linguagem utilizada

A Figura 4.2 faz a análise da opinião dos utilizadores e pode-se observar que estes não apresentam grandes dúvidas acerca dos conceitos utilizados na página web e que mais de 50% dos utilizadores acha mesmo que a linguagem utilizada é bastante clara e totalmente de acordo com o contexto inserida.

Navegabilidade no site

A Figura 4.3 mostra os resultados obtidos relativamente à navegabilidade do web site desenvolvido. Esta foca-se na facilidade proporcionada ao utilizador de encontrar a informação desejada com o menor número de clicks e vai de acordo à forma como a informação é organizada.

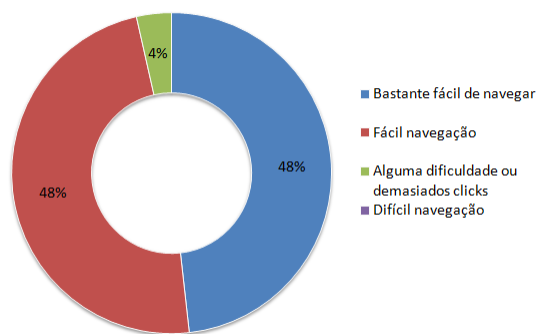


Figura 4.3: Navegabilidade da página web

Pela análise do gráfico da figura pode-se observar que existiu uma pequena minoria dos utilizadores, cerca de 6%, que encontraram algum tipo de dificuldades durante a sua visita no web site. Ainda assim, quase 50% dos utilizadores não sentiu qualquer tipo de problema e outros tanto consideraram fácil a navegação. Este teste não conta com resultados da página web num dispositivo móvel e cuja experiência de navegação será relativamente diferente.

Consistência do site

Esta questão vai mais fundo que a navegabilidade e pretende investigar até que ponto as várias páginas se comportam da mesma forma ou se a interação com estas se desenrola sempre de uma forma espetável. Este comportamento pode também ser responsável por influenciar a navegabilidade entre as várias páginas web.

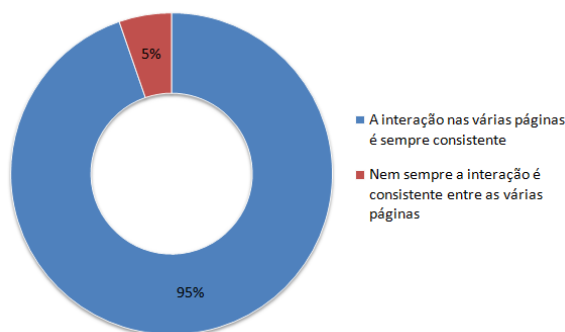


Figura 4.4: Consistência da página web

Os utilizadores foram praticamente unânimes ao afirmar que a interação entre as várias páginas é sempre consistente com quase 95% das respostas.

Feedback após tomada de alguma ação

Nesta questão pretende-se estudar se o utilizador é informado na realização das suas ações, ou seja, ao proceder à realização de uma qualquer operação, o web site informa o utilizador do estado dessa operação?

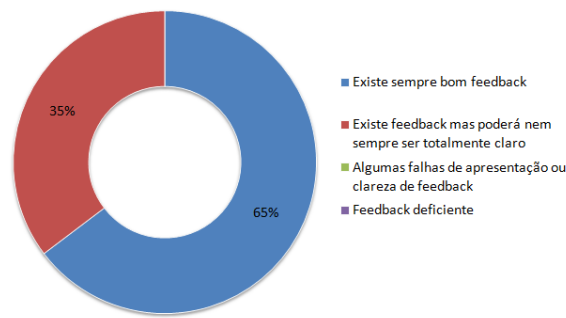


Figura 4.5: Feedback após uma ação

Apenas 35% dos utilizadores acharam que existe sempre feedback mas que de alguma forma este deveria ser mais explícito e claro, enquanto 65% destes concorda que o feedback é muito bom, como se pode observar pela Figura 4.5.

Tempo de resposta do site após uma ação

Não só o feedback após a tomada de uma ação é importante, é também necessário perceber como se comportam os tempos de espera entre as várias ações realizadas.

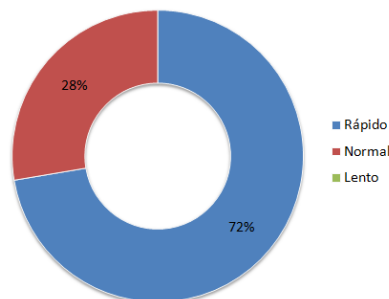


Figura 4.6: Tempos de respostas

De acordo com Figura 4.6 os resultados dos tempos de resposta da página web são muito bons e podem ser explicados por dois motivos. O primeiro é sem dúvida a forma como este foi desenvolvido, ou seja, uma aplicação baseada na arquitetura *Single Page Application* 3.4.6 juntamente com a *framework* Backbone.js 3.4.7 que comunica com mensagens assíncronas com os serviços desenvolvidos. Relativamente ao segundo motivo, este é uma consequência direta de uma estratégia de implementação, ou seja, o web site contém várias animações que ocorrem sempre que é realizada uma ação ou quando acontece uma atualização de parte da página, esta animação pode demorar até cerca de meio segundo, o que na maioria das vezes é o tempo necessário para efetuar um pedido de informação e receber a respectiva resposta, desta forma quando a animação terminar, já a informação está pronta para ser apresentada.

Prevenção de erros do utilizador

A prevenção de erros do utilizador prende-se com a colocação de mensagens de aviso que confirmem a ação que o utilizador irá realizar, tais como ações definitivas que não permitem voltar ao estado anterior.

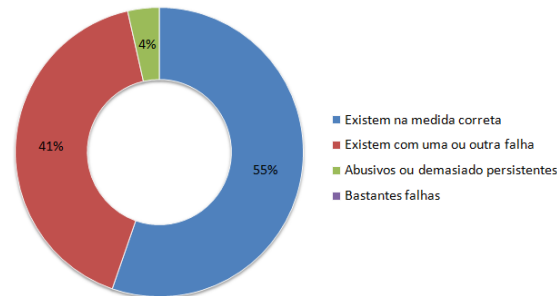


Figura 4.7: Prevenção de erros do utilizador

Esta questão parece ter gerado respostas inconsistentes como demonstrado pela Figura 4.7. Se por um lado existe uma minoria de utilizadores que achou abusivos as mensagens preventivas de ações definitivas e que 40% achou que existiam falhas nestas mensagens, também mais de metade dos utilizadores concordou que estas aparecem na medida correta.

Feedback relativo à submissão de trabalhos

Nesta questão pretende-se estudar a avaliação feita pelos alunos acerca da informação disponibilizada pelo sistema após a submissão de código fonte e respetiva execução.

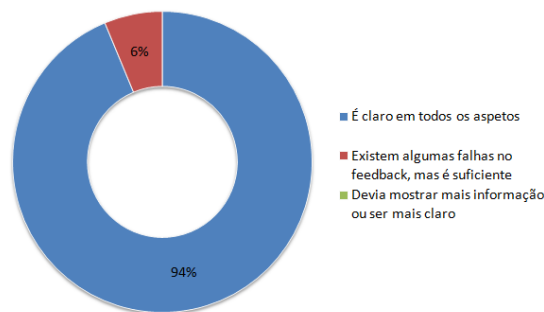


Figura 4.8: Feedback da submissão de trabalhos

Este ponto é de extrema importância por atestar o entendimento obtido pelos alunos nas suas submissões de exercícios e avalia a capacidade que o aluno tem ao entender o detalhe da informação apresentada. Pela Figura 4.8 conclui-se que quase a totalidade dos alunos envolvidos consideram que a informação apresentada é de fácil entendimento e que esta serve para mais facilmente detetarem as suas falhas e procederem às correções necessárias.

4.2 PEPR

A PEPR trabalha num plano que não necessita de uma interface gráfica para contacto direto com os utilizadores do sistema, assim sendo, os resultados são mais objetivos por não existir a componente subjetiva da avaliação de uma página web.

A avaliação desta plataforma relaciona-se com o número de execuções processados por segundo de forma concorrential, memória adicional ocupada pelo sistema e tempo de execução das várias instâncias. Cada execução é caracterizada pela criação de um ambiente virtual para execução do código fonte submetido, cópia dos ficheiros submetidos para dentro desse ambiente, compilação e execução do código e análise do resultado obtido. Os testes abrangidos pela plataforma tentam determinar a capacidade de resposta desta a múltiplas submissões efetuadas pelos alunos. Foram então preparadas várias rondas de execução de um programa em Java, que apenas efetua a divisão de dois números recebidos por argumentos e imprime o resultado, onde apenas varia o número de execuções concorrentes entre 1 e 7. A Figura 4.9 resume a informação do número de execuções em simultâneo (em baixo), a quantidade de memória ocupada em MB por cada execução (esquerda) e o tempo de execução médio de cada instância.

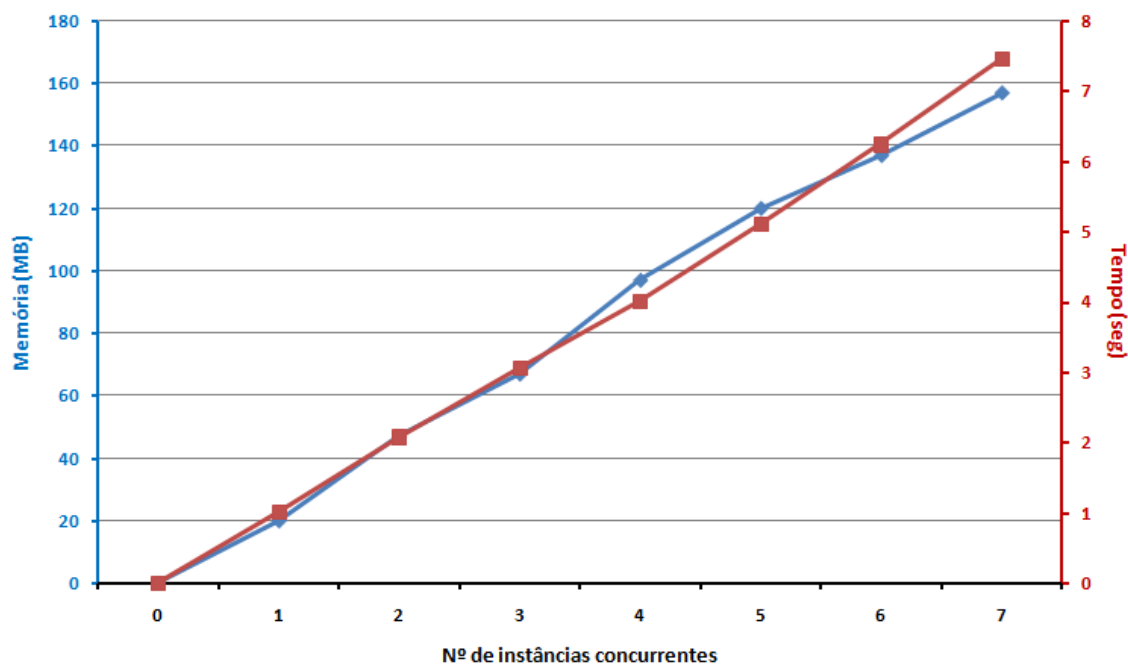


Figura 4.9: Resultados do tempo de execução e memória ocupada por número de execuções

Pela observação dos resultados da Figura 4.9, existe um aumento de memória utilizada de aproximadamente 20MB com apenas uma execução e que toma cerca de 1 segundo a concluir todo o processo de execução. Também se pode verificar que existe um crescimento linear da memória ocupada e do tempo de execução de acordo o número de execuções simultâneas a acontecer na PEPR.

Foram também analisados os tempos de criação e destruição dos ambientes virtuais para execução de código, mas desta vez não envolvendo nenhuma ação de compilação ou execução de código. Concluiu-se com estes testes que para o processo de criação do ambiente virtual, cópia dos ficheiros e

destruição do respetivo ambiente são necessários sensivelmente 40 a 50 milissegundos e foi necessário criar cerca de 20 destes ambientes de forma a incrementar a memória utilizada do sistema em apenas 1MB. Estes tempos seriam ainda mais reduzidos se não existisse a necessidade de efetuar a cópia do ficheiro para cada um dos novos ambientes virtuais. Esta necessidade deve-se à obrigatoriedade que existe no pedido de execuções da PEPR uma vez que não teria sentido efetuar um pedido vazio. Estes são assim rejeitados pela plataforma antes de efetuar qualquer atividade.

Conclui-se assim que os resultados obtidos de memória e tempos de execução da Figura 4.9 se relacionam exclusivamente com a instanciação da JVM.

Pelos dados apresentados e juntamente com a capacidade do SAAL em distribuir a carga dos pedidos de execução por várias PEPR que se encontrem em execução, é então possível partilhar a carga das execuções por múltiplas plataformas mitigando os tempos de necessários para concluir as várias execuções mesmo que estas ocorram em simultâneo.

Capítulo 5

Conclusões

Era pretendido neste trabalho a implementação de um sistema destinado ao ensino de linguagens de programação com uma forte presença das funcionalidades vistas em ferramentas de e-learning. A aproximação de uma solução para resolver a necessidade apresentada neste trabalho passaria pela integração das principais funcionalidades das ferramentas de e-learning com as ferramentas de análise e execução de código fonte, explorando sempre as mais recentes tecnologias e conceitos arquiteturais.

De forma a concretizar uma ferramenta capaz de relacionar estes mundos e assim criar um ambiente de e-learning mais adaptado às necessidades atuais, foram desenvolvidas as duas soluções já apresentadas. O SAAL, atua como *front-end* para professores e alunos, é uma ferramenta que segue os princípios da arquitetura SOA, tendo desta forma um enorme potencial de crescimento e modularidade. Esta ferramenta permite a criação de turmas, exercícios e acompanhar o desempenho dos alunos. Permite igualmente definir testes que são aplicados às submissões. Sendo modular, é possível implementar novos serviços capazes de efetuar diferentes tipos de análise de código, consulta de informação estatística, análise de requisitos programáticos (uso de recursividade, funções iterativas específicas, etc.), entre outros. A PEPR, que consiste no *back-end* do sistema, expõe um serviço REST que permite criar dinamicamente ambientes virtuais protegidos onde é executado e validado o código submetido. Esta ferramenta permite que os ambientes virtuais sejam criados de forma extremamente rápida e garante isolamento da execução. De salientar que existe um forte suporte de auditoria, permitindo identificar todas as ações executadas sobre o sistema de ficheiros, que são registadas e arquivadas para análise. Tendo em conta os requisitos de cada exercício e o resultado dos testes efetuados na PEPR, é fornecida no SAAL informação detalhada que permite ao aluno identificar pontos a melhorar nos seus programas.

Este trabalho exigiu um esforço que permitisse tornar o núcleo das duas aplicações desenvolvidas à prova de erros, foi necessário executar inúmeros testes de integridade para garantir que não existiam falhas no sistema de autorização desenvolvido bem como nas próprias funcionalidades internas de cada serviço. Foi um trabalho com um longo período de desenvolvimento, principalmente devido às inúmeras funcionalidades que foram criadas: um conjunto de oitenta e quatro serviços REST, uma aplicação web baseada na arquitetura *Single Page Application*, e ainda uma plataforma para criação de ambientes protegidos de execução de código.

5.1 Contribuições

O trabalho desenvolvido e agora apresentado contribuiu com mais uma aplicação de apoio ao ensino à distância capaz de efetuar compilação, execução e análise de código fonte. Tais capacidades não eram até então encontradas de forma integrada num único sistema. Foi também submetido um artigo baseado neste trabalho ao Simpósio de Informática INFORUM ¹, a realizar na Universidade de Évora.

5.2 Trabalho Futuro

Existe ainda muito trabalho que poderá ser desenvolvido e integrado nas duas soluções apresentadas. Pelo facto de existir uma aplicação baseada em princípios SOA abre à partida um enorme leque de possibilidades para futuros trabalhos ou integrações com outros sistemas, mesmo sistemas externos.

De um ponto de vista de alto nível, foi desenvolvido um núcleo funcional e relativamente bem estruturado com as funcionalidades mínimas ao bom funcionamento da aplicação para um ambiente de produção. Ainda assim, existe trabalho extra que se poderia desenvolver e passaria pela implementação de funcionalidades de pequena dimensão ou sem grande valor acrescentado, ou mais importante, a criação de novas funcionalidades capazes de acrescentar valor ao trabalho já desenvolvido. São então apresentadas algumas funcionalidades relativas às duas soluções desenvolvidas que poderão ser fatores a ter em consideração para o sucesso desta aplicação:

- deve ser criada a documentação da API desenvolvida para uso externo e para que possam ser criadas outras aplicações que reutilizem tanto o SAAL para gestão, como a PEPR para execução de código.
- a integração com sistemas como o Code.UA [37] ou GitHub [21] e a funcionalidade de executar o código dos vários projetos destas plataformas na PEPR.
- implementação de métodos capazes de deixar de fazer apenas análise cega dos resultados produzidos e passar a avaliar a correta aplicação dos conceitos no código fonte submetido.
- implementação de mecanismos capazes de efetuar a deteção de cópias dos vários programas submetidos. A utilização de ferramentas, como Clone Digger [38], ajudariam neste processo pela análise esquemática da estrutura do código fonte produzido.
- uma vez que todas as configurações são mantidas em ficheiros de texto, criar um script que permita de forma fácil para os administradores, a instalação de todo o sistema.

São também seguidamente apresentadas algumas funcionalidades categorizadas por projeto.

SAAL

Relativamente ao SAAL, é ainda necessário aplicar algum esforço para perceber se as interfaces gráficas desenvolvidas trabalham como esperado. No caso de tal não acontecer, estudar qual a melhor forma de as resolver. Esta necessidade surge do facto de não se conseguir perceber se o número de respostas obtidas são suficientemente demonstrativas de uma população de alunos com diferentes objetivos, conhecimentos e cultura.

¹INFORUM - Simpósio de Informática, <http://inforum.org.pt/INForum>

Existem também algumas pequenas funcionalidades úteis, relacionadas com a interface gráfica da página web, que se podem implementar. Na vista de professor, na listagem de submissões efetuadas, é necessário a implementação de novas funcionalidades que permitam proceder a uma nova execução, tanto de uma submissão específica de determinado aluno, como de todas as submissões realizadas até então. Esta necessidade surge após a alteração de um teste que implica que todas as submissões sejam novamente executadas de acordo com a definição dos novos testes. Ainda na vista do professor, será interessante manter um histórico dos professores que modificam os exercícios ou testes; permitir também que seja possível replicar testes já existentes do mesmo exercício de forma a não obrigar o professor a criar sempre novos testes mesmo quando estes são muito semelhantes.

Seria útil também criar um mecanismo de troca de mensagens entre utilizadores, ou pelo menos um sistema de envio de mensagens de professores para alunos como forma de avisos. Esta última seria útil pois permitira aos professores avisarem os seus alunos de alterações de última hora e estes receberiam em tempo real a mensagem ou quando fizessem autenticação no sistema.

Falta ainda uma página de gestão de utilizadores, os serviços para dar suporte a essa página existem, mas tal página acabou por não ser desenvolvida. Existe a possibilidade de exportação de informação estatística de edições no formato CSV com a qual o professor pode gerar gráficos, mas seria também interessante esta implementação na própria aplicação web.

De um ponto de vista pedagógico seria necessário encontrar uma forma capaz de acompanhar a evolução dos alunos, tanto por turma como individualmente. Poderiam-se assim encontrar mais facilmente os pontos sensíveis que criam mais obstáculos à aprendizagem destas disciplinas em concreto. A criação de um sistema de *achievements* poderia despertar maior interesse pelos alunos no sistema e na concretização de todos os exercícios, mesmo que estes não se encontrem na lista obrigatória.

PEPR

Esta plataforma foi inicialmente pensada com um pequeno conjunto de funcionalidades que mais tarde foram crescendo. Desta forma seria suposto que a sua estrutura inicial tivesse acompanhado tais mudanças que iam ocorrendo ao nível das funcionalidades. Tal não aconteceu da melhor forma e neste momento a PEPR encontra-se estruturalmente não tão bem conseguida como seria de esperar. Esta estruturação não influencia o seu bom funcionamento, mas ao nível programático não cumpre as melhores práticas.

Espera-se que todas as funcionalidades desenvolvidas estejam corretamente implementadas, mas em alguns casos essa implementação não aconteceu da melhor forma, por exemplo, o sistema de monitorização do tempo de execução deveria ser otimizado e desenvolvido de acordo com os paradigmas da programação orientada a objetos.

O resultado de saída de todas as execuções deve também ser guardado num ficheiro a ser arquivado juntamente com todos os ficheiros da execução. A existência deste ficheiro permitiria a realização de análises às execuções quando necessário.

Neste momento a resposta da PEPR a uma execução apenas contem o resultado da saída dessa execução e um campo que anuncia se o resultado obtido se encontra de acordo com o que seria esperado. Numa próxima versão deveria-se também incluir as diferenças entre o que seria esperado e o que realmente resultou, este novo campo permitiria na interface gráfica (web ou móvel ou outra) exibir de melhor forma as diferenças entre ambos, realçando com cores essas diferenças.

Bibliografia

- [1] Michele Amaral dos Santos Silva Abreu. *Avaliação da Intenção de Uso de um Sistema de e-learning por alunos de uma Instituição de Ensino Superior do Rio de Janeiro*. PhD thesis, Universidade do Grande Rio, 2012.
- [2] Moodle Trust. Moodle.org. <https://moodle.org>, Maio 2013.
- [3] Blackboard Inc. Blackboard international | emea. <http://www.blackboard.com/>, Junho 2013.
- [4] California State University. Csus programming contest control pc2. <http://www.ecs.csus.edu/pc2/>.
- [5] Cassio de Campos Carlos Ferreira. Boca: A support system for programming contests. *Brazilian Workshop on Education in Computing*, 2004.
- [6] Sphere Research Labs. Sphere online judge. <http://www.spoj.com>.
- [7] Royal Institute of Technology. Kattis. <https://kth.kattis.scrool.se>, Abr 2013.
- [8] José Paulo Leal. Mooshak. <http://mooshak.dcc.fc.up.pt/zp/mooshak/>.
- [9] Pedro Ribeiro. Oni'2012 - olimpíadas nacionais de informática. <http://www.dcc.fc.up.pt/oni/2012/index.cgi?page=faq>, Maio 2012.
- [10] Alexander Lyabah. Checkio - code gaming and competition platform. <http://www.checkio.org>, Abr 2013.
- [11] Mike Mirzayanov. Codeforces. <http://codeforces.com/>, Maio 2013.
- [12] Princeton University. Elo rating system. http://www.princeton.edu/~achaney/tmve/wiki100_k/docs/Elo_rating_system.html, Maio 2013.
- [13] TopCoder Inc. Topcoder, inc. | home of the world's largest development community. <http://www.topcoder.com>, Maio 2013.
- [14] CrunchBase. Topcoder | crunchbase profile. <http://www.crunchbase.com/company/topcoder>, Março 2013.
- [15] Oracle Corporation. Java programming language. <http://docs.oracle.com/javase/7/docs/technotes/guides/language>, Maio 2013.
- [16] Jason Greene. Why is jboss as 7 so fast? <http://in.relation.to/Bloggers/WhyIsJBossAS7SoFast>, Fevereiro 2013.

- [17] Rebel Labs. The great java application server debate with tomcat, jboss, glassfish, jetty and liberty profile. <http://zeroturnaround.com/rebellabs/the-great-java-application-server-debate-with-tomcat-jboss-glassfish-jetty-and-liberty-profile/>, Fevereiro 2013.
- [18] Apache Foundation. Maven - what is maven? <http://maven.apache.org/what-is-maven.html>, Abr 2013.
- [19] Silvia Benvenuti Mark Boas and Mark Panaghiston. The future of web apps – single page applications. <http://happyworm.com/blog/2010/08/23/the-future-of-web-apps-single-page-applications/>, Maio 2013.
- [20] Inc Twitter. Twitter bootstrap. <http://twitter.github.io/bootstrap>, Maio 2013.
- [21] Inc GitHub. Github. <https://github.com/>, Maio 2013.
- [22] Oracle Corporation. *MySQL 5.6 Reference Manual*. Oracle Corporation, 2013.
- [23] Oracle Corporation. Oracle. www.oracle.com, Maio 2013.
- [24] MariaDB Foundation. Welcome to mariadb! www.mariadb.org, Maio 2013.
- [25] Red Hat. Hibernate - jboss community. <http://www.hibernate.org/>, Mai 2013.
- [26] Eclipse Foundation. Eclipse link home. <http://www.eclipse.org/eclipselink/>, Mai 2013.
- [27] Apache. Apache openjpa. <http://openjpa.apache.org/>, Mai 2013.
- [28] Hal Lockhart Scott Cantor Anil Saldhana Nathan Klingenstein, Thomas Hardjono. Oasis security services (saml) tc, Setembro 2012.
- [29] Scott W. Ambler. *The Elements of UML(TM) 2.0 Style*. University of Cambridge, 2005.
- [30] Carlos Pampulim Caldeira. *Introdução ao Modelo de Dados Relacional*. Universidade de Évora, 2004.
- [31] Jos Hirth. Stringbuilder vs stringbuffer vs string.concat - done right. <http://kaioa.com/node/59>, Maio 2013.
- [32] Manuela Neves. *Introdução à Estatística e à Probabilidade*. International Society of Automation, 2008.
- [33] Automattic Team. Gravatar - globally recognized avatars. <http://pt.gravatar.com/>, Maio 2013.
- [34] Jakob Nielsen. 10 heuristics for user interface design. *Jakob Nielsen's Alertbox*, 1995.
- [35] Canonical Ltd. The world's most popular free os | ubuntu. <http://www.ubuntu.com/>, Maio 2013.
- [36] Python Software Foundation. Python 2.7.5. <http://www.python.org/doc/>, Maio 2013.
- [37] Universidade de Aveiro. Code.ua. <https://code.ua.pt/>, Junho 2013.
- [38] Clone digger. <http://clonedigger.sourceforge.net/>, Junho 2013.
- [39] Solid IT group. Db-engines ranking. <http://db-engines.com/en/ranking>.
- [40] Artur Adib. Hello backbone.js. <http://arturadib.com/hello-backbonejs/>, Dez. 2012.

- [41] Jeremy Ashkenas. Backbone.js. <http://backbonejs.org/>, Março 2013.
- [42] Derick Bailey. Asynchronously load html templates for backbone views. <http://lostechies.com/derickbailey/2012/02/09/asynchronously-load-html-templates-for-backbone-views/>, Feb 2012.
- [43] Eric Bidelman. Html5 rocks - a resource for open web html5 developers. <http://www.html5rocks.com/en/>.
- [44] Bill Burke. *RESTful Java with JAX-RS*. O'Reilly, 2010.
- [45] Carlos Ferreira Cassio de Campos. Boca online contest administrator. <http://www.ime.usp.br/cassio/boca/>.
- [46] Gavin King Christian Baver. *Java Persistence with Hibernate*.
- [47] CodeChefe. Programming community, programming contest platform | codechef. <http://www.codechef.com/>.
- [48] Thomas Davis. *Backbone Tutorials - Beginner, Intermediate and Advanced*. Jun. 2012.
- [49] Fabien Doiron. alertify.js - browser dialogs never looked so good. <http://fabien-d.github.io/alertify.js/>, Maio 2013.
- [50] Fredrik Niemela Pehr Soderman Emma Enstrom, Gunnar Kreitz and Viggo Kann. Five years with kattis using an automated assessment system in teaching. *41st ASEE/IEEE Frontiers in Education Conference*, 2011.
- [51] Joseph Gradecki Eric Pugh. *Professional Hibernate*.
- [52] Pedro Miguel Gonçalves Ferreira. *Dados Abertos @ UA*. PhD thesis, Universidade de Aveiro, 2012.
- [53] jQuery Foundation. jquery. <http://jquery.com/>, Março 2013.
- [54] Mozilla Foundation. Javascript. <https://developer.mozilla.org/en-US/docs/JavaScript>, Nov 2012.
- [55] Steve Friedl. Best practices for unic chroot() operations. <http://www.unixwiz.net/techtips/chroot-practices.html>, Maio 2013.
- [56] António Gonçalves. *Beginning Java EE 6 Platform with GlassFish 3*. 2010.
- [57] Arun Gupta. *Java EE 6*. 2012.
- [58] Colin Hughes. Project euler. <http://projecteuler.net>, Feb. 2012.
- [59] Dave Minter Jeff Linwood. *Beginning Hibernate*. Apress, 2010.
- [60] Ian Robinson Jim Webber, Savas Parastatidis. *REST in Practice*. O'Reilly Media, 2010.
- [61] Sam Ruby Leonard Richardson. *RESTful Web Services*. O'Reilly Media, 2007.
- [62] Fernando Mantoan. Backbone.js tutorials series. <https://github.com/fernandomantoan/backbone-tutorial-series>, Maio 2013.

- [63] Addy Osmani. *Learning JavaScript Design Patterns*. O'Reilly, 2012.
- [64] Addy Osmani. *Developing Backbone.js Applications Early Release*. O'Reilly, out. edition, 2012.
- [65] Universidade Estadual Paulista. Aprender: O que é moodle? <http://aprender.rosana.unesp.br/mod/resource/view.php?id=254>, Maio 2013.
- [66] Mark Pilgrim. *Dive Into Python, Python from novice to pro*. Apress, 2004.
- [67] Kevin Roebuck. *Security Assertion Markup Language (SAML): High-impact Strategies*. Tebbo, 2011.
- [68] Mikito Takada. *Single page apps in depth*. On-line book, 2012.
- [69] Ed Simon Takeshi Imamura, Blair Dillaway. W3c, xml encryption syntax and processing, Dezembro 2002.
- [70] Spencer Tipping. *Javascript in Ten Minutes*. 2011.
- [71] Neil Paiva Tizzo. Arquitetura orientada a serviços. *GEINFO*, 2010.
- [72] Wikibooks.org. *MySQL*. Wikibooks.org, 2013.
- [73] Tarek Ziadé. A new development era (essay). <http://blog.ziade.org/2013/01/25/a-new-development-era-essay>, Jan. 2013.

Anexos

Lista de Serviços

Na tabela 1 são apresentados os identificadores de cada um dos serviços desenvolvidos que necessitam de autenticação do utilizador, o nome do identificador é descritivo da funcionalidade do serviço em causa. É também possível consultar as permissões do grupo de utilizadores pelos respetivos serviços.

Tabela 1: Lista de permissões de serviços por grupo

	Administrador	Professor	Aluno
curricula.get.getUser	Sim	Sim	Sim
curricula.get.getUserByNMec	Sim	Sim	Sim
curricula.get.getUserByUU	Sim	Sim	Sim
curricula.get.course	Sim	Sim	Sim
curricula.get.getAllCourses	Sim	Sim	Sim
curricula.post.postCourse	Sim	Não	Não
curricula.put.putCourse	Sim	Não	Não
curricula.get.getSubject	Sim	Sim	Sim
curricula.get.getSubjectByEdition	Sim	Sim	Sim
curricula.get.getSubjects	Sim	Sim	Sim
curricula.get.getAllSubjects	Sim	Não	Não
curricula.post.postSubject	Sim	Não	Não
curricula.put.putSubject	Sim	Não	Não
curricula.delete.deleteSubject	Sim	Não	Não
curricula.get.getEdition	Sim	Sim	Sim
curricula.get.getMyEditions	Sim	Sim	Sim
curricula.get.getAllEditionsBySubject	Sim	Não	Não
curricula.get.getMyEditionsBySubject	Sim	Sim	Sim
curricula.post.postEdition	Sim	Não	Não
curricula.put.putEdition	Sim	Sim	Não
curricula.delete.deleteEdition	Sim	Não	Não
curricula.get.getClass	Sim	Sim	Sim
curricula.get.getClassesByEdition	Sim	Sim	Sim
curricula.post.postClass	Sim	Sim	Não
curricula.put.putClass	Sim	Sim	Não
curricula.delete.deleteClass	Sim	Sim	Não
curricula.get.getStudent	Sim	Sim	Sim

curricula.get.getStudentsByclass	Sim	Sim	Sim
curricula.get.getStudentsByEdition	Sim	Sim	Sim
curricula.post.postStudent	Sim	Sim	Não
curricula.put.putStudent	Não	Não	Sim
curricula.delete.deleteStudent	Sim	Sim	Não
curricula.get.getProfessor	Sim	Sim	Sim
exercises.get.importExercisesByEdition	Sim	Sim	Não
curricula.get.getProfessorsByClass	Sim	Sim	Sim
curricula.put.putProfessor	Sim	Sim	Não
curricula.get.getProfessorsByEdition	Sim	Sim	Sim
curricula.get.getDirectorsByEdition	Sim	Sim	Sim
curricula.delete.deleteCourse	Sim	Não	Não
exercises.get.getModule	Sim	Sim	Sim
exercises.get.getModulesByEdition	Sim	Sim	Sim
exercises.get.getExercise	Sim	Sim	Sim
exercises.get.getExercisesByModule	Sim	Sim	Sim
exercises.post.postModule	Sim	Sim	Não
exercises.put.putModule	Sim	Sim	Não
exercises.delete.deleteModule	Sim	Sim	Não
exercises.post.postExercise	Sim	Sim	Não
exercises.put.putExercise	Sim	Sim	Não
exercises.delete.deleteExercise	Sim	Sim	Não
exercises.get.getLanguages	Sim	Sim	Sim
exercises.post.postLanguage	Sim	Sim	Não
exercises.put.putLanguage	Sim	Sim	Não
exercises.delete.deleteLanguage	Sim	Sim	Não
curricula.get.getGroup	Sim	Sim	Sim
curricula.get.getAllGroupsByStudent	Sim	Sim	Sim
curricula.get.getMyGroupsByEdition	Não	Não	Sim
curricula.post.postGroup	Não	Não	Sim
curricula.put.putGroup	Não	Não	Sim
curricula.delete.deleteGroup	Não	Não	Sim
exercises.get.getFile	Sim	Sim	Sim
exercises.get.getFilesByExercise	Sim	Sim	Sim
exercises.post.postFile	Sim	Sim	Sim
exercises.delete.deleteFile	Sim	Sim	Sim
exercises.get.getTest	Sim	Sim	Sim
exercises.get.getTestsByExercise	Sim	Sim	Sim
exercises.post.postTest	Sim	Sim	Não
exercises.put.putTest	Sim	Sim	Não
exercises.delete.deleteTest	Sim	Sim	Não
exercises.get.file	Sim	Sim	Sim
exercises.get.getCommit	Sim	Sim	Sim
exercises.get.getAllCommitsByExercise	Sim	Sim	Não
exercises.get.getMyCommitsByExercise	Não	Não	Sim

exercises.post.postCommit	Não	Não	Sim
exercises.put.putFile	Sim	Sim	Não
exercises.get.getStatsByEdition	Sim	Sim	Não