UNIVERSIDADE D
COIMBRA

Bruno Rafel Primo Vicente

IMPLEMENTAÇÃO DE BACKOFFICE WEB PARA GESTÃO DO OPENSIMULATOR DA UAB

Dissertação no âmbito do Mestrado em Engenharia Informática, especialização em Engenharia de Software orientada pelo Professor Doutor Pedro Furtado e pelo Professor Doutor Leonel Morgado e apresentada à Faculdade de Ciências e Tecnologia / Departamento de Engenharia Informática.

Janeiro de 2019

Agradecimentos

Na realização desta Dissertação de Mestrado contei com o apoio de várias pessoas e instituiçoes sem as quais não seria possível a realização deste trabalho.

Em primeiro lugar quero agradecer aos meus orientadores, Professor Doutor Pedro Furtado e Professor Doutor Leonel Morgado, por todas as orientações e críticas forncecidas ao longo do decorrer deste trabalho que elevaram a qualidade do resultado final.

Agradeço também à Universidade Aberta pelas infraestruturas disponibilizadas para a realização do trabalho e aos seus administradores de sistemas de informação pelo tempo disponibilizado e pelas opiniões e críticas ao trabalho realizado, tendo sido estas bastante úteis para a aumentar qualidade do trabalho.

Aos meus colegas de curso agradeço todos os momentos vividos durante o meu percurso académico. Amizades e momentos que serão relembrados durante toda a vida.

À minha namorada, Sofia, agradeço todo o apoio prestado ao longo não só desta dissertação como ao longo de todo o meu percurso académico. Sem o apoio e a coragem transmitidos não teria sido possível a conclusão deste trabalho.

Agradecer, por último, à minha família todo o apoio prestado durante todo o meu percurso académico. Toda a coragem e carinho transmitidos foram também indispensáveis para a conclusão desta etapa.

A todos vocês dedico o meu trabalho.

Esta página foi deixada em branco intencionalmente.

Resumo

A utilização de mundos virtuais em ambientes pedagógicos tem sido alvo de estudo devido tanto às suas capacidades de prototipagem, como de despertar um maior interesse nos alunos. Muitas instituições tentaram já incorporar o OpenSimulator como uma ferramenta de ensino, mas a dificuldade em gerir esta tecnologia e a falta de funcionalidades desta para este tipo de uso é evidente.

Neste estágio foi utilizada uma abordagem de Engenharia de Software para o desenvolvimento de uma ferramenta de gestão que permita a gestão da tecnologia OpenSimulator em ambiente académico. Para este efeito, a identificação de requisitos funcionais foi efectuada revendo a literatura acerca da adopção de OpenSim em ambientes empresariais assim como a literatura pedagógica com esta tecnologia para identificar que tipos de tarefas são executadas com esta tecnologia. Para além disto foram entrevistados administradores de sistemas de forma a identificar os desafios da gestão de um sistema de informação e identificar requisitos relativos à administração do sistema. Após o denvolvimento da solução foram efectuados testes ao sistema de forma a validar backoffice web.

O produto deste estágio é um protótipo de uma ferramenta que permite uma fácil gestão do OpenSimulator em ambiente pedagógico. Para além de gerir o OpenSim o *backoffice* tem também várias funcionalidades complementares como a gestão de eventos no mundo virtual, o controlo de presenças e a gestão dos logs do sistema.

Palavras-chave

OpenSimulator; OpenSim; mundos virtuais; gestão de sistemas; administração de sistemas.

Esta página foi deixada em branco intencionalmente.

Abstract

The use of virtual worlds in teaching environments has been a study target due to its prototyping capabilities, as well as its ability to arouse a greater interest from students. Many institutions already tried to incorporate OpenSimulator as a teaching tool, but the management difficulties of this technology and its lack of functionalities for this type of usage is evident.

In this internship a Software Engineering approach was used in order to develop a management tool that enables the management of OpenSimulator in teaching environments. For this effect, the identification of functional requirements was done reviewing the literature about the usage of OpeSim in corporate environments as well as the pedagogical literature with this technology to identify what types of tasks are fulfilled with this technology. Futhermore the system's administrators were interviewed in order to identify the challenges of managing an information system and identify requirements related to sytems management. After the development phase various tests were executed in order to validate the backoffice web, namely unit testing, robustness testing and integration testing as well as security testing.

The product of this internship is a prototype of a tool that eases the management of OpenSimulator in a pedagogical environment. In addition to manage the OpenSim, the backoffice has various complementary functionalities like event management in the virtual world, attendance control and the management of systems' logs.

Keywords

OpenSimulator; OpenSim; virtual worlds; systems management; systems administration.

Esta página foi deixada em branco intencionalmente.

Conteúdo

Lista de Figuras

Esta página foi deixada em branco intencionalmente.

Lista de Tabelas

Capítulo 1

Introdução

O uso de mundos virtuais na educação tem vindo a ser explorado nos últimos anos por várias instituições de ensino [2]. A capacidade de prototipagem destas tecnologias faz com que seja possível criar e testar objectos, teorias e ideias rapidamente e com poucos custos.

O OpenSimulator é uma plataforma *open source* de hospedagem de mundos virtuais que permite controlar o espaço 3D. Esta tecnologia, apesar de ter uma lista de funcionalidades extensa, necessita de alguns ajustes e novas funcionalidades para o seu uso num ambiente pedagógico. Também não fornece aos administradores uma solução para a gestão da instalação que seja intuitiva, levando assim a que tenham um esforço enorme para o fazer. Embora algumas ferramentas de gestão tenham sido criadas para a solução deste problema, estas têm poucas funcionalidades e são bastante antigas.

1.1 Enquandramento do projecto

A Universidade Aberta(UAb) é uma instituição de ensino à distância, disponibilizando o acesso ao ensino superior a milhares de alunos por todo o mundo. Sendo uma instituição baseada em ensino à distância e *elearning* gostaria de ter uma ferramenta de ensino mais apelativa e que facilitasse o ensino de alguns conceitos que são muito difíceis de avaliar à distância, como por exemplo a avaliação de um aluno no que toca à montagem de uma exposição de arte. Através do uso do mundo virtual os alunos podem modificar a exposição, mudando a localização das peças de arte, permitindo assim a avaliação totalmente virtual de algo que, no mundo real, seria muito complicado de fazer.

1.2 Descrição do problema

O OpenSimulator disponibiliza uma ferramenta de gestão para os administradores de sistemas que é pobre em termos de usabilidade. Trata-se de um conjunto de comandos que podem ser introduzidos directamente no terminal do servidor. Esta

ferramenta não facilita a gestão desta tecnologia uma vez que, normalmente, o uso é esporádico em ambientes académicos, o que também faz com que não seja viável ter um administrador de sistemas responsável por esta tecnologia a tempo inteiro.

Várias ferramentas de gestão *web* foram criadas para facilitar a gestão de uma instalação OpenSim, mas, apesar do vasto uso pedagógico, nenhuma foi concebida com o intuito de satisfazer as necessidades deste tipo de uso do OpenSimulator. Este tipo de uso necessita de funcionalidades bastante específicas, como o controlo de presenças em sessões, a necessidade de haver sessões privadas, etc, assim como a possível alteração da própria tecnologia OpenSim de modo a suportar as novas funcionalidades indispensáveis ao uso do OpenSim como ferramenta de aprendizagem.

Estas soluções de administração não são viáveis para a disponibilização do OpenSimulator numa instituição contituida por centenas de docentes e milhares de alunos.

1.3 Objectivos

Como referido, o OpenSimulator não apresenta nenhuma ferramenta de administração que satisfaça as necessidades de gestão do uso desta tecnologia em ambiente pedagógico, sendo que também o OpenSim não tem as funcionalidades requeridas para uso pedagógico. Desta forma é esperado que neste estágio seja desenvolvido um produto que facilite a adopção da tecnologia OpenSimulator em ambientes pedagógicos como ferramenta de ensino virtual. Nomeadamente é esperado o desenvolvimento de um *backoffice* que permita a gestão da tecnologia OpenSimulator através de uma interface *web* em detrimento do uso do terminal e de uma lista de comandos ou até mesmo alterações directas à base de dados. De modo a completar e suportar as funcionalidades do produto mencionado poderá ser também necessário alterar o OpenSimulator.

O objectivo deste estágio é o desenvolvimento de um *backoffice* que permite aos administradores de sistemas da Universidade Aberta uma fácil gestão desta tecnologia, permitindo assim a adopção desta tecnologia em ambiente pedagógico. Para atingir este objectivo foram identificados requisitos relativos à administração de sistemas, como gestão dos logs do OpenSim e um sistema de suporte, de forma a que esta tecnologia possa ser disponibilizada como ferramenta pedagógica a milhares de utilizadores. Outro objectivo deste estágio será a identificação e implementação de requisitos não presentes no OpenSimulator e que são necessários para o uso desta tecnologia para o ensino e avaliação de alunos de uma determinada unidade curricular. Para a identificação destes requisitos recorri a uma revisão da literatura referente ao uso do OpenSimulator em ambientes de aprendizagem.

1.4 Contribuições

Ao longo deste projecto foram identificados e implementados alguns requisitos únicos da solução desenvolvida, contribuindo assim para a mais fácil adopção do OpenSimulator em ambientes de ensino. Para o uso do OpenSimulator como ferramenta

pedagógica foram identificados requisitos importantes para a adopção ser possível. Foi implementada uma forma de guardar e carregar regiões de forma a que seja bastante simples a criação de regiões iguais a regiões que já tenham sido criadas. A gestão de eventos inerente a este *backoffice* também é única pois este permite a criação e terminação de eventos no mundo virtual de forma automática não necessitando assim que o administrador de sistemas tenha de gerir as autorizações de acesso às regiões no momento do começo do evento. Associada a esta gestão de eventos foi também implementada uma forma de controlar as presenças nos eventos do sistema, permitindo assim o ensino e avaliação de alunos no mundo virtual sem perder a capacidade de saber que alunos estavam presentes na sessão. Por último, foi implementado um controlador de estado dos simuladores de forma a saber que regiões este deve iniciar quando o processo tem início. Esta funcionalidade foi decidida ao longo do desenvolvimento após perceber que os simuladores ou não iniciavam as regiões correctas ou não iniciavam qualquer região, mesmo que esta tivesse sido criado anteriormente.

1.5 Estrutura

Para além da introdução, este documento está dividido em mais 7 capítulos:

1. Metodologia, que contém os passos a percorrer para o desenvolvimento do produto;
2. Contexto, onde são apresentados os sistemas de informação da UAb com os quais a solução terá de comunicar;
3. Estado da arte, onde é apresentada uma avaliação das ferramentas de gestão actuais para o OpenSim assim como uma revisão da literatura pedagógica do OpenSim;
4. Requisitos funcionais, que contém os requisitos funcionais identificados que devem estar presentes no produto final;
5. Atributos de qualidade e arquitectura, que consiste na exposição das qualidades pertinentes para o sistema, a sua arquitectura e a apresentação das tecnologias a utilizar;
6. Implementação, que contém os detalhes de desenvolvimento do projecto assim como as várias páginas *web* desenvolvidas;
7. Testes, onde são apresentados todos os testes efectuados ao sistema e o seu resultado;
8. Planeamento e riscos, onde o planeamento do projecto é exposto e os riscos associados a este apresentados;
9. Conclusões, capítulo em que é realizada uma reflexão do trabalho realizado no primeiro semestre assim como é apresentado o trabalho realizado no segundo semestre.

Esta página foi deixada em branco intencionalmente.

Capítulo 2

Contexto

2.1 Sistemas de informação da Universidade Aberta

Tal como descrito na secção 1 e aprofundado na secção 4 e 5, o *backoffice* a desenvolver terá de interagir com vários sistemas de informação já presentes na Universidade Aberta. Desta forma, é necessário detalhar como funcionam estes sistemas. O funcionamento destes foi identificado através da entrevista aos administradores de sistemas, tal como referido na secção 3.

A Universidade Aberta, sendo uma universidade de ensino à distância, fornece aos seus alunos e docentes uma plataforma de *elearning*. A arquitectura desta plataforma, como podemos ver na figura 2.1, é constituida por 6 máquinas: 4 responsáveis pelo *frontend*, 1 responsável pelo *backend* e ainda uma máquina responsável apenas pela gestão de notificações. A distribuição de carga pelas 4 máquinas responsáveis pelo *frontend* é assegurada por um balanceador de carga implementado em *software* que distribui uniformemente a carga. Os materiais alojados nesta plataforma são disponibilizados através do Moodle [3].

A UAb dispõe ainda de mais sistemas de informação, como seria de esperar. Todos estes têm a mesma forma de autenticação, incluindo a plataforma de elearning. Existe um servidor de autenticação onde todos os acessos são autenticados e autorizados de modo a permitir que o utilizador se autentique em todas as aplicações com apenas uma autenticação (*single sign-on*). Este servidor segue o protocolo *Central Authentication Service* (CAS). É este protocolo que permite um *login* único para todos os sistemas de informação da UAb. Quando um utilizador pede informação a algum dos sistemas de informação da UAb, caso não esteja autenticado, é retornado um formulário de *login*. Caso as credenciais estejam correctas, o utilizador é autenticado e o servidor envia um *token* de sessão que é guardado no *browser* do utilizador. Este *token* é uma sessão *single sign-on*, podendo então o utilizador aceder aos outros sistemas de informação sem necessitar de realizar outro *login* [4]. Os dados de autenticação estão guardados num serviço que guarda e controla o acesso a estes dados chamado Active Directory. Este serviço suporta LDAP (*Lightweight Directory Access Protocol*), um protocolo para aceder aos conteúdos de directórios. O servidor de autenticação suporta ainda a autenticação através apenas do protocolo LDAP ou LDAPS (*Lightweight Directory Access Protocol Over Secure Socket*

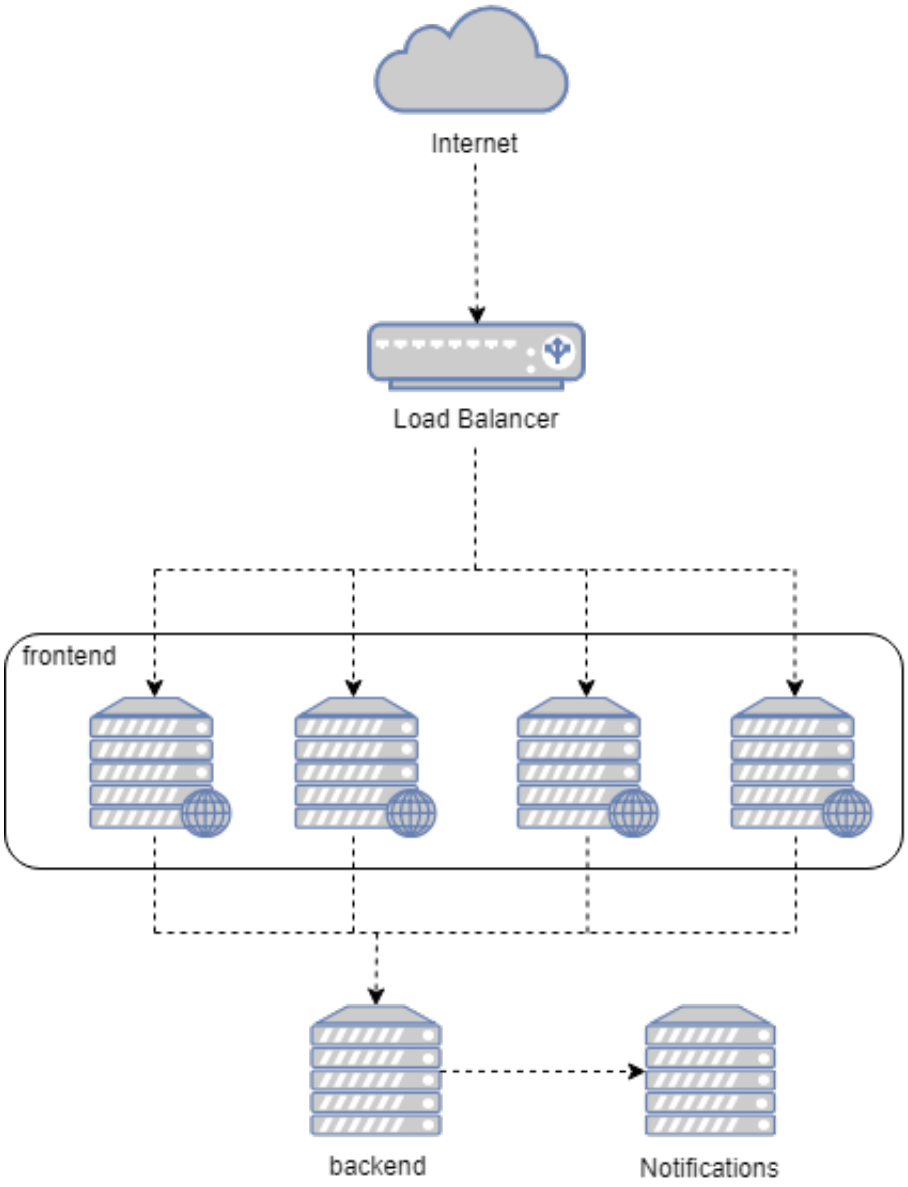

Figura 2.1: Arquitectura da plataforma de *elearning* da UAb.

Layer), sendo a maior desvantagem de utilizar estes protocolos sem CAS a impossibilidade de *single sign-on* para aceder aos outros sistemas de informação. Para um cliente se autenticar utilizando o primeiro método, LDAP, é necessário conectar-se ao servidor, operação normalmente denominada de *bind*. Para o fazer o cliente necessita de enviar uma palavra chave e um *directory name*. Caso a palavra chave corresponda à palavra chave do directório, este cliente é autenticado e gerado um *token* de acesso [5]. Utilizando o segundo método o processo é um pouco mais complicado pois envolve a utilização de SSL (*Secure Sockets Layer*). Neste método o protocolo SSL é utilizado para encriptar as mensagens trocadas entre o cliente e o servidor LDAP. Para este processo de encriptação podem ser utilizados certificados de forma ao cliente e servidor confirmarem a identidade um do outro ou apenas as cifras utilizadas para encriptar os dados [6].

O correio eléctronico na UAb é assegurado por um servidor Microsoft Exchange presente nas instalações. A máquina responsável pelas notificações da plataforma

de *elearning* envia notificações em formato de email sendo estes alojados no servidor referido.

2.2 OpenSimulator

O OpenSimulator[7] é uma tecnologia *open source* que permite a criação e manutenção de mundos virtuais. Esta tecnologia partilha imensas semelhanças com o Second Life[1], tendo utilizado os mesmos protocolos de comunicação e até tendo o mesmo cliente (habitualmente chamado *"viewer"*) durante bastante tempo.

De forma a aceder ao mundo virtual proporcionado pela tecnologia OpenSimulator, o utilizador necessita em primeiro lugar de configurar o seu *viewer* de modo a que aceda ao mundo virtual pretendido, necessitando de identificar o seu endereço IP e a porta de acesso. Após este passo, um utilizador, com o auxílio de um *viewer* e depois do processo de registo e autenticação, tem acesso a uma região, isto é, a um espaço 3D virtual com normalmente 256 metros por 256 metros (uma vez que a maior parte dos *viewers* suporta no máximo esta área) que é parte de um mundo virtual. Os utilizadores interagem com este mundo virtual através de uma representação virtual sua, denominada avatar. O espaço virtual pode ter personagens controladas pelo próprio simulador (geralmente denominadas NPCs), objectos e partículas. Um objecto é um conjunto de formas geométricas, conhecidas como primitivas, que pode conter scripts e que pertence ao mundo virtual. Uma partícula é um elemento visual reproduzido por um *script* que tem um tempo de vida pequeno e que não pertence ao mundo virtual, como gotas de água. Neste espaço o utilizador pode interagir com os objectos do ambiente, tendo total liberdade dentro do mundo virtual, podendo voar, teleportar-se para outra localização (definida por coordenadas no plano xOy) e andar ou correr ao longo da região.

2.2.1 Arquitectura

O OpenSimulator tem 3 opções de instalação: *"standalone"*, *"grid"* e *"hypergrid"*. Nas instalações standalone apenas um processo controla todos os serviços necessários ao funcionamento do simulador logo apenas uma máquina é utilizada para manter o servidor OpenSimulator (fig. 2.2). Este tipo de opção deve ser utilizada por servidores com baixa afluência pois apenas uma máquina pode ser utilizada. *"Grid"* é um tipo de instalação em que os serviços estão disponíveis num processo e os simuladores estão disponíveis noutros processos (fig. 2.3). Assim, permite que a instalação do OpenSim seja dividida entre várias máquinas, melhorando a *performance* e permitindo um maior número de utilizadores simultâneos sem haver disrupção do serviço. *"Hypergrid"* é um tipo de instalação que permite ligar o nosso servidor a um outro servidor disponível na Internet.

Como se pode ver na figura 2.3, para ter acesso aos mundos virtuais proporcionados por uma instalação OpenSim, o utilizador necessita de um *viewer* que irá efectuar uma ligação ao servidor e utilizar os serviços por este disponibilizados. Também se

[1] www.secondlife.com

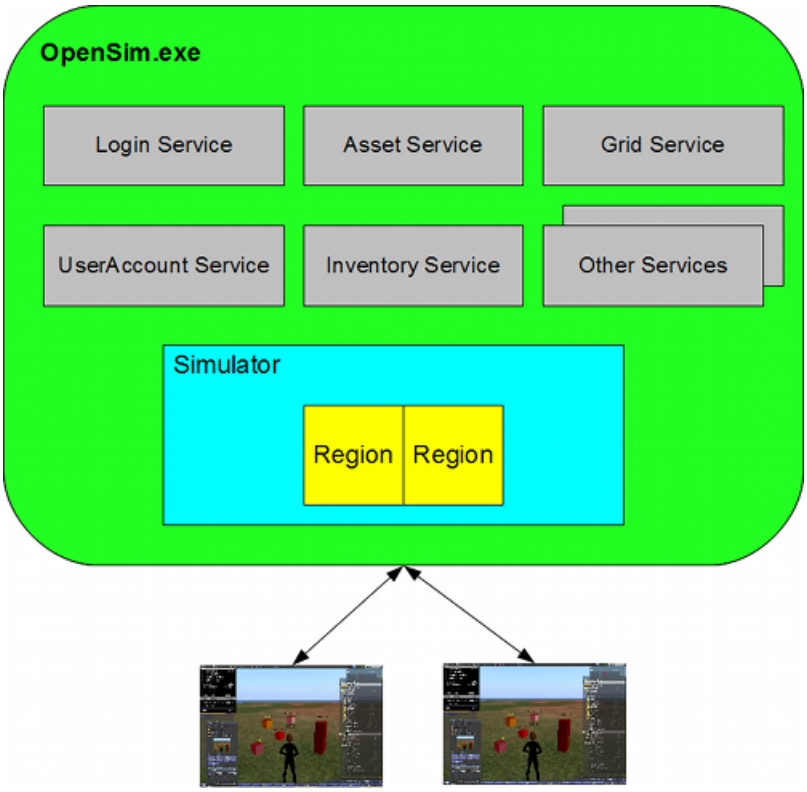

Figura 2.2: Arquitectura do OpenSimulator no modo *standalone*.

Fonte: `http://opensimulator.org/wiki/File:Opensim-standalone.png`

pode verificar que um servidor OpenSimulator é constituido por vários componentes. Um componente que tem todos os serviços necessários ao normal funcionamento da tecnologia e por um ou mais simuladores que podem ter uma ou mais regiões. De facto, podemos retratar os simuladores como servidores, responsáveis por gerir uma ou mais regiões, que comunicam com um servidor central que tem as informações necessárias para a disponibilização do serviço.

Em termos de persistência de dados, o OpenSimulator tem suporte para dois sistemas de gestão de dados, MySql e SQLite. Um destes dois sistemas pode ser utilizado para guardar dados em relação aos serviços e às regiões. No modo *standalone* apenas uma base de dados é utilizada pois toda a funcionalidade da instalação OpenSim está apenas numa máquina. Porém, numa instalação *grid*, toda a informação referente aos serviços é guardada numa base de dados, enquanto toda a informação referente a um simulador é guardada noutra devido à flexibilidade de esta instalação poder ser dividida em diferentes máquinas.

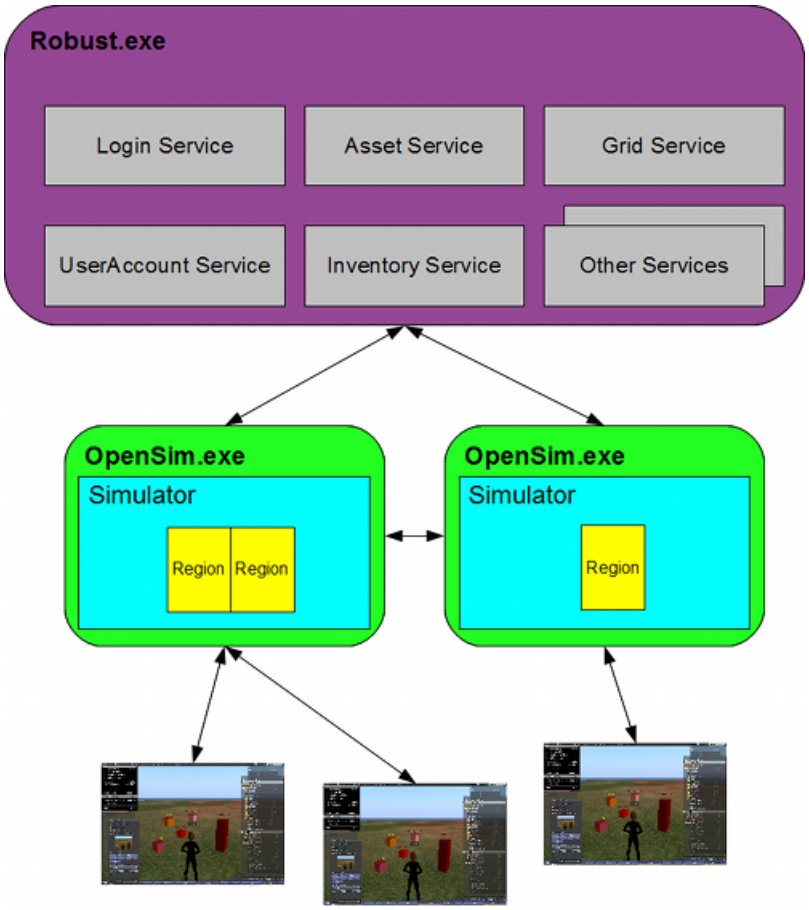

Figura 2.3: Arquitectura do OpenSimulator no modo *grid*.

Source: `http://opensimulator.org/wiki/File:Opensim-grid-simple.png`

2.2.2 Funcionamento

O OpenSimulator funciona de dois modos diferentes, dependendo do tipo de instalação efectuada. Se for do tipo *standalone*, segue um funcionamento servidor-cliente. O *viewer* (cliente) envia pedidos ao servidor aos quais este responde. Se for do tipo *grid* em vez de dois pontos de comunicação podemos ter mais, tal como explicado na secção 2.2.1. Neste caso a comunicação é bastante diferente. O *viewer* necessitará de realizar os pedidos ao simulador que tem a região à qual este se está a ligar. Depois terá de realizar um pedido à máquina onde estão a ser disponibilizados os serviços de modo a obter os dados necessários para o seu funcionamento.

Serviços

Tal como referido na secção 2.2.1, o OpenSimulator disponibiliza os serviços através de um processo que pode ser executado numa máquina diferente da(s) do(s) simulador(es). De modo a garantir o correcto funcionamento dos mundos virtuais, o OpenSimulator disponibiliza e utiliza vários serviços. Na tabela 2.1 estão presentes os ditos serviços assim como uma descrição da sua funcionalidade e a que processo

pertencem.

Serviço	Local	Descrição
Asset	Robust	Guarda os recursos (texturas, scripts, etc) e fornece-os ao serem pedidos
Authentication	Robust	Responsável pela autenticação (manutenção dos tokens de acesso e palavras chave)
Authorization	Robust	Verifica se o avatar tem permissão para entrar numa dada região
Avatar	Robust	Guarda os apectos relacionados com os avatares (altura, aparência, etc)
Freeswitch	Robust	Serviço de comunicações por voz[2]
Friends	Robust	Guarda e fornece a lista de amigos dos utilizadores
Grid	Robust	Guarda informação sobre as regiões que pertencem à *"grid"*
GridUser	Robust	Guarda informação de um avatar na *"grid"* (por exempo, última posição, posição base, se está *online* ou não, etc)
Groups	Robust	Guarda informação sobre os grupos de utilizadores (grupos constituidos por 2 ou mais utilizadores)
HyperGrid	Robust	Serviço relativos a instalações *"Hypergrid"* (normalmente substitui alguns serviços das outras instalações)
Inventory	Robust	Guarda a informação sobre os inventários dos utilizadores
Login	Robust	Responsável pelo *login*
MapImage	Robust	Guarda e fornece blocos de imagens do mapa principal
Presence	Robust	Guarda os dados relativos às sessões dos utilizadores *online*
UserAccount	Robust	Gere os dados relativos às contas dos utilizadores (nome do avatar, permissões do utilizador, etc)
Land	Simulador	Fornece dados relativos ao terreno de um simulador
Library	Simulador	Guarda e disponibiliza recursos organizados por pastas
Simulation	Simulador	Funções gerais de simulação (criar agente, criar objecto, teleportar, etc)

Tabela 2.1: Descrição dos serviços do OpenSimulator.

Comunicação *Viewer*-Servidor

Os protocolos de comunicação utilizados para a comunicação variam de serviço para serviço. Devido à pobre documentação desta tecnologia apenas foi encontrada informação sobre como é feita a comunicação ao aceder ao serviço *Asset*.

De modo a criar a imagem do terreno ao qual o utilizador está a aceder e actualizar o estado do avatar, após o *login*, o *viewer* necessita de aceder aos serviços descritos na tabela 2.1. Um destes é o serviço *Asset*, que é responsável por fornecer texturas, scripts, etc ao serem pedidos. Na figura 2.4 podemos ver um esquema que resume como é realizada a comunicação.

Para obter informações sobre as texturas, por exemplo sobre a textura base da zona em que o avatar está presente, o *viewer* necessita de invocar o serviço *Asset*. O simulador, para iniciar a invocação do serviço, envia uma actualização de estado

[2] freeswitch.org

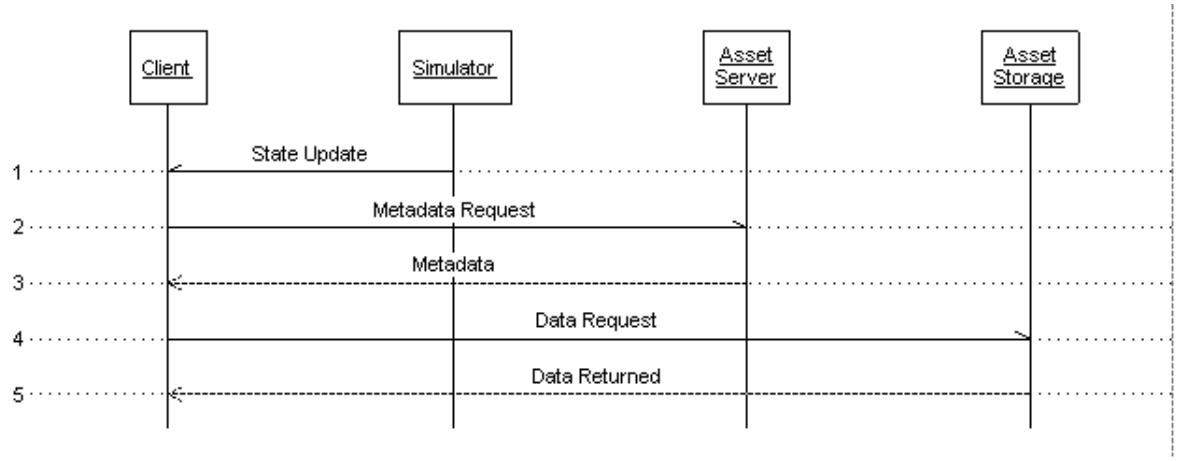

Figura 2.4: Exemplo de comunicação entre o *viewer* e um serviço.

Source:`http://opensimulator.org/images/d/d7/Client-sim-assetserver-%distributed.png`

ao *viewer*, por exemplo, devido a ter andado no espaço virtual. Este serviço tem uma particularidade relacionada com a não criação de *overhead* num só pedido. Em primeiro lugar, o *viewer* realiza um pedido ao serviço referido para obter os metadados dos recursos provenientes da actualização de estado. Só de seguida, e conforme a utilidade do recurso, o *viewer* irá optar ou não por realizar outro(s) pedido(s) para descarregar o(s) recurso(s). Esta forma de comunicação reduz drasticamente o *overhead* de um pedido devido ao serviço não devolver num único pedido todos os recursos sobre os quais o *viewer* pediu os metadados.

Gestão do OpenSimulator

De forma a gerir a instalação OpenSim, existem comandos de administração que podem ser introduzidos no terminal que está a executar o processo. Estes podem ser de dois tipos, os que estão relacionados com cada simulador e os que estão relacionados com os serviços *grid*. Numa instalação standalone, ambos os tipos são inseridos na mesma consola pois existe apenas um processo que é responsável pelo OpenSim. numa instalação *grid* os comandos que estão relacionados com os serviços, como criar um utilizador, devem ser introduzidos no terminal onde está a executar o processo ROBUST e os comandos relativos aos simuladores devem ser executados na consola do simulador onde se deseja realizar a alteração. A lista destes comandos varia consoante a versão do OpenSimulator e a melhor forma de verificar quais estão disponíveis é através do comando *"help"*.

Gerir uma instalação OpenSim através desta ferramenta poderá ser algo dispendioso se o sistema for complexo pois o administrador deste necessita de saber que comando deve utilizar para realizar uma operação e quais os parâmetros necessários (lista completa destes comandos em [8]). Porém, quase todas as funcionalidades necessárias para a manutenção do sistema estão presentes nesta ferramenta, o que indica que para um utilizador avançado, que conheça bem os comandos e a tecnologia OpenSim, pode ser uma boa ferramenta de manutenção.

Para além desta ferramenta de gestão existe também uma abordagem mais complexa. Esta consiste em alterar as bases de dados que servem de suporte ao OpenSim manualmente. Apesar de ser possível alterar quase todos os aspectos do OpenSim desta forma, isto requer um conhecimento muito profundo do sistema e, por isso, não é uma abordagem de manutenção desejada.

Capítulo 3

Metodologia

Tal como afirmado no capítulo 1, o objectivo deste estágio curricular é o desenvolvimento de um *backoffice web* para o OpenSimulator. Para atingir esta meta, é necessário completar um conjunto de passos:

1. Estudar e compreender a tecnologia OpenSimulator (subcapítulo 2.2.1);
2. Perceber o que já foi feito em termos de ferramentas de gestão para o OpenSimulator assim como estudar tentativas de implementação do OpenSimulator como ferramenta de aprendizagem (capítulo 4);
3. Obter e especificar os requisitos funcionais que deverão estar presentes na ferramenta a desenvolver, assim como restrições técnicas e de negócio que deverão ser respeitadas (subcapítulo 2.1 e capítulo 5);
4. Perceber que atributos de qualidades são importantes e esboçar a arquitectura da solução (capítulo 6);
5. Desenvolver *mockups* que devem ser validados pelos futuros utilizadores do sistema;
6. Implementar a solução respeitando os passos anteriores (capítulo 7);
7. Testar a solução(capítulo 8).

De modo a cumprir o primeiro passo foi necessário estudar o sistema. Para isto foi necessário instalar o OpenSimulator e perceber como este funciona, assim como ler a documentação presente no *website* desta tecnologia.

O segundo passo foi concluido ao instalar as ferramentas disponíveis para a gestão do OpenSim e realizar um levantamento das funcionalidades que estas fornecem aos utilizadores. Foi também necessária uma revisão da literatura. Esta foi efectuada tendo em conta que era fundamental perceber os usos pedagógicos do OpenSim e que funcionalidades seriam úteis no *backoffice web*.

De modo a concluir o passo 3, foi imprescindível compreender as actividades de um administrador de sistemas uma vez que serão estes os utilizadores finais do sistema a

desenvolver. Também foi crucial perceber como funcionam os sistemas de informação da UAb uma vez que seria necessário interagir com estes. Os requisitos resultantes foram priorizados seguindo o método MoSCoW [9].Para a conclusão deste passo foi necessária uma entrevista semi-estruturada aos administradores de sistemas.

O passo 4 foi fechado quando os atributos de qualidade foram priorizados e foram efectados vários cenários para cada um deles que deverão ser cumpridos pelo sistema final e que deveriam reflectir-se na arquitectura resultante.

Para concluir o passo 5 foi necessário esboçar a *interface web* do sistema e obter a validação por parte dos administradores de sistemas.

O passo 6 passa pela implementação do sistema. Para este passo ser concluido com sucesso foi necessário que todos os requisitos *must have* fossem implementados no sistema. Caso estes fossem implementados e a meta planeada para o desenvolvimento não tivesse sido ultrapassada, os outros requisitos deveriam também ser implementados, seguindo as prioridades destes, até que a data limite fosse atingida.

Para concluir o passo 7 deveriam ser realizados testes ao sistema. Estes testes deveriam ser automatizados no que toca aos testes de robustez e unitários ao sistema. Ao longo da execução deste passo deveriam ser executadas correcções ao sistema provenientes de erros obtidos pelos testes e estes deveriam voltar a ser efectuados.

Capítulo 4

Estado da arte

Neste capítulo serão apresentadas as ferramentas de administração actuais disponíveis para o OpenSimulator assim como uma revisão da literatura de forma a tentar encontrar possíveis requisitos funcionais que deverão estar presentes no sistema a desenvolver. É também apresentada uma comparação entre as tecnologias actuais para a alojamento e simulação de mundos virtuais. Também serão estudadas as tecnologias para alojamento de mundos virtuais de forma a justificar a escolha do *OpenSimulator* como ferramenta de ensino virtual.

4.1 Tecnologias para alojamento de mundos virtuais

Existem várias tecnologias disponíveis para hospedar mundos virtuais. Nesta secção estas serão estudadas de forma a avaliar qual delas é a melhor escolha em termos de ferramenta pedagógica.

4.1.1 *Second Life*

O *Second Life*[1] é uma tecnologia criada pela *Linden Lab*[2]. Muitas instituições utilizam ou utilizaram esta ferramenta como forma de ensino à distância para vários temas como aprendizagem de uma língua. Esta tecnologia é muito parecida com o *OpenSimulator*.

Para o uso desta tecnologia como ferramenta de ensino é necessário comprar terreno virtual o que pode constituir uma desvantagem. A manutenção da região tem de ser efectuada através do cliente (*viewer*)(não suporta todas as funcionalidades de gestão) ou tem de ser efectuado um pedido de ajuda à equipa de suporte do *Second Life*, ficando assim a identidade que compra o terreno dependente da equipa de suporte do *Second Life* em termos de alterações e gestão do mundo virtual.

[1] https://secondlife.com/
[2] https://www.lindenlab.com/

As vantagens desta tecnologia são que quase todos os utilizadores conseguiram ter acesso ao mundo virtual pois existe um cliente para a utilização num computador pessoal que não necessita de muito poder de computação assim como uma aplicação para Android (*Lumiya*[3]) e que toda a administração do sistema é assegurada pela *Linden Lab*.

4.1.2 *OpenSimulator*

O *OpenSimulator* é um *software* parecido com o *Second Life* e é *open-source*, facilitando assim a sua adopção por parte das instituições que podem instalar os seus próprios servidores para o uso desta tecnologia como ferramenta pedagógica. O *OpenSim* permite assim uma gestão completa de todos os recursos virtuais.

Em termos de facilidade de acesso estão também presentes as mesmas premissas da tecnologia *Second Life*. O *viewer* que o utilizador tem de instalar para aceder ao mundo virtual necessita de pouco poder de computação e a mesma aplicação para Android pode ser utilizada.

4.1.3 *Sansar*

Sansar[4] foi também desenvolvido e é propriedade da *Linden Lab*. Esta tecnologia pode ser acedida através de um cliente para computadores pessoais, mas, ao contrário das duas tecnologias vistas anteriormente, é necessário um elevado poder de computaçao e de renderização para aceder ao mundo virtual. O *Sansar* também é compatível com equipamentos de realidade virtual.

A obtenção e gestão do espaço virtual são iguais ao *Second Life* e portanto a liberdade de gestão das regiões é limitada.

4.1.4 *High Fidelity*

High Fidelity[5] é uma tecnologia *open-source* que permite a utilização de equipamentos de realidade virtual para interagir com o mundo virtual. À semelhança do *Sansar* esta tecnologia também necessita de um elevado poder de computação e de renderização. *High Fidelity* também permite o acesso através de uma aplicação Android.

Sendo *open-source* esta tecnologia permite a instalação de servidores e, assim sendo, permite a gestão de todos os aspectos do mundo virtual.

[3]https://play.google.com/store/apps/details?id=com.lumiyaviewer.lumiya
[4]https://www.sansar.com
[5]https://highfidelity.com

4.1.5 *Unity 3D*

Unity 3D[6] é um motor de jogos criado pela *Unity Technologies* e lançado em 2005. Esta tecnologia tem ferramentas rápidas e intuitivas para a criação de ambientes 2D e 3D. Apesar de não ser desenvolvida para a alojamento de mundos virtuais esta ferramenta permite a criação de ambientes 3D que podem ser utilizados como ferramenta pedagógica.

4.1.6 Comparação destas tecnologias

Todas as cinco tecnologias têm vantagens e desvantagens. O *Second Life* permite que uma instituição não tenha de alocar recursos localmente para instalar a ferramenta, mas necessita da compra e aluguer de terreno virtual para utilizar esta tecnologia. Também não permite total controlo sobre a instalação uma vez que a gestão tem de ser efectuada através do *viewer*. O uso de telemóveis para aceder ao mundo virtual é uma vantagem do *Second Life* assim como do *OpenSimulator* e do *High Fidelity*, apesar do último ser apenas compatíveis com telemóveis topo de gama o que dificulta o acesso ao mundo virtual. O *High Fidelity* e o *Sansar* necessitam um elevado poder de computação e de renderização para aceder ao mundo virtual dificultando o acesso ao mundo virtual. O *OpenSim* permite a instalação de servidores locais permitindo assim gerir toda a instalação *OpenSim*. Permite também o acesso ao mundo virtual através de um *viewer* que necessita de pouco poder de computação assim como através de uma aplicação Android o que facilita o acesso ao espaço virtual por partes de todas as pessoas que pretendem integrar este.

O Unity 3D permite o desenvolvimento de um ambiente 3D multi-utilizador para uso pedagógico. Assim sendo, esta ferramenta pode ser útil para a execução de formações rápidas e temporárias. Caso haja duas formações ao mesmo tempo não é possível a interacção entre as duas aplicações. Desta forma esta tecnologia não é uma boa opção para uso pedagógico pois é necessário o desenvolvimento de aplicações completamente diferentes para sessões de treino diferentes.

O uso de uma tecnologia de alojamento de mundos virtuais como ferramenta pedagógica necessita de facilitar o acesso dos utilizadores ao mundo virtual assim como permitir a gestão do mundo virtual. O *OpenSimulator* é a tecnologia que permite a maior acessibilidade assim como a gestão total do mundo virtual. Aliado a estes factos o *OpenSim* também não necessita de comprar e alugar terreno virtual apesar de necessitar de alocação de recursos locais, mas ao contrário do aluguer de terreno é possível realocar os recursos de forma a gerir eficientemente os recursos da instituição.

4.2 Ferramentas de administração actuais

Como referido na secção 2.2.2, gerir uma instalação OpenSimulator complexa pode ser um enorme problema para as pessoas resposáveis por fazê-lo. Para um admi-

[6] https://unity3d.com/

nistrador de sistemas de uma instituição educativa o sistema será apenas mais uma tecnologia entre tantas outras que tem de ser dominada. E se o uso não for generalizado dentro da instituição, o serviço cairá fora de rotinas diárias, fazendo com que o administrador do sistema tenha de rever os procedimentos e refrescar conhecimento acerca dos esquemas de base de dados e dos protocolos de comunicação cliente-servidor. Para facilitar esta tarefa foram desenvolvidas e disponibilizadas algumas ferramentas de gestão do OpenSim. Para realizar a avaliação destas ferramentas deparámo-nos com um problema de descontinuação: a maioria das soluções existentes forem desenvolvidas em tecnologias que estão descontinuadas, obsoletas, ou foram desenvolvidas numa versão bastante desactualizada destas, uma vez que foram desenvolvidas há até 9 anos atrás. Conseguimos instalar 3 das ferramentas presentes no website do OpenSimulator [10] e uma disponibilizada pelo MyOpenGrid[7]. A avalização das funcionalidades foi efectuada lendo as descrições das ferramentas e documentação (quando esta estava disponível, como no caso do jOpenSim) e instalando e analisando a interface *web* de cada uma destas depois de configuradas para estarem integradas na nossa instalação OpenSim.

De seguida as ferramentas e a descrição de cada uma, seguido de uma análise das funcionalidades disponíveis em cada uma. De realçar que não conseguimos instalar algumas das soluções disponíveis em [10] pelas razões referidas anteriormente. Tentámos contactar os criadores das referidas ferramentas, mas não obtivemos suporte.

A informação apresentada nesta sub-secção está presente no artigo presente no anexo C e que foi submitido e aceite na conferência Videojogos 2018[8] estando as atas no prelo de momento.[11]

4.2.1 Remote Admin

De modo a aceder ao Opensim remotamente, esta tecnologia fornece o *endpoint* Remote Admin para o protocolo XML-RPC.

Além de fornecer aos administradores de sistemas um modo de acesso remoto aos serviços OpenSim especificamente (sem ter que abrir um terminal remoto com acesso à máquina inteira) o Remote Admin não fornece nenhum suporte, dado que a sua operação é idêntica às tarefas de administração locais.

4.2.2 OpenSimulator Manager Web

OpenSimulator Manager Web, ou apenas OSMW, é uma ferramenta de administração escrita em PHP que fornece aos administradores de instalações OpenSim uma interface *web* funcionalidades de gestão de utilizadores, exportar alguns dados, monitorizar e fazer o *backup* de regiões, e alguma gestão de ficheiros de configuração.. Esta ferramenta foi desenvolvida há 3 anos atrás e algum do código necessita de ser reconstruido devido às mudanças ocorridas na linguagem PHP desde a versão 5.5.29

[7] http://myopengrid.com/page/home
[8] https://vj2018.fba.up.pt/

até à versão 7.1.14 que descontinuaram algumas funções e mudaram alguma da semântica desta linguagem.

4.2.3 MWI

MWI é uma interface *web* para OpenSim, também intencionada a ser um sistema de gestão de conteúdos completo, com todas as funcionalidades habituais deste tipo de ferramenta. Tem duas partes principais: um *front-end* para visitantes e utilizadores e uma área de administração para os gestores de *grids*, onde estes podem mudar as características do sistema. Fornece aos administradores de sistemas gestão de contas, gestão de grupos de utilizadores, e mais algumas funcionalidades (tabela 4.1). Também é possível automatizar algumas tarefas, como o registo de utilizadores e recuperação de palavra chave, e aceder facilmente informações sobre as regiões da *grid*.

4.2.4 Páginas WiFi do OpenSimulator

WiFi é um sistema constituido por um conjunto de componentes que podem ser usados tanto em simuladores *standalone* como em instalações *grid* para gerir o OpenSim. Concretamente, as suas contas, *updates*, palavras chave e gestão do inventário dos utilzadores. É dito que as suas propriedades fazem com que seja uma boa opção para a gestão de instalações OpenSim pequenas e médias [12]. Não é necessária a instalação de um servidor *web* uma vez que apenas utiliza funcionalidades já presentes no OpenSimulator.

4.2.5 jOpenSim Web Interface

jOpenSim é uma extensão para Joomla! que permite a interacção com o OpenSimulator e 2 módulos (*grid status* e *friends online*). Este componente pode interagir com instalações *standalone* ou *grid* com MySQL como motor de base de dados [10, 13]. Uma vez que é uma extensão para Joomla!, o administrador pode criar um *website* que preencha as suas necessidades e gostos usando funcionalidades disponibilizadas pelos componentes e módulos fornecidos pelo jOpenSim. Fornece gestão de utilizadores, gestão de grupos de utilizadores, gestão de eventos, etc. (tabela 4.1).

4.2.6 Comparação das ferramentas de administração

Como visto na tabela 4.1, três tipos de funcionalidades emergem: as que consideramos básicas (funcionalidades presentes em quase todas as ferramentas estudadas), aquelas que são específicas a um tópico específico, tais como gestão de regiões, gestão de contas, ou funcionalidades relacionadas com a configuração e administração, e, por último, funcionalidades que apenas estão disponíveis numa determinada solução.

Algumas ferramentas disponibilizam uma variedade muito maior do que outras. Comparando as páginas WiFi do OpenSim com as outras ferramentas administra-

tivas podemos ver que é a única que permite importar inventário, mas que é a ferramenta com o menor número de funcionalidades.

As páginas WiFi são uma ferramenta de administração bastante limitada uma vez que apenas têm funcionalidades relacionadas com a gestão de contas de utilizadores.

O Remote Admin, como mencionado anteriormente, simplesmente permite aos administradores gerir a instalação OpenSIm sem ter de abrir uma ligação remota à máquina inteira (i.e., SSH, Remote Desktop). Mas é a única forma de aceder um conjunto de funcionalidades de gestão de regiões remotamente, que nenhuma outra ferramenta fornece, expondo assim a falta de cobertura das funcionalidades das outras ferramentas.

O OSMW, juntamente com o jOpenSim, é uma das ferramentas com mais funcionalidades. O primeiro tem o conjunto básico de funcionalidades presente em todas as ferramentas e foca-se na manutenção da instalação. Tem algumas funcionalidades únicas como gerir NPCs, guardar zonas, gerir os logs do sistema e enviar mensagens para todas as regiões. O jOpenSim é a única ferramenta que permite a criação de regiões, embora as outras funcionalidades relacionadas com a gestão de regiões não são suportadas. Foi criada para gerir uma hypergrid, isto é, onde cada região tem o seu próprio administrador. Também fornece as funcionalidades básicas assim como algumas funcionalidades únicas como listar as transações monetárias virtuais, gestão de eventos e uma procura no mundo virtual que permite aos gestores encontrar items presentes no mundo virtual.

MWI é uma das duas ferramentas analizadas que tem a capacidade de gerir grupos de utilizadores. Esta funcionalidade pode não ser importante para instalações OpenSim pequenas, mas para *grids* mais complexas é muito importante, uma vez que os grupos podem ser criados por um conjunto de 2 ou mais avatares. As outras funcionalidades são semelhantes às fornecidas pelas páginas WiFi e, por isso, podemos dizer que esta é a única vantagem do MWI em relação à última ferramenta.

A única ferramenta que fornece a capacidade de autorizar, ou desautorizar, o acesso a certas partes do mundo virtual por parte dos utilizadores através de uma interface gráfica é a MWI. Esta funcionalidade pode ser importante se existirem sessões virtuais a ocorrer em qualquer parte do espaço virtual.

Através de uma análise mais completa da tabela 4.1, podemos verificar que a ferramenta mais adequada para um administrador varia com o ambiente computacional e com o ambiente de uso da instalação OpenSim. Por exemplo, se o OpenSim for apenas utilizado para estudar uma forma de socialização virtual, onde poucas pessoas se ligam ao servidor, talvez as únicas funcionalidades necessárias sejam criar, editar e remover utilizadores do mundo virtual. Se assim for, as páginas WiFi são suficientes para gerir o sistema. Por outro lado, se o OpenSim for utilizador para fins de treino e formação, é provável que nenhuma das ferramentas de administração exploradas neste artigo forneçam as funcionalidades necessárias e suficientes, requerendo a execução de algum trabalho de gestão na consola.

	Remote Admin	Manager Web	MWI	Wifi Pages	jOpenSim
Acesso remoto ao OpenSim	yes	yes	yes	yes	yes
Criar utilizadores	command line	yes	yes	yes	yes
Editar utilizadores	command line	yes	yes	yes	yes
Definir avatares pré-definidos				yes	yes
Exportar inventário		yes			
Importar inventário				yes	
Gerir grupos de utilizadores			yes		yes
Ver o estados de regiões	command line	yes	yes	yes	yes
Ver mapa		yes			yes
Criar região	command line				yes
Editar regiões	command line	yes			
Reiniciar regiões	command line				
Obter lista de acesso de uma região	command line				
Adicionar utilizadores à lista de acesso da região	command line				
Remover utilizadores da lista de acesso da região	command line				
Guardar região	command line	yes			
Carregar região	command line				
Gestão de NPCs		yes			
Guardar zona		yes			
Gestão de ficheiros		yes			
Gestão de logs		yes			
Editar ficheiros de configuração		yes			
Editar tabela de base de dados dos simuladores		yes			
Definir localização *home*			yes		yes
Enviar mensagem a todas as regiões		yes			yes
Procura de objectos no mundo virtual					yes
Definir região base			yes	yes	yes
Gestão de eventos					yes
Listar transacções de dinheiro					yes
Gestão de terminais					yes
Obter versão do OpenSim	command line				yes
Teleportar agentes	command line				

Tabela 4.1: Funcionalidades disponibilizadas pelas ferramentas de gestão.

4.3 Revisão da literatura

De modo a tentar levantar e perceber os requisitos que devem estar presentes na ferramenta de administração de uma instalação OpenSimulator foi realizada uma pesquisa de artigos relacionados com a especificação de uma instalação OpenSim para ambientes organizacionais. Segue um resumo e análise aos dois artigos encontrados neste contexto.

Em "Privacy challenges and methods for virtual classrooms in Second Life Grid and OpenSimulator"[14], os autores descrevem os problemas de privacidade presentes no OpenSimulator e no Second Life, sugerindo também soluções para estes. Os problemas identificados são divididos em 5 categorias: presença do avatar, comunicação por voz, chat de texto, objectos e partículas e controlo da câmara. Em termos de presença do avatar, o OpenSimulator não tem nenhuma forma de garantir a presença de apenas avatares autorizados numa zona em que estão a decorrer actividades que requerem privacidade. Como existem 3 modos de navegar pelo mundo virtual no OpenSimulator (voar, andar e teleportar) e sendo estas controladas pelo dono do avatar, podendo este teleportar-se para qualquer lugar, os autores sugerem que, de forma a garantir a privacidade da sessão, se crie um objecto com um script que verifica se os avatares presentes na sala estão autorizados. Este script ejectaria os

avatares não desejados para uma zona de recepção, onde não conseguiriam espiar as comunicações por texto nem por voz. Os problemas relacionados com a comunicação por voz, usando OpenSimulator, podem ser resolvidos instalando um módulo de comunicação por voz que permita isolar as comunicações, fazendo com que estas apenas estejam disponíveis a membros que pertencem à sessão. A possível falta de privacidade da comunicação por chat de texto prende-se com o facto de ser possível ver mensagens que sejam enviadas num raio de 20 metros, de um dado avatar, caso sejam enviadas normalmente, ou num raio de 96 metros caso seja utilizada a funcionalidade "gritar"do OpenSimulator. A sugestão dada para resolver este problema é o distanciamento entre as zonas que requerem privacidade e as restantes ser de pelo menos 96 metros uma vez que, mesmo que apenas se possa ter acesso ao chat de texto com o avatar a 20 metros da zona de treino, é possível perturbar as sessões de treino a até 96 metros uma vez que o utilizador do avatar com esse objectivo pode "gritar", incomodando assim o chat dos utilizadores presentes na sessão. Em relação aos objectos e às particulas, o problema advém de ser possível criar scripts e juntá-los a um objecto, podendo assim ter uma ferramenta para enviar toda a informação do que está a acontecer na sessão de treino para utilizadores que não possuam permissão para a obter através de serviços externos com auxílio a pedidos HTTP ou protocolos de transferência de e-mails. Outro problema seria a utilização de objectos com scripts para realizar "spam", fazendo com que fosse praticamente impossível realizar a sessão de treino. A solução apresentada pelos autores é a desactivação da funcionalidade de criação de objectos ou então apenas permitir a criação de objectos por um grupos de utilizadores. Contudo, esta solução não afecta objectos que estejam associados aos próprios avatares, sendo que os autores não descrevém nenhuma sugestão para resolver este problema, apenas sugerindo o uso de avatares pré definidos. Por último, os problemas relacionados com o controlo da câmera que interferem com a privacidade de sessões de treino prendem-se com o facto de o utilizador poder mover a câmera para qualquer sítio que deseje num raio de 512 metros. O utilizador pode assim ver o que está a ocurrer na sessão de treino, bastando para isso mover a câmara até o sítio onde esta decorre e configurando a distância máxima até à qual as particulas são renderizadas. A sugestão de solução deste problema apresentada é a separação das zonas que têm sessões de treino por 512 metros, tanto horizontalmente, como verticalmente, permitindo assim a uma região ter um máximo de 19 zonas de sessões de treino.

Deste artigo podemos retirar alguns requisitos para o *Backoffice*. Deverá ser necessário ter uma lista de avatares autorizados numa zona de uma região onde podem estar a decorrer eventos que exijam privacidade, assim como a possibilidade de criar eventos públicos e privados. Para além disto, e devido aos problemas de privacidade identificados, é necessário controlar também o espaçamento entre as zonas em que estão a decorrer sessões privadas e as outras, de modo a garantir que não existem utilizadores não autorizados com acesso às informações presentes na sessão.

Em "Requirements for the use of virtual worlds in corporate training"[1], os autores descrevem os requisitos necessários para a utilização de mundos virtuais, em concreto de OpenSimulator, num ambiente de treino empresarial. O objectivo deste trabalho era o de especificar os requisitos necessários que permitisse o correcto funcionamento de um ambiente de treino virtual para a formação dos elementos da PT Inovação (actual Altice Labs). Retirei deste trabalho alguns requisitos que acho importantes

para o sistema que irei implementar, presentes na tabela 4.2, justificando a razão pela qual o fiz. De notar que o índice presente na primeira coluna é referente ao indicado no artigo.

Requisito	Descrição	Justificação
C1/R1	Privacidade das sessões de treino	Poderá haver sessões em que apenas utilizadores autorizados possam participar
C2	Gravar e reproduzir o comportamento dos actores e outros elementos	De modo a avaliar algumas sessões de treino e/ou verificar o que correu mal é benéfico poder reproduzir a sessão
R3a	Gravar interações dos utilizadores com ferramentas do mundo virtual	Para verificar as interacções dos utilizadores com o mundo virtual, em sessões de treino ou não, poderá ser útil poder reproduzir as interacções avatar-objecto
R2f-10	Gravar comunicações via audio nas sessões	Útil para reproduzir as comunicações ocorridas na sessão
R2f-9	Gravar comunicações via texto nas sessões	Útil para reproduzir as comunicações ocorridas na sessão
R2f, R2h	Criar, gerir e apagar sessões de treino automaticamente	Poderá ser bastante importante principalmente para sessões calendarizadas pois retira carga horária aos administradores do sistema
R2f-1	Poder calendarizar sessões de treino	Importante para saber que sessões estão ou irão decorrer num determinado intervalo de tempo
R2f-7	Nomes de utilizador do LMS devem ser associados ao OpenSim	Tem de ser implementado visto que é um dos objectivos do sistema
R2f-3	Poder definir os participantes das sessões	De modo a ter sessões privadas é necessário definir os utilizadores autorizados a participar
R2g-1	Poder enviar aos formandos um *link* para aceder ao espaço virtual	Bastante útil para facilitar a entrada no mundo virtual

R2f-6	Autenticação realizada através do nome de utilizador do LMS	Tem de ser implementado visto que é um dos objectivos do sistema
R13a	Avatars associados ao nome real	Bastante importante visto que facilita o reconhecimento dos utilizadores no mundo virtual
R13e	Utilizadores apenas podem mudar a palavra chave através do LMS	Visto a autenticação ser apenas efectuada através das credenciais do LMS é necessário que apenas lá se possa alterar a palavra chave
R2f-4	Tarefas de manutenção realizadas sem preocupações com questões técnicas	De modo a facilitar a manutenção da instalação OpenSim é desejável que os administradores de sistemas não tenham de se preocupar com detalhes técnicos desta tecnologia
R2f-5	Possibilidade de associar uma sessão de treino a um espaço já existente	Facilita o trabalho dos administradores de sistema ao não terem de criar um espaço novo para todas as sessões de treino
R2g-3	Controlar as presenças nas sessões	Pode ser necessário para verificar que utilizadores participaram em sessões que talvez sejam obrigatórias no contexto da Universidade Aberta
R2f-11	Poder dividir a comunicação em subgrupos de participantes	Útil para não haver poluição sonora no espaço virtual para utilizadores próximos a uma sessão e também para resolver alguns problemas de privacidade
R6b	Poder silenciar as comunicações dos participantes	Necessário para evitar a disrupção de sessões por parte de um utilizador

R13b-4	Fornecer aos participantes avatars pré-preparados	Para haver sessões específicas poderá ser necessário que os avatares tenham alguma característica específica (por exemplo, um *script* que realize determinada acção)
R13a	Nomes reais dos avatars devem ser visiveis a formadores e formandos	Necessário para a rápida identificação do utilizador
R13d-1	Poder restaurar a aparência dos avatars	É necessário uma alternativa fácil para o administrador de sistemas ou o próprio utilizador reverter as alterações efectuadas ao avatar
C6	**Acesso aos dados e serviços do LMS no mundo virtual**	Poderá ser útil o acesso a algumas funcionalidades no espaço virtual
R2h-1	Ter acesso aos avisos e notificações do LMS no espaço virtual	Desta forma, os utilizadores são notificados rapidamente se estiverem no mundo virtual
R2h-2	Ter acesso às projecções multimédia do LMS no espaço virtual	Desta forma os utilizadores não necessitam de aceder ao LMS para ver as projecções multimédia
R2h-3	Localização/Espaço para aceder aos conteúdos do LMS no espaço virtual	Poder aceder a conteúdos que são importantes para as sessões em curso
R2h-4	Ter acesso a documentos de texto simples do LMS no espaço virtual	Poder aceder a conteúdos que são importantes para as sessões em curso
R2h-5	Ter acesso aos sumários dos tópicos do LMS no espaço virtual	Útil para sessões que sejam efectuadas no âmbito de uma unidade curricular

Tabela 4.2: Possíveis requisitos para o sistema, provenientes de [1].

Estes dois artigos são insuficientes para o processo de levantamento de requisitos uma vez que não existem requisitos que representem todas as actividades executadas por um administrador de sistemas, como por exemplo a resposta a *tickets* (pedidos de ajuda de utilizadores) [15]. Desta forma, uma pesquisa foi efectuada de modo a

encontrar outros trabalhos que descrevessem as actividades pedagógicas realizadas no OpenSim e, assim, tentar perceber que outras actividades de gestão poderão ter de ser realizadas nesta ferramenta de alojamento de mundos virtuais. Estes possíveis requisitos estão representados na tabela 4.3 com a seguinte estrutura: artigo do qual se origina o requisito, secção que levou à ideia deste requisito, página do artigo e descrição do requisito.

Índice	Origem	Secção	Página	Descrição
1	[2]	2. Advantages and drawbacks of virtual laboratories	3	Gerir objectos
2	[16]	III.B. Social Interactions	2	Gerir grupos de utilizadores
3	[16]	III.C. Localization functions	2	Localizar avatares e objectos
4	[16]	III.E. Integration with Moodle system	3	Interagir com um LMS
5	[16]	III.F. Schedules and events	3	Criar, modificar e eliminar eventos
6	[16]	III.G. Avatar customization	3	Restaurar aparência a um avatar
7	[16]	III.H. Virtual sub-space	3	Guardar zonas do mundo virtual
7	[16]	III.H. Virtual sub-space	3	Replicar zonas privadas ou públicas
8	[16]	III.H. Virtual sub-space	3	Ajuste de zonas privadas ou públicas
9	[17]	1. Management Guidance for Educators	3	Gerir grupos de utilizadores
10	[17]	1.1. User Training for Virtual World Management	4	Sistema de suporte
11	[18]	III.C. CL Techniques	4	Gerir grupos de utilizadores
12	[18]	III.C. CL Techniques	4	Comunicação por voz/texto apenas entre subgrupos
13	[18]	III.C. CL Techniques	4	Silenciar utilizadores de um subgrupo
14	[19]	1. Introduction	1	Gerir repositórios de objectos

Tabela 4.3: Possíveis requisitos para o sistema, levantados através da literatura pedagógica com OpenSim.

Após a revisão da literatura pedagógica muitos dos possíveis requisitos encontrados já estão presentes na tabela 4.2 com a devida justificação. Assim sendo, será apresentada de seguida uma justificação para os novos possíveis requisitos, começando pelo topo da tabela.

Gerir objectos poderá ser algo fundamental devido à possibilidade de existir uma grande quantidade destes no mundo virtual. Desta forma, deverá ser possível remover objectos do mundo virtual, assim como obter e alterar a sua localização pois esta pode variar ao longo do tempo. Localizar objectos no mundo virtual está directamente relacionado com este requisito.

Localizar utilizadores poderá ser um requisito importante para verificar a distribuição de avatares no mapa do mundo virtual e também para perceber se algum utilizador está na zona em que era suposto estar ou nalguma zona onde nao era suposto estar (por exemplo, numa zona privada e sem autorização para aceder-lhe).

Guardar, replicar e ajustar zonas no mundo virtual poderá também ser muito importante. Desta forma, ao criar uma zona, o(s) responsável(eis) pela instalação OpenSim poderão guardá-la para depois ser replicada mais facilmente, sem o esforço de construir a zona novamente (isto é, sem necessitar de desenhar um edifício com um objecto para reprodução de slides, por exemplo, de novo). Ajustar a zona refere-se apenas a editar a localização desta, aumentar/diminuir as suas dimensões, etc, o que facilitará estas tarefas devido a não ter de executar o respectivo comando

no terminal da respectiva máquina (como descrito na secção 2.2.2).

Esta página foi deixada em branco intencionalmente.

Capítulo 5

Requisitos

De modo a identificar requisitos funcionais para a instalação OpenSim e para o *Backoffice web* foram executadas 3 estratégias, estando as duas primeiras descritas na secção 4.2 e 4.3, respectivamente. A primeira consistiu em identificar, instalar e avaliar quanto às suas funcionalidades as ferramentas de administração actuais. De seguida foi feita uma revisão da literatura com vista a identificar tentativas anteriores de implementar uma instalação OpenSim numa instituição com a finalidade de dar formação aos seus funcionários, assim como uma revisão da literatura pedagógica com o OpenSim de forma a tentar identificar requisitos funcionais que complementassem a solução. Por último, uma entrevista foi conduzida aos administradores de sistemas da Universidade Aberta de forma a ter uma melhor compreensão das suas tarefas do dia-a-dia e de maneira a identificar requisitos funcionais relacionados com a administração de sistemas. A preparação e estrutura desta está presente no anexo A.

Após a realização destes 3 passos, tornou-se claro que é necessário o sistema ter 3 tipos de actores:

1. Utilizadores externos: utilizadores que não pertencem à Uab, mas que são livres de utilizar o sistema após o seu registo neste, apesar de não terem acesso a algumas funcionalidades relativas à integração com um LMS e possivelmente a zonas do sistema;

2. Estudantes e professores da UAb: estudantes e professores da UAb que terão acesso a todas as funcionalidades menos as relativas à administração da instalação OpenSim;

3. Administradores de sistemas: gestores da instalação que terão acesso a todas as funcionalidades da solução implementada.

5.1 Requisitos funcionais

Nesta secção serão apresentados os requisitos funcionais levantados para o *backoffice web* assim como as alterações necessárias ao OpenSimulator (que no fundo também

são requisitos funcionais, estando indirectamente ligados ao *backoffice*) para que o sistema final seja utilizado por parte da UAb para realizar formações, treinos e avaliações nesta plataforma. Para melhor identificar e validar os requisitos funcionais defini cenários e *user stories*, sendo que, desta forma, é mais fácil identificar e discutir a validade destes requisitos para com os administradores de sistemas devido a estes não terem qualquer experiência nesta tecnologia. De seguida são apresentados os cenários definidos para o sistema.

Cenário 1

Um professor da unidade curricular Estética e Teoria da Arte do curso Licenciatura em Estudos Artísticos pretende ter um espaço virtual onde possa treinar os seus alunos em como montar uma exposição de arte. Para esse efeito pede ao administrador da instalação OpenSim para criar uma região privada à qual apenas podem aceder alunos pertencentes a essa unidade curricular. O administrador cria a região e verifica no mapa das regiões se a região está nas coordenadas correctas informando depois o professor. O professor pede então a criação de uma sessão de treino a acontecer nessa região no dia 15 de janeiro e que comece às 15:00 e acabe no dia 30 de janeiro às 17:00. O professor depois da sessão verifica as presenças nessa sessão. Visto que o professor apenas estava interessado no ambiente virtual para esta sessão de treino informa o administrador que não necessita do espaço virtual até ao próximo ano lectivo. Desta forma, o administrador desliga a região de forma a poder recomeçá-la quando for necessário. O administrador guarda também os conteúdos da região caso o professor da unidade curricular Arte do Ocidente Europeu queira utilizar o ambiente virtual para avaliar obras de arte expostas sendo assim possível carregar este *template* noutra região.

Cenário 2

O professor da unidade curricular Estética e Teoria da Arte informa o administrador de sistemas que alguns alunos não conseguem encontrar a região no mundo virtual. Obtendo esta informação o administrador de sistemas teleporta os utilizadores para a região e coordenadas correctas. Ao decorrer da sessão um utilizador altera o seu *avatar* para um dragão que ocupa todo o ecrã, deteriorando assim a experiência dos outros utilizadores. O aluno pede então apoio ao administrador de forma a que restaure a sua aparência no mundo virtual pois está a prejudicar a sessão de treino.

Cenário 3

Um professor não consegue ir à região da sua unidade curricular informando o administrador. Este verifica se a região está online e, verificando que não é o caso, verifica os logs do sistema para identificar o problema. Para solucionar o problema o administrador necessita de desligar todo o sistema OpenSim. Desta forma, envia uma mensagem para todas as regiões de forma a informar todos os utilizadores de que em alguns minutos o sistema irá ficar offline durante alguns minutos.

Na tabela 5.1 estão presentes as *user stories* criadas de forma a identificar os requisitos funcionais.

ID da US	*User story*
US-1	Como utilizador exterior à UAb, devo conseguir criar uma conta, fornecendo os meus dados pessoais (primeiro e último nome, *password*, email) para poder posteriormente aceder ao OpenSimulator.
US-2	Como utilizador exterior à UAb, devo conseguir alterar as minhas informações pessoais como email e *password*.
US-3	Como utilizador exterior à UAb, devo conseguir recuperar os dados de acesso através do meu email, de forma a poder aceder ao OpenSimulator caso me esqueça destes.
US-4	Como administrador de sistemas devo conseguir banir utilizadores que não sejam administradores e voltar a permitir o uso a utilizadores que estejam banidos.
US-5	Como administrador de sistemas devo conseguir visualizar uma lista de todos os utilizadores do sistema.
US-6	Como utilizador, no momento do registo ou do primeiro acesso, devo conseguir escolher a aparência do meu avatar para que os outros utilizadores me vejam no mundo virtual como escolhi.
US-7	Como administrador do sistema devo conseguir ver o estado das regiões, de modo a perceber se alguma região está indisponível.
US-8	Como administrador do sistema devo conseguir ver o mapa do mundo virtual de modo a percepcionar a localização das regiões no mundo virtual.
US-9	Como administrador do sistema devo conseguir ver a distribuição de utilizadores no mundo virtual de forma a perceber quais as áreas com mais utilizadores.
US-10	Como administrador do sistema devo conseguir criar novas regiões que são réplicas de regiões guardadas e deverei poder indicar a sua privacidade, sendo que se estas forem privadas a região é criada de forma automática em circunstâncias onde a privacidade é garantida.
US-11	Como administrador do sistema devo ser capaz de editar o *template* da região, que é escolhido a partir das regiões já guardadas.
US-12	Como administrador do sistema devo conseguir reiniciar uma região caso seja necessário.
US-13	Como administrador de sistema devo conseguir desligar regiões, mantendo-as prontos a reiniciar a qualquer momento.
US-14	Como administrador do sistema devo ser capaz de guardar as regiões, de modo a utilizá-las futuramente.
US-15	Como administrador do sistema deve-me ser apresentada uma listagem de utilizadores que têm acesso a uma região privada.
US-16	Como administrador do sistema, devo conseguir autorizar utilizadores a visitarem zonas privadas assim como revogar os seus direitos.
US-17	Como administrador do sistema devo autorizar os estudantes de uma unidade curricular a aceder a uma região privada, de modo a que estes consigam entrar na zona desejada.

US-18	Como administrador do sistema devo ser capaz de gerir os NPCs do sistema. Devo conseguir ver as suas coordenadas no mundo virtual, criá-los conforme através de *templates* de NPCs e removê-los do mundo virtual.
US-19	Como administrador do sistema devo ter conseguir ver os logs do sistema, de forma a conseguir identificar e mais facilmente resolver problemas do sistema.
US-20	Como administrador do sistema devo conseguir definir a localização onde se localizam os avatares no primeiro acesso.
US-21	Como administrador do sistema devo conseguir enviar mensagens para todas as regiões de forma a transmitir conteúdo necessário aos utilizadores.
US-22	Como administrador do sistema devo conseguir criar eventos no OpenSim. Devo conseguir definir a data e hora do evento, a quem se destina e onde decorre. Este eventos deverá ser criado e removido automaticamente no mundo virtual na data e hora escolhidas.
US-23	Como administrador do sistema devo conseguir editar eventos. Devo conseguir editar a data e hora do evento, a quem se destina e onde decorre.
US-24	Como administrador do sistema devo conseguir remover eventos já criados, cuja data seja superior à do dia actual.
US-25	Como administrador de sistema devo conseguir ver todos os eventos e sessões de treino.
US-26	Como utilizador externo e estudante ou professor devo conseguir ver todos os eventos e sessões de treino às quais tenha ou tivesse tido acesso.
US-27	Como administrador do sistema devo conseguir teleportar utilizadores para uma localização definida por mim, através de coordenadas.
US-28	Como estudante ou professor da UAb devo conseguir aceder ao OpenSim através dos dados de acesso do LMS.
US-29	Como utilizador externo à UAb devo conseguir alterar os dados de acesso no *backoffice web*.
US-30	Como administrador de sistemas devo conseguir associar uma sessão de treino a um espaço já presente no mundo virtual e que não esteja a ser utilizado por outra sessão de treino.
US-31	Como administrador do sistema devo conseguir consultar a lista de presenças nas sessões já executadas.
US-32	Como administrador do sistema devo conseguir dividir a comunicação em subgrupos de participantes, isto é, os utilizadores apenas ouvirão ou lerão os utilizadores permitidos. Para isto deverei selecionar os utilizadores que pertencerão a um subgrupo a partir da lista de utilizadores.
US-33	Como administrador do sistema devo conseguir fornecer aos utilizadores avatares pré-preparados com algum objecto ou *script* necessário para a realização da sessão.

US-34	Como administrador do sistema devo conseguir restaurar a aparência de qualquer avatar, selecionando o utilizador de uma lista de utilizadores.
US-35	Como utilizador externo, estudante ou professor devo conseguir pedir ajuda ao administrador de sistemas caso tenha alguma dúvida ou problema relacionado com o OpenSim. Este pedido deverá ser feito através de um formulário de contacto presente no *backoffice web*.

Tabela 5.1: User stories.

Os requisitos foram priorizados para conseguir facilmente identificar os requisitos mais importantes para o sistema. A priorização destes seguiu o método MoSCoW que define que existem quatro níveis de prioridade:

1. *Must Have*, para requisitos que sejam fundamentais para o produto e que necessitam de estar implementados para o sucesso do produto;
2. *Should have*, para requisitos que são importantes, mas não são fundamentais. Caso não estejam implementados a solução ainda é viável;
3. *Could Have*, para requisitos que seja desejável estarem presentes;
4. *Won't Have*, para requisitos que não irão ser implementados.

Na tabela 1 (em anexo) são apresentados os requisitos funcionais resultantes das 3 fases referidas no ínicio deste capítulo. A tabela é constituida por 3 colunas: a primeria indica o ID do requisito funcional de forma a ser mais fácil indicar o requisito ao qual me refiro quando for necessário, uma descrição do requisito funcional e a sua prioridade seguindo o método MoSCoW, referido acima. De modo a facilitar a leitura estes requisitos foram divididos em 4 categorias e as suas consequentes tabelas.

Na tabela 5.2 podemos ver os requisitos relacionados com a gestão de utilizadores e o controlo de acesso.

ID	Descrição	Prioridade
RF-1	Registo de utilizadores	*Must have*
RF-2	Editar utilizadores	*Must have*
RF-3	Recuperar dados de acesso	*Must have*
RF-5	Gerir grupos de utilizadores	*Should have*
RF-8	Ver distribuição de utilizadores no mundo virtual	*Should have*
RF-13	Listagem de utilizadores com acesso a zonas privadas	*Must have*
RF-14	Autorizar e revogar o acesso de utilizadores a uma região privada	*Must have*
RF-15	Autorizar estudantes de uma unidade curricular a aceder a uma região privada	*Could have*
RF-18	Definir localização base para os utilizadores	*Must have*
RF-23	Teleportar utilizadores	*Must have*

RF-25	Autenticação realizada através dos dados de acesso do LMS	*Must have*
RF-26	Estudantes apenas podem mudar a palavra chave através do LMS	*Must have*
RF-29	Poder dividir a comunicação em subgrupos de participantes	*Should have*
RF-33	Listagem de utilizadores	*Must have*
RF-36	Banir/Voltar a autorizar utilizadores	*Must have*

Tabela 5.2: Requisitos funcionais relacionados com a gestão de utilizadores e controlo de acesso.

Na tabela 5.3 podemos encontrar os requisitos relacionados com a gestão do terreno virtual. Deverá ser possível guardar regiões e caregá-las como *templates* ao criar ou editar regiões.

ID	Descrição	Prioridade
RF-6	Estado das regiões	*Must have*
RF-7	Ver mapa do mundo virtual	*Must have*
RF-9	Criar regiões	*Must have*
RF-10	Editar regiões	*Must have*
RF-11	Reiniciar regiões	*Must have*
RF-12	Guardar regiões	*Must have*
RF-34	Listagem de regiões	*Must have*
RF-37	Desligar regiões	*Must have*
RF-38	Guardar o estado do sistema (regiões)	*Must have*

Tabela 5.3: Requisitos funcionais relacionados com a gestão do terreno virtual.

Na tabela 5.4 estão inseridos os requisitos funcionais acerca de gestão de eventos e de sessões de treino. Os administradores de sistemas deverão conseguir criar, editar e eliminar sessões de treino assim como deverá ser possível a criação automática de sessões de treino calendarizadas.

ID	Descrição	Prioridade
RF-23	Criar eventos	*Must have*
RF-24	Editar eventos	*Must have*
RF-25	Eliminar eventos	*Must have*
RF-35	Listagem de eventos	*Must have*
RF-27	Criar e eliminar sessões de treino automaticamente	*Must have*
RF-30	Associar uma sessão de treino a um espaço já existente	*Must have*
RF-31	Controlar as presenças nas sessões de treino	*Must have*
RF-34	Fornecer aos participantes avatars pré-preparados para as sessões	*Should have*

Tabela 5.4: Requisitos funcionais relacionados com a gestão de eventos sessões de treino.

Na tabela 5.5 estão os requisitos que não se encontravam em nenhuma das outras 3 categorias nem se relacionavam entre si o suficiente para criar uma nova categoria. Nesta tabela estão requisitos relativos à facilidade de resolução de problemas como a gestão dos *logs* do sistema ou um sistema que permita aos utilizadores reportar problemas de modo a que sejam resolvidos pelos administradores do sistema.

ID	Descrição	Prioridade
RF-4	Definir avatar pré-definido	*Must have*
RF-19	Gestão de NPCs	*Should have*
RF-20	Gestão de logs	*Must have*
RF-22	Enviar mensagens para todas as regiões	*Must have*
RF-35	Poder restaurar a aparência dos avatars	*Must have*
RF-37	Sistema de suporte	*Must have*

Tabela 5.5: Restantes requisitos funcionais do sistema a desenvolver.

De modo a implementar os requisitos referidos acima será necessário realizar algumas alterações no OpenSimulator. Para além disto, alguns requisitos encontrados na secção 4.3 requerem também alterações nesta tecnologia. Na tabela 5.7 são apresentadas as mudanças exigidas para a implementação dos requisitos presentes na tabela 1 assim como alguns requisitos que apenas incidem no mundo virtual, não tendo qualquer impacto no *backoffice web*. Na solução final não foi necessário efectuar qualquer alteração no código fonte do OpenSimulator, mas caso os requisitos *should have* e *could have* fossem implementados seria necessário alterá-lo.

ID do RF	Descrição	Prioridade
RF-38	Ter acesso aos avisos e notificações do LMS no espaço virtual	*Should have*
RF-39	Localização/Espaço para aceder aos conteúdos do LMS no espaço virtual	*Should have*
RF-40	Ter acesso aos sumários dos tópicos do LMS no espaço virtual	*Could have*
RF-41	Gravar o comportamento dos actores e outros elementos	*Won't have*
RF-42	Gravar comunicações via audio	*Should have*
RF-43	Nomes de utilizador do LMS devem ser associados ao OpenSim	*Must have*
RF-44	Autenticação no OpenSim realizada usando os dados de acesso do LMS	*Must have*

Tabela 5.7: Alterações a efectuar no OpenSimulator.

5.2 Restrições técnicas

As restrições técnicas inerentes ao sistema são:

1. O sistema tem de seguir os princípios de *Security by Design*;

5.3 Restrições de negócio

As restrições de negócio inerentes ao sistema são:

1. O projecto tem de estar acabado até 23 de janeiro de 2019;
2. Deverá ser criado um manual de instalação para o produto final.

Capítulo 6

Atributos de qualidade e Arquitectura

No desenho de um sistema, os atributos de qualidade devem ser considerados de forma a que sejam encontradas as qualidades mais pertinentes para o projecto a desenvolver pois existe sempre um *trade-off* entre a qualidade que queremos realçar e outra(s) qualidade(s). Neste capítulo avalio e identifico os atributos de qualidade do *backoffice web*, justificando as minhas escolhas e esboçando cenários para cada um destes. De seguida a arquitectura do sistema é esboçada respeitando as qualidades pertinentes. Finalmente, são também discutidas as tecnologias a utilizar.

6.1 Atributos de qualidade

Os atributos de qualidade do projecto foram identificados tendo por base a entrevista aos administradores de sistemas da Universidade Aberta mencionada na secção 3 e a literatura estudada para identificar as principais tarefas e dificuldades dos administradores de sistemas [15, 20]. Foram identificadas as seguintes qualidades pertinentes ao sistema:

1. **Usabilidade:** a importância desta qualidade tornou-se evidente depois da entrevista e de estudar as principais dificuldades de administradores de sistemas. Quando um sistema de informação é utilizado periodicamente é bastante díficil para o gestor recordar as acções necessárias que tem de realizar para uma determinada finalidade. Assim, se o sistema for fácil de utilizar e perceber os administradores de sistemas demorarão menos tempo a gerir a instalação OpenSim.

2. **Segurança:** o OpenSimulator tem dados pessoais dos utilizadores, como nome, email, etc. Assim, é necessário garantir que estes dados não estejam acessíveis a ninguém sem ser o próprio utilizador.

3. **Responsividade:** esta qualidade também se demonstra importante uma vez que, em primeiro lugar, nem todas as operações são idempotentes o que pode

levar a situações indesejadas caso sejam realizadas mais do que uma vez, e, em segundo lugar, existem operações , como a reinicialização de regiões, que não têm resposta imediata o que poderá levar a que os administradores do sistema não saibam se a operação foi realizada, está a ser realizada, ou não foi realizada.

De seguida são apresentados alguns cenários que descrevem a resposta do sistema em relação a estes atributos de qualidade.

Cenário 1	**Usabilidade (Ajuda contextual)**
Fonte do estímulo	Utilizador autenticado no *backoffice*
Estímulo	Utilizador pousa o cursor do rato em cima de um campo de um formulário
Ambiente	*Backoffice* em funcionamento normal
Artefacto	*Backoffice web*
Resposta	Um pop-up aparece com ajuda contextual
Medição	A ajuda contextual é apresentada após o cursor do rato parar durante exactamente 2 segundos

Tabela 6.1: Cenário 1 de Usabilidade.

Cenário 2	**Usabilidade**
Fonte do estímulo	Administrador do sistema
Estímulo	Administrador do sistema necessita de criar uma sessão de treino
Ambiente	*Backoffice* em funcionamento normal
Artefacto	*Backoffice web*
Resposta	A sessão de treino é criada com sucesso
Medição	O administrador do sistema executa a acção em menos de 45 segundos

Tabela 6.2: Cenário 2 de Usabilidade.

Cenário 4	**Segurança (Autenticação)**
Fonte do estímulo	Visitante do *backoffice web*
Estímulo	Visitante não autenticado tenta aceder a uma página *web* através do URL
Ambiente	*Backoffice* em funcionamento normal
Artefacto	*Backoffice web*
Resposta	O visitante é redirecionado para a página inicial
Medição	Nenhuma página que necessite de autenticação é apresentada a um visitante

Tabela 6.3: Cenário 1 de Segurança.

Cenário 5	**Segurança (Autorização)**
Fonte do estímulo	Estudante autenticado no *backoffice*
Estímulo	Estudante tenta reiniciar uma região através do URL
Ambiente	*Backoffice* em funcionamento normal
Artefacto	*Backoffice web*
Resposta	Estudante é redirecionado para a página inicial
Medição	Nenhuma página é apresentada a um membro não autorizado a usá-la

Tabela 6.4: Cenário 2 de Segurança.

Cenário 6	**Segurança (*Cross-site Scripting*)**
Fonte do estímulo	Atacante
Estímulo	Atacante tenta injectar um *script* malicioso no sistema
Ambiente	*Backoffice* em funcionamento normal
Artefacto	*Backoffice web*
Resposta	O atacante é redirecionado para a página inicial
Medição	O *script* malicioso não é executado

Tabela 6.5: Cenário 3 de Segurança

Cenário 7	**Responsividade (Resposta imediata)**
Fonte do estímulo	Administrador do sistema autenticado no *backoffice*
Estímulo	Administrador tenta eliminar um evento
Ambiente	*Backoffice* em funcionamento normal
Artefacto	*Backoffice web*
Resposta	Um *pop-up* é apresentado ao administrador a dar conta de que o evento foi eliminado
Medição	Um aviso deve ser apresentado ao administrador em menos de 3 segundos

Tabela 6.6: Cenário 1 de Responsividade.

Cenário 8	**Responsividade (Feedback imediato)**
Fonte do estímulo	Administrador do sistema autenticado no *backoffice*
Estímulo	Administrador tenta alterar um espaço no mundo virtual
Ambiente	*Backoffice* em funcionamento normal
Artefacto	*Backoffice web*
Resposta	Um *pop-up* é apresentado ao administrador a informar que a operação não é imediata e o tempo máximo da operação
Medição	Um aviso deve ser apresentado ao administrador em menos de 3 segundos

Tabela 6.7: Cenário 2 de Responsividade.

6.2 Arquitectura

A arquitectura de um sistema de informação é condicionada pelos atributos de qualidade e pelo seu impacto na arquitectura. Neste caso, nenhuma das qualidades identificadas tem um impacto bastante elevado na arquitectura, sendo a segurança o atributo de qualidade que mais influencia o desenho da arquitectura do sistema. Desta forma, optei por uma arquitectura de três camadas. Neste tipo de arquitectura, existem três camadas:

- A camada de apresentação, responsável pela apresentação das informações e pela interação com o utilizador. Esta camada nunca comunica directamente com a camada de dados, o que leva a um aumento na segurança do sistema, comunicando apenas com a camada de negócio;
- A camada de negócio (ou lógica), responsável por gerir toda a lógica da aplicação. Esta camada é o ponto central neste tipo de arquitectura, sendo

que é lá que a maior parte do processo é executado e onde são tomadas decisões. Esta camada é também o ponto de ligação entre a camada de dados e a camada de apresentação;

- A camada de dados, onde os dados relativos à aplicação são guardados. Comunica apenas com a camada de negócio, devolvendo os dados que esta necessita para responder a um pedido efectuado pelo cliente, na camada de apresentação.

Na figura 6.1 está presente a vista geral da arquitectura do sistema. Este contém uma instalação do OpenSimulator no modo *grid* visto que o sistema terá de disponibilizar esta tecnologia a centenas ou talvez milhares de utilizadores. Esta configuração, como visto no capítulo 2.2, permite a distribuição de simuladores por várias máquinas comunicando todas com uma máquina central onde são disponibilizados os serviços necessários ao correcto funcionamento do OpenSimulator. Outra vantagem inerente a desenvolver um sistema direcionado a este tipo de configuração é o facto de se funcionar numa instalação *grid* uma instalação *standalone* também é suportada. De modo a comunicar com cada um dos simuladores do OpenSim é utilizado o RemoteAdmin, um componente que permite a comunicação com os simuladores através do protocolo XML-RPC (tal como visto na secção 4.2), pois desta forma o *backoffice* está preparado para lidar com um número arbitrário de servidores. A plataforma de elearning, o servidor de autenticação central e o servidor de email da UAb são descritos na secção 2.1.

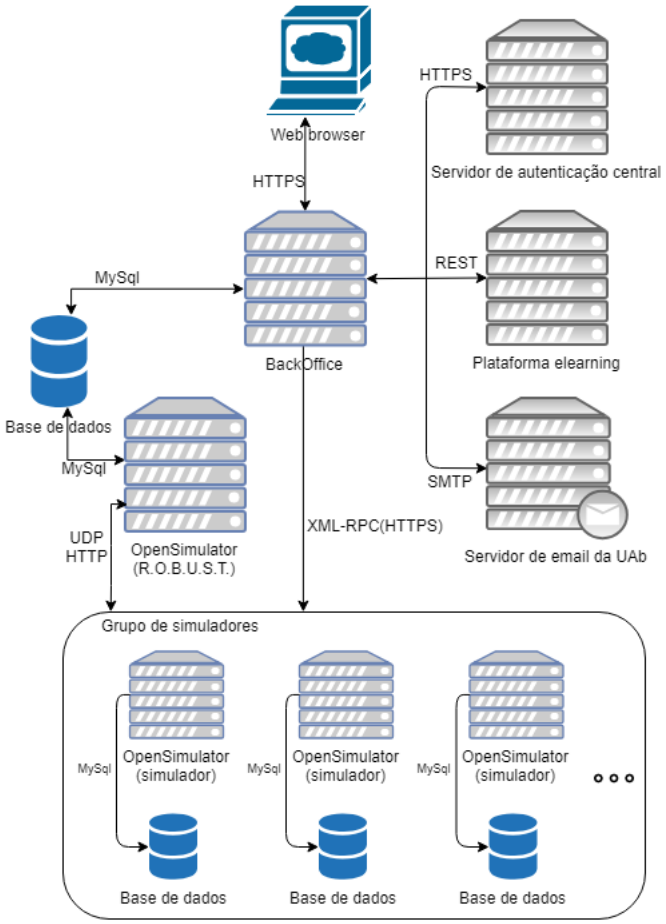

Figura 6.1: Vista geral da arquitectura do sistema.

Como referido anteriormente e como pode ser visto na imagem 6.2, o *backoffice web* está dividido em três camadas: a de apresentação, a de negócio e a de dados. A camada de negócio é responsável pela lógica do sistema. Assim sendo comunica com todos os sistemas externos ao sistema. Por um lado comunica com os sistemas de informação da UAb, como referido anteriormente. Por outro lado é a camada que comunica com o Redis de modo a que este indique ao Celery quais as tarefas a escalonar de modo a serem executadas assíncronamente pelos *workers*. Esta necessidade de executar pedidos assíncronos numa determinada data e hora deve-se à criação e eliminação de eventos automaticamente no mundo virtual, como pode ser lido na secção **??**. Também é esta camada que trata dos pedidos XML-RPC efectuados aos simuladores OpenSimulator. A camada de dados é a única que comunica com a base de dados, que é compartilhada com a instância R.O.B.U.S.T. que trata dos serviços OpenSim. Esta solução de comunicação é explicada com mais detalhe na subsecção **??**.

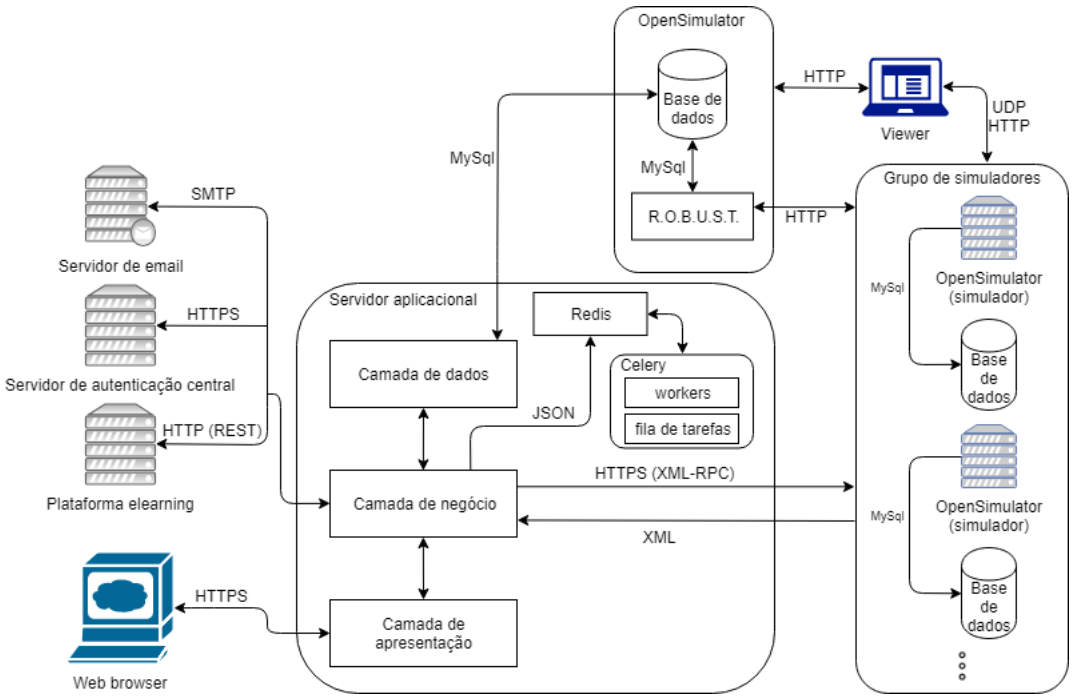

Figura 6.2: Vista de decomposição e camadas da arquitectura do sistema.

6.3 Tecnologias

Para o desenvolvimento do projecto utilizei as seguintes tecnologias:

- **Django:** *framework* escrita em Python para desenvolvimento de aplicações *web*;
- **Python:** linguagem de programação de alto nível;
- **HTML5:** linguagem usada para a construção de páginas *web*;
- **CSS:** linguagem de design que permite definir a aparência de páginas *web*;

- **JavaScript:** linguagem de programação que permite implementar operações complexas em páginas *web* com maior facilidade;
- **Bootstrap:** biblioteca para o desenvolvimento de páginas *web*;
- **MySQL:** motor de base de dados;
- **Apache:** servidor *web*.
- **Celeris:** fila de tarefas assíncrona.
- **Redis:** Corretor de mensagens *open-source*.

De realçar que algumas tecnologias foram escolhidas de forma a respeitarem os atributos de qualidade do sistema ou de forma a prevenir incompatibilidades entre os dados presentes no sistema.

De forma a proteger o sistema contra ataques irá ser utilizada a plataforma Django. Esta tecnologia previne o sistema contra *Cross-site Scripting* (XSS), *SQL Injection* e *Cross-site Request Forgery* (CSRF) [21]. Os ataques XSS são prevenidos escapando automaticamente todos os caracteres especiais HTML, não permitindo assim a execução *scripts* maliciosos. *SQL Injection* é prevenido escapando os caracteres especiais presentes nos parâmetros das *queries* SQL, que podem ser introduzidos pelo utilizador. De modo a proteger o sistema contra CSRF, todos os templates Django têm um *token* aleatório sempre que apresentados, verificando sempre o valor deste presente no pedido HTTP e que deve ser igual ao *token* apresentado para o pedido ser válido.

Já a utilização de MySQL prende-se com o facto do OpenSimulator utilizar este motor de base de dados, sendo assim mais fácil a integração e prevenindo possíveis incompatibilidades de dados entre diferentes motores de bases de dados (por exemplo, em MySQL existe TINYINT, o tipo de dados mais pequeno para inteiros, sendo que o PostgreSQL tem SMALLINT como a unidade mais pequena para inteiros).

Esta página foi deixada em branco intencionalmente.

Capítulo 7

Implementação

Neste capítulo estão descritos os detalhes de desenvolvimento da solução assim como a listagem de todas as suas funcionalidades. Este capítulo está dividido em 3 secções:

1. *Front-end*: que descreve o desenvolvimento da interface *web*;
2. *Back-end*: que descreve o desenvolvimento do *back-end*;
3. Funcionalidades: que descreve as funcionalidades implementadas no sistema.

7.1 *Front-end*

Nas subsecções seguintes serão apresentados os detalhes da implementação do *front-end*, tais como as ferramentas utilizadas no seu desenvolvimento, a disponibilização das páginas *web* e a protecção contra *Cross-site Request Forgery*(CRSF).

7.1.1 *Bootstrap*

De modo a facilitar o desenvolvimento da interface *web* foi utilizada uma biblioteca *open-source* chamada Bootstrap [1]. Foi ainda escolhido um tema responsivo encontrado no *website* Bootswatch[2] com o nome Sandstone[3], desenvolvido por Thomas Park e distribuido através da *MIT License*. Este tema consiste num ficheiro *cascading style sheets*(CSS) que deve ser utilizado em conjunto com a instalação base do Bootstrap.

7.1.2 *Templates*

Para facilitar a geração de HTML dinamicamente, o Django utiliza *templates*. Um *template* é constituido por uma parte estática, código HTML, e por uma parte

[1]https://getbootstrap.com/
[2]https://bootswatch.com/
[3]https://bootswatch.com/sandstone/

dinâmica que descreve como o conteúdo deve ser apresentada na página *web*. O Django disponibiliza suporte a dois tipos de linguagens para *templates*, Django Template Language(DTL) e Jinja2. Utilizei HTML e DTL para a implementação das páginas *web* uma vez que não tinha experiência com Jinja2 e a documentação da DTL está muito bem estruturada e integrada nos tutoriais presente na documentação do Django.

Os *templates* permitem a extensão e inclusão de código HTML automaticamente evitando assim a replicação de código e permitindo um menor esforço computacional ao servidor uma vez que este apenas necessita de gerar código HTML para as partes exclusivas à página *web* requisitada o que leva a um menor tempo de resposta e consequentemente a uma experiência mais agradável por parte do utilizador.

Na aplicação desenvolvida existe uma página base, chamada *base_template.html*. Esta página tem todo o conteúdo comum a todas as páginas *web* e também importa todas as bibliotecas externas necessárias ao funcionamento do sistema. Esta página tem também duas instâncias da *tag* {% *block* <*nome do bloco*>%}, uma para alterar o conteúdo da *tag* HTML *body*, que permite que as outras páginas introduzam aí o seu conteúdo, e outra para a importação de bibliotecas externas e ficheiros locais necessários que apenas são necessários ao funcionamento de uma página *web* em específico. De forma a estender a página base as outras páginas necessitam de ter a tag {% *extends base_template.html* %} no topo e utilizar a *tag* {% *block* <*nome do bloco*>%} para alterar o seu conteúdo.

7.1.3 Django Channels

Django Channels é uma biblioteca que permite a um projecto Django a execução de código assíncrono e processar não só pedidos HTTP como pedidos de outros protocolos de comunicação, como WebSockets. Esta flexibilidade permite, por exemplo, que páginas *web* sejam alteradas assíncrona e dinamicamente.

A utilização desta biblioteca neste projecto prende-se com o facto de os pedidos XML-RPC efectuados pelo *backoffice web* aos simuladores OpenSim terem um tempo de resposta indeterminado existe a necessidade dos pedidos serem assíncronos. Consequentemente as páginas *web* que resultam em um pedido assíncrono a um simulador necessitam de ser actualizadas dinamicamente quando o pedido termina. Este comportamento é conseguido através da utilização de *WebSockets*. Ao carregar uma página *web* o *browser* cria um *WebSocket* entre si mesmo e o *backoffice*. O processamento dos pedidos recebidos pelo lado do servidor é efectuado através de um *consumer*, uma unidade do Django Channels que consome eventos. Quando é recebida uma mensagem o *consumer* irá efectuar o pedido XML-RPC ao simulador OpenSim e esperar pela resposta. Quando a resposta é recebida uma mensagem será enviada ao cliente que altera a página *web* de forma a apresentar o resultado do pedido XML-RPC.

7.1.4 Protecção contra CRSF

A protecção contra *Cross Site Request Forgery* é garantida através da utilização de um *token* gerado pelo Django e presente em todos os formulários. Este *token* é gerado através de um número aleatório e depois é ainda aplicado um *salt* aleatório sobre o *token*. Este é apenas utilizado uma única vez de modo a prevenir ataques como *cookie hijacking* sendo renovado sempre que um formulário é submetido. Desta forma é garantido que os dados recebidos pelo servidor tiveram origem num formulário criado por este.

A utilização deste *token* nos *templates* Django é extremamente fácil, sendo para isso apenas necessária a utilização da *tag* **{% csrf_token %}** em todos os formulários. É necessário também garantir que a opção *django.middleware.csrf.CsrfViewMiddleware* está presente na secção *middleware* e antes de qualquer opção de *middleware* que assuma a segurança em relação a ataques CSRF.

7.2 Back-end

Nas subsecções seguintes serão apresentados os detalhes da implementação do *back-end*, tais como o detalhe das configurações do Django e do próprio *Backoffice*, as tecnologias utilizadas e as camadas que constituem o sistema.

7.2.1 Configuração do Django

A configuração do Django está presente no ficheiro *setting.py*. Ao criar um projecto Django algumas das configurações necessárias já são disponibilizadas, mas, devido à natureza do projecto e às decisões que tomei, necessitei de fazer bastantes alterações neste ficheiro, nomeadamente alterar a configuração da ligação à base de dados de forma a utilizar uma base de dados MySql que já estava criada.

Este ficheiro tem as configurações relativas à protecção contra CRSF, à localização do URL Dispatcher(explicado na secção 7.2.3, à integração com o Celery, etc.

7.2.2 Configuração do *backoffice*

Devido à necessidade de integração com o OpenSimulator é necessário para o *backoffice* funcionar correctamente configurar algumas variáveis. Estas variáveis estão presentes em três ficheiros no formato JSON:

1. database_details.json: ficheiro que contém as informações necessárias para a ligação à base de dados, como o endereço ip, o porto, o nome da base de dados e as credenciais para a autenticação;

2. email_details.json: que contém as informações necessárias para o envio automático de emails;

3. connection_details.json: que contém as informações necessárias para a utilização do RemoteAdmin, o endereço IP dos simuladores, etc;

4. support_email.json: que contém o endereço de email para onde são enviadas os pedidos do sistema de suporte.

Estes ficheiros são explicados com detalhe no manual de instalação escrito para os administradores de sistemas da Universidade Aberta.

7.2.3 URL dispatcher

O URL dispatcher, presente no ficheiro urls.py, associa os URLs com a camada lógica da aplicação, as *views*. Para isso, o Django compara o URL presente no pedido com os URLs presentes no ficheiro urls.py por ordem e, quando encontra um URL que seja igual ao do pedido, executa a função correspondente presente na camada de negócio. Os URLs presentes no *dispatcher* podem conter expressões regulares que resultam numa maior flexibilidade nos URLs da aplicação.

7.2.4 Camada de dados

A camada de dados da aplicação é constituida pelo ficheiro models.py. Este ficheiro contém os modelos de dados presentes na aplicação e mapeia-os em tabelas na base de dados. Um modelo é uma classe que contém atributos. A classe é mapeada numa tabela da base de dados e os seus atributos em colunas dessa tabela.

No *backoffice* desenvolvido foi necessária a integração com uma base de dados previamente definida. Desta forma, foi utilizado o comando *python manage.py inspectdb ¿ models.py* de forma a mapear as tabelas da base de dados dos serviços R.O.B.U.S.T. do OpenSimulator em modelos que poderiam ser alterados e consultados. Visto que foram necessárias alterações à base de dados foi necessário executar migrações de dados de forma a alterar a base de dados. Estas foram criadas com recurso ao comando *python manage.py makemigrations* e executadas pelo comando *python manage.py migrate*.

7.2.5 Camada de negócio

A camada de negócio é o ponto central da aplicação pois é a única que consegue comunicar com a camada de apresentação e a camada de dados e também porque é nesta camada que toda a lógica da aplicação está presente.

7.2.6 Celery e Redis

O Celery é uma biblioteca que permite a execução assíncrona e periódica ou esporádica de tarefas recorrendo ao escalonador de tarefas crontab[4]. Estas tarefas são

[4]https://pypi.org/project/python-crontab/

executadas pelos *workers* que depois podem guardar o resultado de diversas formas, entre elas utilizando bases de dados de forma a garantir a persistência dos dados. O Redis é um corrector de mensagens que tem como função gerir a fila de mensagens e traduzir a linguagem utilizada pelo remetente e traduzir numa linguagem entendida pelo receptor. Neste caso, traduzindo a tarefa(a mensagem) de Python para um formato JSON entendido pelo *worker* que o irá executar.

Para a execução das tarefas assíncronas, nomeadamente a criação e eliminação de eventos automaticamente no mundo virtual é necessário ter pelo menos um *worker* Celery e o servidor Redis a funcionar. A integração do Celery e Redis com o Django foi efectuado seguindo um tutorial[5] presente na documentação do Celery.

7.3 Funcionalidades

Nesta secção estão descritas as funcionalidades do sistema, incluindo como foram implementadas e quais as dificuldades encontradas. O acesso a estas funcionalidades pode ser dividido em dois tipos de utilizadores: os administradores do sistema e todos os outros utilizadores. Os administradores têm acesso a todas as funcionalidades do *backoffice web*, já os outros utilizadores têm acesso à autenticação, à criação de utilizadores, à recuperação da *password*, à edição do próprio utilizador, ao sistema de suporte e à listagem de eventos. Este controlo é efectuado utilizando um *token* aleatório gerado no momento do *login* e o nível de permissões do utilizador.

7.3.1 Criar e editar utilizador

A funcionalidade de criar e editar utilizadores de uma instalação *grid* do OpenSimulator é possível apenas através de um pedido HTTP ao servidor que está a executar os serviços do OpenSim ou alterando a base de dados directamente.

A grande desvantagem da primeira opção é o facto de, depois de activar a manipulação de utilizadores na configuração do OpenSim, o pedido HTTP referido não ter qualquer tipo de segurança[6]. O pedido HTTP é feito através de um POST cujos parâmetros são apenas o método que se pretende invocar, neste caso *createuser* (para a criação de utilizadores) ou *setaccount* (para a edição de utilizadores), e as informações da conta como nome de utilizador e *password*. Desta forma não é garantido qualquer tipo de segurança visto que qualquer pessoa conseguiria enviar este pedido ao servidor tendo assim acesso a este quando não tem qualquer tipo de permissão. Esta alternativa para a criação e edição de utilizadores pode ter algum tipo de segurança, como referido na documentação do OpenSim, caso, através do uso de uma *firewall*, seja apenas permitido o acesso a estes URLs dos IPs pertencentes às máquinas dos administradores da instalação. No caso específico da solução desenvolvida seria necessário permitir o acesso do IP pertencente à máquina a executar o *backoffice* o que levantaria alguns problemas de segurança pois um atacante, caso obtenha acesso a esta máquina, consegue criar e editar utilizadores de forma a que

[5] http://docs.celeryproject.org/en/latest/django/first-steps-with-django.html
[6] http://opensimulator.org/wiki/UserManipulation

estes se tornem a administradores.

Assim sendo, optei por implementar a segunda alternativa o que levou a algumas dificuldades. A documentação do OpenSimulator não tem presente quais as tabelas da base de dados que são necessárias alterar para criar com sucesso um utilizador. Desta forma, precisei de criar utilizadores através da consola disponibilizada na máquina a correr o processo R.O.B.U.S.T. e consultar manualmente todas as tabelas da base de dados que eram alteradas com a criação deste utilizador e entender o que seginificam algumas colunas de algumas tabelas da base de dados que não estão especificadas na documentação[7]. Após ínumeras tentativas percebi que o utilizador necessita de um inventário de forma a este ser criado.

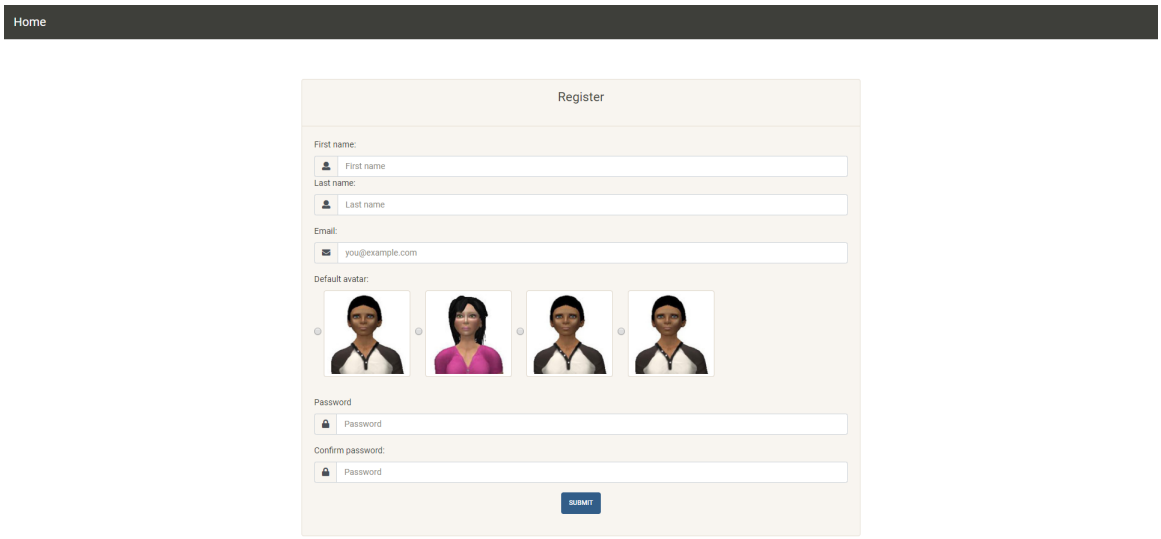

Figura 7.1: Página de criação de um utilizador.

Na figura 7.1 está presente o ecrã de criação de utilizadores. Como se vê, para o registo de um utilizador é necessário a introdução do primeiro e último nome, de um email e a escolha de um avatar que será a aparência do utilizador no mundo virtual (como explicado na subsecção 7.3.4. A página de edição de um utilizador, presente na figura 7.2, é uma variantes do ecrã de registo. Este ecrã apenas permite a alteração da *password*, do primeiro e último nome e do email a utilizadores externos à Universidade Aberta. A utilizadores internos é apenas permitida a alteração da sua aparência no mundo virtual(fig. 7.3).

7.3.2 Autenticação de utilizadores

De forma a autenticar os utilizadores o OpenSimulator guarda na base de dados um *salt*, uma *hash* do algoritmo MD5 gerada através de um UUID (*Unique Universal Identifier*) aleatório, e uma *hash* gerada a partir do seguinte esquema: MD5(MD5(password) + ":"+ MD5(salt)) em que MD5 corresponde a uma função

[7]http://opensimulator.org/wiki/Database:Documentation

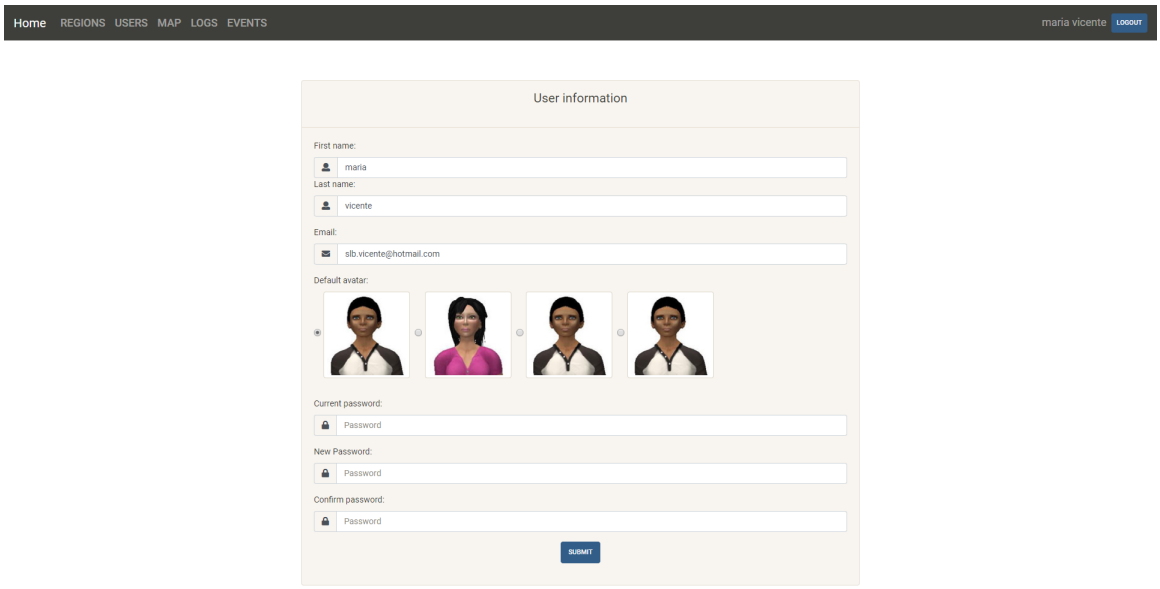

Figura 7.2: Página de edição de um utilizador externo.

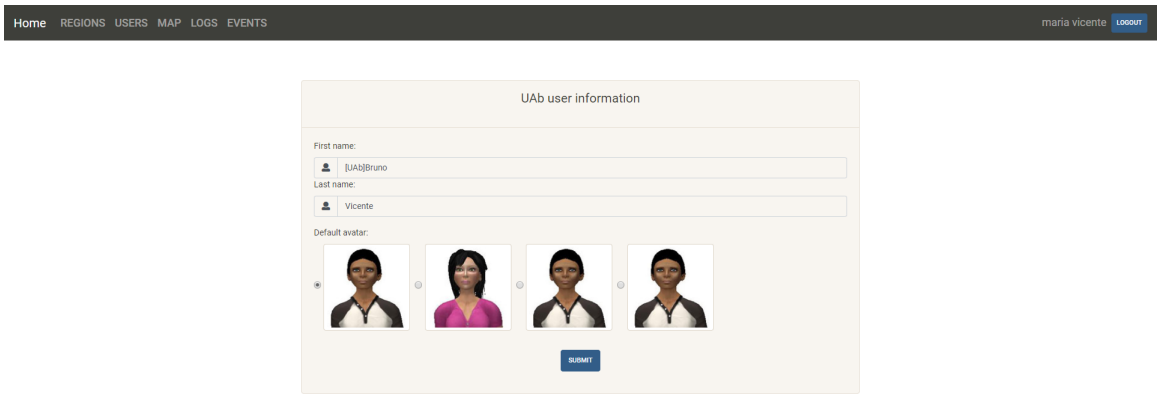

Figura 7.3: Página de edição de um utilizador da UAb.

que retorna a *hash* resultante do algoritmo MD5, password corresponde à *password* introduzida pelo utilizador e salt o UUID gerado aleatoriamente.

A autenticação de utilizadores pode ser dividida em duas partes: a autenticação de utilizadores externos à Universidade Aberta e autenticação de utilizadores da Universidade Aberta. Para autenticar os utilizadores externos ao OpenSimulator é seguido o algoritmo descrito no parágrafo anterior, tendo o utilizador de introduzir o primeiro e último nome e a *password* na página inicial (figura 7.4). Para a autenticação de utilizadores da Universidade Aberta é necessário autenticar o utilizador através das credenciais da Universidade Aberta. De forma a permitir esta autenticação tanto no *viewer* como no *backoffice* é necessário guardar na base de dados o nome do utilizador e uma *hash* que corresponde ao resultado do esquema já descrito. A solução encontrada para este problema foi criar um utilizador na base de dados do OpenSimulator depois da primeira autenticação tanto no *viewer* como no *backoffice*. Depois de tentar alterar o código fonte do OpenSimulator de forma a permitir esta autenticação num serviço externo, procedi a testar esta autenticação

através do *viewer* sendo que reparei que este envia ao simulador uma hash MD5 da *password*. Isto leva a que não se consiga depois tentar a autenticação no servidor de autenticação central da Universidade Aberta. De seguida, consultei o código fonte de um *viewer* de forma a tentar alterar a forma de enviar a *password* do *viewer* ao servidor OpenSim. Não consegui identificar onde alterar esta funcionalidade e, desta forma, procedi à implementação de outra solução.

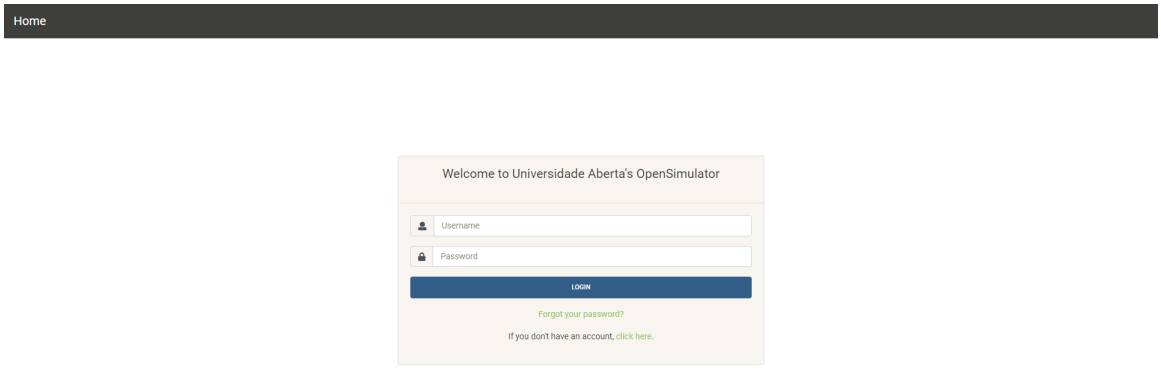

Figura 7.4: Página de inicial do *backoffice web*.

De forma a um utilizador da UAb ter acesso ao mundo virtual através do *viewer* é necessário realizar pelo menos a primeira autenticação através do *backoffice*. Depois da autenticação ser efectuada com sucesso no servidor de autenticação central é criado um utilizador OpenSim guardando o primeiro e último nome, o *username* da UAb e uma *hash* documentada na secção 7.3.1. De forma a identificar este utilizador como utilizador da Universidade Aberta é inserido o prefixo "[UAb]"ao primeiro nome.

Esta abordagem implica que o utilizador não tenha a *password* sincronizada entre o OpenSimulator e o servidor de autenticação central da UAb. De forma a colmatar este problema, quando um utilizador se autentica pela segunda e posteriores vezes a *password* deste utilizador é actualizada.

7.3.3 Recuperação dos dados de acesso

Esta funcionalidade pode ser acedida através da página inicial (fig. 7.4) e o ecrã correspondente está presente na figura 7.5. Neste o utilizador necessita de inserir o primeiro e último nome e o email caso o utilizador seja externo à universidade aberta. Se os dados introduzidos corresponderem a um utilizador pertencente à UAb este é reencaminhado para o ecrã de esquecimento de *password* dos Sistemas de Informação da UAb. Caso seja um utilizador externo à Universidade Aberta e os dados estejam correctos é enviada uma nova *password* ao utilizador via email. Esta *password* é gerada no *backoffice* e actualizada na base de dados. É de realçar que ninguém tem acesso à nova *password* gerada através de 16 caracteres aleatórios e enviada através do protocolo SMTP através do porto 465, ou seja, o email enviado é enviado através de SMTPS (*Simple Mail Transfer Protocol over SSL*) garantindo assim a encriptação da mensagem.

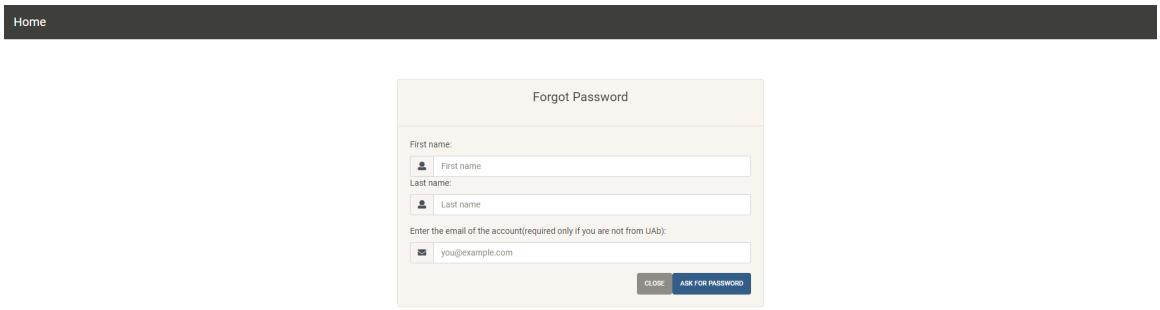

Figura 7.5: Página de recuperação de *password*.

7.3.4 Definir aparência

Esta funcionalidade foi implementada através de mudanças directas na base de dados. Para perceber o que e como deveria ser criado e/ou alterado consultei a documentação do OpenSimulator assim como criei utilizadores e verifiquei o que era alterado nas tabelas da base de dados sempre que o fazia uma vez que a documentação do modelo de dados do OpenSim não está completa.

O utilizador pode definir o seu *avatar* quando se regista ou altera as suas informações para os utilizadores externos à UAb ou quando altera as suas informações para utilizadores da UAb (como pode ser visto nas figuras 7.1 e 7.2).

Os administradores de sistemas podem definir as aparências entre as quais os utilizadores podem escolher adicionando a imagem (que pode ser uma captura de ecrã) à directoria "/static/images"e adicionando o nome pelo qual é identificada a aparência ao ficheiro *connection_details.json*. É ainda necessário adicionar os *assets*(componentes necessários à renderização do *avatar*) ao serviço *Library*. Estes passos estão detalhados no manual de utilização.

7.3.5 Restaurar aparência

Para restaurar a aparência de um *avatar* o administrador necessita de clicar no botão com o indicador 1 no ecrã de listagem de utilizadores presente na figura 7.6. Para implementar esta funcionalidade é guardado na base de dados o nome do *avatar* escolhido pelo utilizador quando este se regista e alterado quando é editado. Depois é executada a mesma função de definir a aparência do utilizador, descrita na sub-secção 7.3.4.

7.3.6 Teleportar utilizadores

Tal como ilustrado no cenário 1 presente na secção 5.1, esta funcionalidade é extremamente útil no caso de alguns utilizadores não encontrarem algum evento ou alguma região, não conseguindo assim participar numa sessão de formação no mundo virtual.

À semelhança da funcionalidade anterior, esta também pode ser acedida através

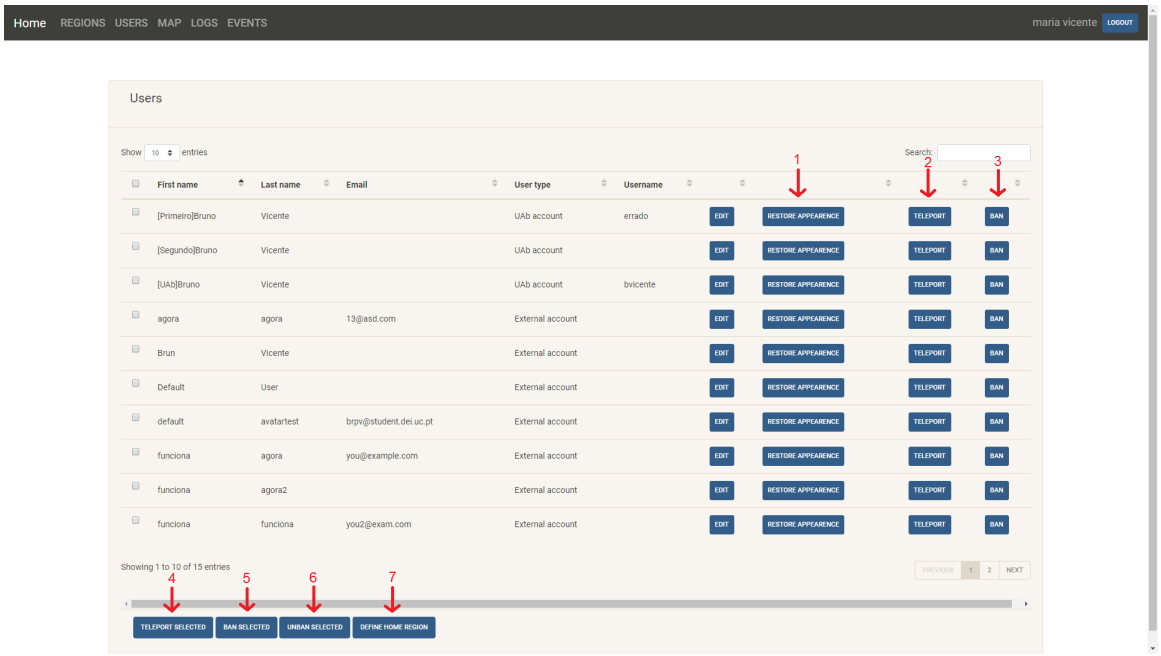

Figura 7.6: Página de listagem de utilizadores.

da página de listagem de utilizadores. Esta funcionalidade permite, clicando no botão com o indicador 2, teleportar apenas um utilizador e, clicando no botão com o indicador 4, teleportar todos os utilizadores selecionados.

O teleporte de utilizadores permite ao administrador transferir um utilizador de uma região para outra ou simplesmente dentro da mesma região ao alterar as coordenadas. Este requisito funcional foi implementado em duas fases: em primeiro lugar é necessário verificar qual a região na qual o utilizador está presente e, em segundo lugar, é preciso enviar um pedido XML-RPC ao servidor através da interface *RemoteAdmin*[8](descrita na sub-secção 4.2.1) de forma a teleportar o utilizador para a região e coordenadas pretendidas.

Para o teleporte funcionar é necessário que o utilizador a teleportar esteja *online*. Caso não esteja o teleporte não funciona e é apresentada uma mensagem de erro a indicar isto mesmo. Caso o administrador tente teleportar vários utilizadores ao mesmo tempo e alguns estejam *online* e outros *offline* é apresentada uma mensagem que indica quais os utilizadores que foram teleportados com sucesso e uma mensagem de erro a indicar quais os que não foram teleportados.

7.3.7 Banir e voltar a autorizar utilizadores

Esta funcionalidade é útil para o administrador quando algum utilizador no mundo virtual tem comportamentos não desejados, por exemplo insulta outros utilizadores no mundo virtual e o administrador necessita de bani-lo de forma a não incomodar mais os outros utilizadores.

Banir ou voltar a autorizar utilizadores é também possível através da página de

[8]http://opensimulator.org/wiki/Remoteadmin:admin_teleport_agent

listagem de utilizadores, ao clicar nos botões com o indicador 3 para banir ou voltar a autorizar um utilizador e através dos botões com os indicadores 5 e 6 para o fazer a todos os utilizadores selecionados.

Para o OpenSimulator, os utilizadores com um nível abaixo de 0 não são permitidos na *grid*, os utilizadores com um nível igual ou superior a 200 são os administradores desta e os utilizadores com o nível 0 são utilizadores comuns[9]. Desta forma, esta funcionalidade apenas altera o nível(eis) do(s) utilizador(es) desejado(s) para -1 caso o pedidos seja para banir o(s) utilizador(es) e altera para 0 caso o pedido seja para voltar a autorizar o acesso ao(s) utilizador(es). Caso um administrador tente banir um utilizador que é administrador uma mensagem de erro é mostrada a indicar que não é possível banir um utilizador que é administrador.

7.3.8 Definir região base para os utilizadores

Esta funcionalidade é de novo acessível através da mesma página através do botão com o indicador 7 (fig. 7.6). Ao clicar neste botão é mostrado um formulário no qual o administrador pode selecionar a região que deseja tornar a região à qual os utilizadores se ligam quando fazem *login*. Desta forma, caso os utilizadores não tenham autorização para se ligarem à região a que se estão a tentar ligar o servidor OpenSimulator irá enviá-los para a região base definida da forma explicada.

7.3.9 Criar região

A criação de uma região é uma funcionalidade fundamental para a criaçao de um mundo virtual. Uma região pode significar um espaço no qual todos os objectos aí presentes estão direcionados a uma determinada unidade curricular ou a um determinado tema.

Para criar uma região o administrador necessita de inserir um nome único para esta região, se quer que a região seja persistente (guarde um ficheiro com os detalhes de ligação da região no simulador), as coordenadas da região e a sua privacidade, como pode ser visto na figura 7.7. Depois de submeter o formulário é utilizado um método do RemoteAdmin que permite a criação de uma região[10]. Caso o nome da região ou as coordenadas já estejam a ser utilizadas por outra região é então apresentada uma mensagem de erro ao utilizador. Caso a região tenha sido criada é apresentada uma mensagem de sucesso.

Implementei também um algoritmo que tenta obter um balanceamento da carga caso haja vários simuladores activos. Este algoritmo irá criar a região no simulador com o menor número de regiões e, como a porta na qual a região escuta pedidos tem de ser única, a porta da nova região será a maior porta utilizada mais 1. Desta forma é garantido que não exista um simulador com todas as regiões e os outros simuladores da *grid* não tenham qualquer região.

[9] http://opensimulator.org/wiki/Userlevel
[10] http://opensimulator.org/wiki/RemoteAdmin:admin_create_region

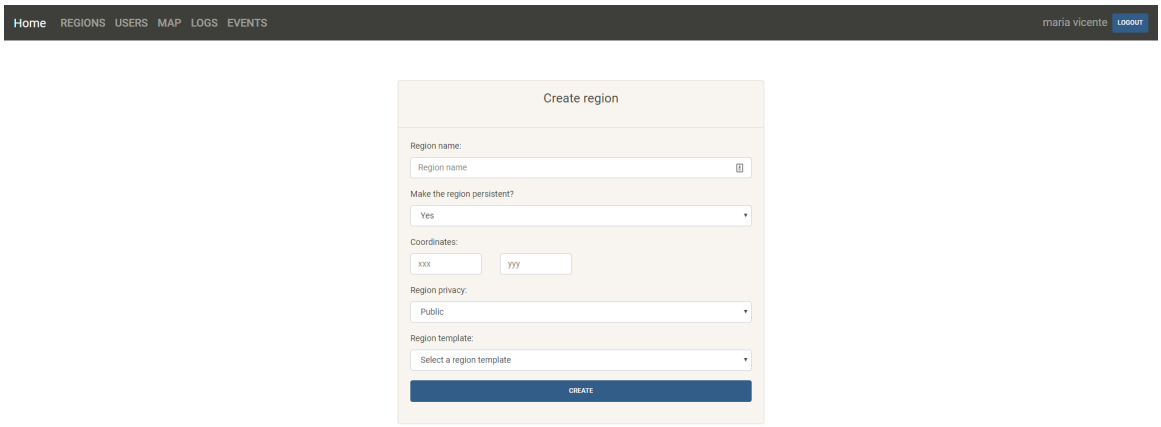

Figura 7.7: Página de criação de região.

Na implementação desta funcionalidade tive algumas dificuldades devido à documentação do OpenSimulator não ser clara e estar errada. Na documentação relativa ao método RemoteAdmin pode ser lido que o parâmetro "*public*"define se uma região é pública ou privada. No entanto esse não é o caso. Ao criar uma região é necessário definir um estado para essa região. Um estado é um grupo de regiões que define algumas das opções e permissões das regiões pertencentes. Desta forma, ao criar uma região pública num estado onde tenha sido anteriormente criada uma região privada a última tornar-se-á uma região pública. Para assegurar que isto não acontece as regiões são criadas em dois estados: "*UAb private estate*"(um estado privado) e *UAb public estate*(um estado público).

7.3.10 Autorizar/Revogar acesso a uma região privada

A autorização e revogação do acesso de utilizadores, tal como visto no cenário 2 da secção 5.1, é útil para, por exemplo, dar acesso a todos os alunos de uma unidade curricular acesso a uma região que contém conteúdos relativos a essa unidade curricular.

Esta funcionalidade pode ser acedida através da página de listagem de regiões, ao clicar no botão com o indicador 1 representado na figura 7.8. Ao clicar no botão é apresentado um pop-up com a listagem de todos os utilizadores de uma região (fig. 7.9). Os utilizadores cuja linha está selecionada sao os utilizadores com acesso à região indicada. Caso o administrador queira alterar os utiliadores com acesso pode selecionar e eliminar a selecção pode simplesmentar clicar na *checkbox* do utilizador respectivo e submeter o formulário, alterando assim a lista de acesso à região.

A implementação foi efectuada com recurso a três métodos RemoteAdmin: *admin_acl_list*[11], *admin_acl_clear*[12] e *admin_acl_add*[13]. O primeiro método é utilizado de forma a obter a lista de utilizadores com acesso a uma região para depois apresentar esta informação ao administrador. O sengundo é utilizado para limpar a lista de acesso a uma região de forma a depois inserir os utilizadores com acesso ao último

[11]http://opensimulator.org/wiki/RemoteAdmin:admin_acl_list
[12]http://opensimulator.org/wiki/RemoteAdmin:admin_acl_clear
[13]http://opensimulator.org/wiki/RemoteAdmin:admin_acl_add

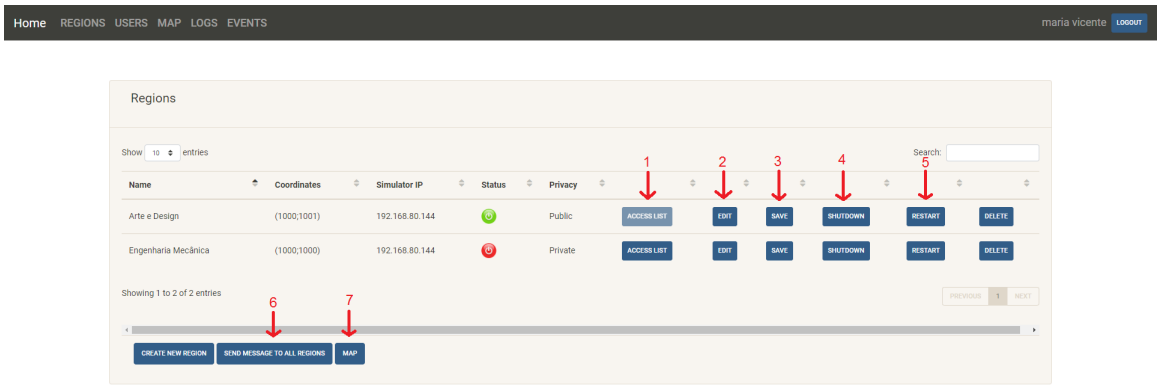

Figura 7.8: Página de listagem de regiões.

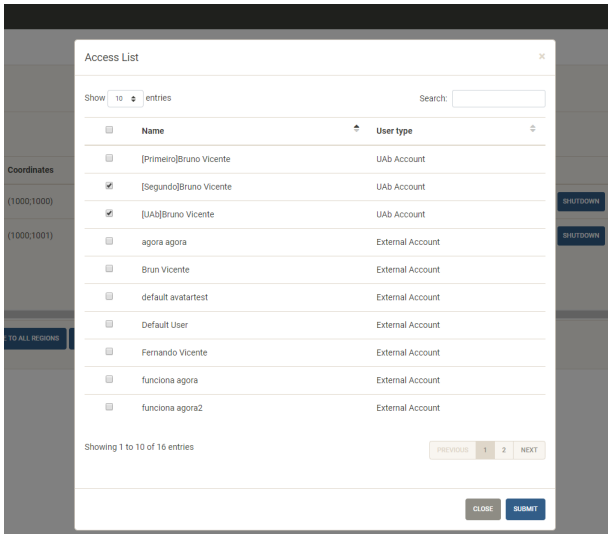

Figura 7.9: Página da lista de acesso a uma região.

comando.

De novo, a dificuldade sentida ao implementar esta funcionalidade foi a informação errada que consta na documentação do último método RemoteAdmin referido. Na documentação é afirmado que o parâmetro "*users*" contém os utilizadores a adicionar à lista de acesso de uma região e que este parâmetro é uma lista com os nomes dos utilizadores a adicionar à lista de acesso. Mesmo o exemplo apresentado no final dessa página *web* está errado sendo que, depois de inspeccionar o código fonte do OpenSimulator consegui identificar que esse parâmetro não é uma lista, mas um dicionário cuja chave pode ser qualquer coisa desde que única no dicionário e cada valor a ser acedido com essa chave de ser o nome de um utilizador a adicionar à lista de acesso de uma região.

7.3.11 Ver estado das regiões

Como administrador da instalação OpenSim devo conseguir ver qual o estado das regiões, nomeadamente se estão ligadas e a aceitar pedidos ou se estão desligadas e, consequentemente, não aceitam pedidos. Esta funcionalidade é importante de

modo a ser mais fácil identificar problemas com as regiões e a conseguir verificar a quais destas os utilizadores se conseguem conectar para aceder à sua representação no mundo virtual.

Quando um administrador acede à página de listagem de regiões(fig. 7.8) ao estado das regiões é apresentado como um circulo amarelo que representa que ainda não pode ser apresentado o estado da região. Quando o estado das regiões pode ser apresentado ele é ilustrado através de um botão verde caso a região esteja *online* e a aceitar pedidos e um botão vermelho caso a região não esteja a aceitar pedidos, estando *online* ou *offline*.

Para a informação acerca do estado das regiões ser apresentada é efectuada uma chamada XML-RPC[14] por região ao respectivo simulador de forma a obter o estado desta. A documentação do OpenSimulator não indica o significado dos valores retornados. Para o entender como este pedido funciona necessitei de inspecionar de novo o código fonte de forma a entender o que os o parâmetro retornado "*health*"significa.

7.3.12 Guardar e carregar template da região

Como visto no cenário 1 na secção 5.1, guardar e voltar a carregar toda a informação que uma região contém é extremamente útil para o uso do OpenSimulator em ambiente pedagógico, nomeadamente para o caso de voltar a criar um mundo virtual de uma unidade curricular quando existe uma reedição desta, ou seja, quando uma unidade curricular de primeiro semestre termina e o responsável por esta unidade curricular pretende voltar a ter o mesmo espaço virtual no ano seguinte.

Esta funcionalidade guarda todos os conteúdos do mundo virtual, isto é, guarda o seu *heightmap* (o mapa de relevo utilizado para renderizar o terreno virtual) e os objectos pertencentes ao mundo virtual como casas, portões, simuladores de dispositivos reais como um giroscópio, etc, permitindo que estes sejam copiados e renderizados noutra região. De notar que o carregamento do *template* numa região já criada elimina todo o conteúdo presente nesta, substituindo-o pelo conteúdo presente no *template*.

Guardar o *template* de uma região pode ser acedido através do botão com o indicador 3 na página de listagem de regiões (fig. 7.8). Já o carregamento de uma região pode ser efectuado ao criar uma região (disponível no ecrã presente na figura 7.7) ou ao editar esta(fig. 7.10), sendo que editar uma região apenas fornece uma forma de alterar o *template* desta.

Para implementar esta funcionalidade utilizei quatro métodos da interface RemoteAdmin:

1. *admin_save_heightmap*[15] para guardar o mapa de relevo da região;

2. *admin_save_xml*[16] para guardar os conteúdos da região;

[14]http://opensimulator.org/wiki/RemoteAdmin:admin_region_query
[15]http://opensimulator.org/wiki/RemoteAdmin:admin_save_heightmap
[16]http://opensimulator.org/wiki/RemoteAdmin:admin_save_xml

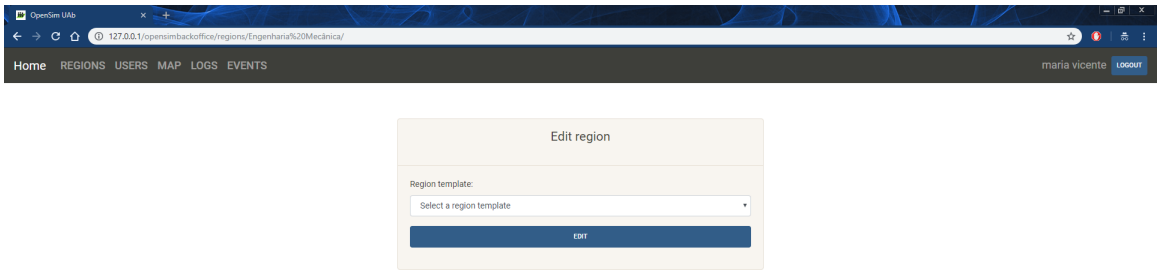

Figura 7.10: Página de editar uma região.

3. *admin_load_heightmap*[17] para carregar o mapa de relevo de uma região noutra região;

4. *admin_load_xml*[18] para para carregar os conteúdos de uma região noutra região;

De forma a ser possível copiar os conteúdos de uma região presente num simulador a executar numa máquina para outra presente noutro simulador a executar noutra máquina é necessário encontrar uma solução para aceder aos ficheiros presentes em redes diferentes. Os dois métodos utilizados para carregar os conteúdos da região permitem apenas a utilização de um caminho para um ficheiro em disco ou um URL no qual está presente o ficheiro. Desta forma a solução encontrada foi utilizar um servidor HTTP de forma a servir os ficheiros entre redes diferentes. Esta solução é explicada com maior detalhe no manual de instalação.

7.3.13 Desligar e recomeçar região

O administrador da instalação OpenSimulator pode necessitar de desligar uma região e voltar a ligá-la noutra altura para, por exemplo, diminuir a carga computacional de um simulador desligando uma região que não está a ser utilizada e não é necessária e voltar a ligar a região quando esta for precisa ou o simulador tiver algum poder computacional disponível.

Desligar e recomeçar uma região pode ser feito através dos botões com os indicadores 4 e 5, respectivamente, através da página de listagem de regiões (fig. 7.8). Uma mensagem de erro é mostrada caso o administrador tente desligar uma região que já esteja desligada e uma mensagem de sucesso é mostrada ao recomeçar uma região com sucesso.

Para a implementação deste requisito funcional recorri aos métodos *admin_close_region*[19] e *admin_restart*[20] para desligar e recomeçar as regiões. O segundo método apenas funciona caso a região esteja *online*, fazendo um *restart* à região. Quando a região é desligada através do método referido esta é eliminada do seu simulador, no entanto a sua informação continua guardada na base de dados. Desta forma, a solução

[17] http://opensimulator.org/wiki/RemoteAdmin:admin_load_heightmap
[18] http://opensimulator.org/wiki/RemoteAdmin:admin_load_xml
[19] http://opensimulator.org/wiki/RemoteAdmin:admin_close_region
[20] http://opensimulator.org/wiki/RemoteAdmin:admin_restart

encontrada para este problema foi voltar a criar a região através do método RemoteAdmin referido na sub-secção 7.3.9 com o mesmo UUID da região que foi desligada, de forma a manter toda a informação da região que foi desligada.

7.3.14 Enviar mensagem para todas as regiões

Caso o administrador de sistemas queira informar todos os utilizadores presentes na *grid* da Universidade Aberta de que o OpenSim irá ser desligado, tal como representado no cenário 3, esta funcionalidade permite fazê-lo facilmente. Esta é acessível através da página de listagem de regiões(fig. 7.8) ao clicar no botão com o indicador 6. Um formulário que consiste apenas na mensagem a enviar aparecerá e uma mensagem será enviada a todas as regiões que estejam *online* ao submeter o formulário(fig. 7.11).

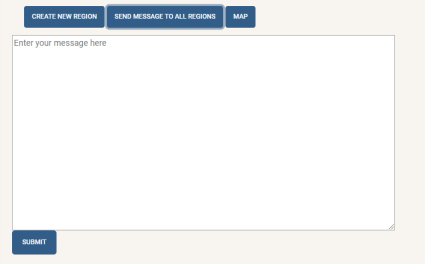

Figura 7.11: Formulário de envio de mensagem a todas as regiões.

Para enviar uma mensagem a todas as regiões que estão *online* utilizei o método *admin_broadcast*[21] do RemoteAdmin que permite o envio de uma mensagem a uma dada região. Para o fazer para todas as regiões é necessário percorrer a tabela da base de dados relativa às regiões[22] e enviar uma mensagem a cada uma das regiões aí presentes.

7.3.15 Mapa

Um mapa que mostre cada uma das regiões que estão *online* na *grid* é útil, como referido e explicado no cenário 1 da sub-secção 5.1, para verificar quais as coordenadas nas quais estão presentes as regiões e se uma região é criada na localização pretendida. Sendo um administrador podemos aceder ao mapa ao clicar no botão com o indicador 7 na página de listagem de regiões(fig. 7.8) ou ao clicar em "*Map*"no cabeçalho presente em toda a aplicação.

A figura 7.12 apresenta o mapa da *grid* obtido. Ao clicar numa região é apresentado o nome desta e as coordenadas desta no mundo virtual.

Esta funcionalidade foi implementada adaptando uma aplicação que apresenta um mapa para instalações *grid* do OpenSimulator presente no GitHub[23] e desenvolvida

[21]http://opensimulator.org/wiki/RemoteAdmin:admin_broadcast
[22]http://opensimulator.org/wiki/Regions_(database_table)
[23]https://github.com/

Figura 7.12: Página de apresentação do mapa das regiões.

pelo utilizador talat77[24]. Esta aplicação integra os *heightmaps* das regiões com a API do Google Maps[25]. O problema desta aplicação é o facto de não ser dinâmico, isto é, é necessário guardar as imagens(*heightmaps*) num determinado directório de forma a renderizá-las no mapa logo mudanças realizadas no mundo virtual não são representadas no mapa. A solução encontrada para este problema foi realizar um pedido HTTP a cada simulador do OpenSimulator de forma a ter uma representação real da região no mundo virtual uma vez que cada simulador, ao receber um pedido HTTP específico[26], gera um *heightmap* actualizado da região. Foi também necessário adaptar o código *javascript* para alterar algumas opções e eliminar algumas funcionalidades que não eram necessárias para este projecto.

7.3.16 Guardar estado dos simuladores

Este requisito foi encontrado ao longo do desenvolvimento do projecto. Este resulta da necessidade de guardar quais as regiões que o simulador deve ligar quando este é desligado e volta a ser ligado.

A forma como o simulador está pré-configurado indica ao processo(OpenSim.exe) que as regiões a iniciar estão num determinado directório no formato ".ini"(um formato utilizado em ficheiros de configuração[27]). O problema aqui presente deve-se ao facto de ao eliminar uma região esta é eliminada do simulador, mas o seu ficheiro continua presente no sistema de ficheiros do simulador. Como consequência, quando um simulador é desligado e volta a iniciar são criadas todas as regiões presentes nos ficheiros de configuração e assim são novamente criadas regiões que tinham sido

[24]https://github.com/talat77/opensimmaps
[25]https://www.google.com/maps
[26]http://opensimulator.org/wiki/Map
[27]http://linuxcnc.org/docs/2.6/html/config/ini_config.html

eliminadas anteriormente.

De forma a solucionar o problema referido, o simulador, quando inicia, envia um pedido HTTP ao *backoffice* de forma a obter a lista das regiões que deve iniciar e obtém a resposta em formato XML. Para o *backoffice* fornecer esta lista de regiões aos simuladores todas as informações são guardadas em formato XML quando é criada uma região numa tabela da base de dados criada para o efeito e quando é eliminada uma região é também eliminada a entrada dessa região da tabela da base de dados.

7.3.17 Eventos

Os eventos a ocorrer no mundo virtual podem ter as mais variadas intenções: por um lado podem ser sessões de treino ou avaliação que ocorrerão no mundo virtual às quais apenas têm acesso os alunos de uma determinada unidade curricular, por outro lado podem ser sessões de esclarecimento acerca do uso do mundo virtual em que todos os utilizadores podem participar.

Para a implementação de eventos foi necessário criar novas tabelas na base de dados de forma a suportar as informações referentes aos eventos, as pessoas com permissão para aceder a estes eventos e o controlo de presenças nestes eventos.

Listagem de eventos

A listagem de eventos, representada na figura 7.13, pode ser consultada ao clicar em "*Events*"no cabeçalho da aplicação. Caso o utilizador que acedeu à página de listar eventos tenha permissões de administração são mostrados todos os eventos ocorridos de forma cronológica e são também mostrados botões de controlo sobre estes eventos. Se o evento já terminou ou está a decorrer são mostradas as opções de eliminar o evento e de obter as presenças desse evento(descrito na sub-secção seguinte, 7.3.17), representadas pelos indicadores 3 e 2, respectivamente. Caso o evento ainda não tenha começado o administrador tem acesso apenas às informações do evento, não sendo apresentadas as opções de administração descritas.

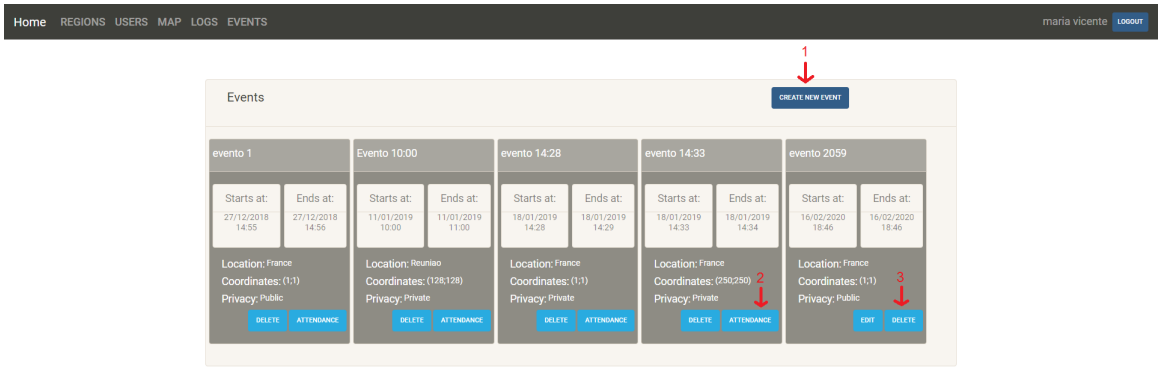

Figura 7.13: Página de listagem de eventos.

Criar eventos

A página de criação de eventos(fig. 7.14) é apresentada depois de clicar no botão com o indicador 1 na página de listagem de eventos(fig. 7.13). Este botão é apenas apresentado caso o utilizador seja administrador do sistema. Na página de criação de eventos o administrador necessita de inserir o nome do evento a criar, a data e hora a que este começa e termina, a região e as coordenadas na regiões na qual irá decorrer e a privacidade do evento. De notar que só é possível criar eventos privados em regiões privadas e públicos em regiões públicas.

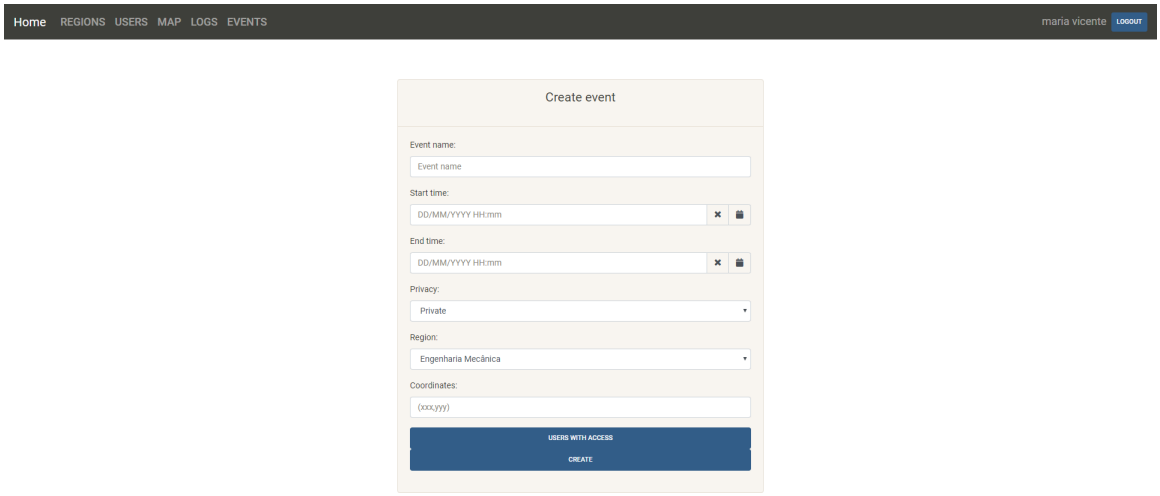

Figura 7.14: Página de criação de um evento.

Para implementar a selecção da data recorri à biblioteca externa *Tempus Dominus*[28]. Esta biblioteca permite definir que a data e hora de começo do evento não pode ser anterior ao momento actual e a data e hora em que termina tem de ser superior à do começo do evento.

Quando o evento é definido como privado aparece um botão acima do botão de criação do evento que permite definir que utilizadores têm acesso ao evento. Após a criação de um evento privado é necessário criar automaticamente o evento no mundo virtual. Isto resume-se a guardar a informação presente na lista de acesso da região antes do evento ocorrer, modificar a lista de acesso dessa região para a lista de utilizadores com acesso à região e, após a conclusão do evento, é necessário repôr a lista de acesso original. De forma a automatizar estes passos recorri ao Celery e ao Redis. Quando um evento privado é criada uma tarefa que irá executar 5 minutos antes da data e hora de começo do evento, guardando a lista de acesso da região e alterando esta para a lista de acesso do evento. Caso isto seja feito com sucesso a tarefa cria outra tarefa que irá ser executada 5 minutos depois da data e hora de *terminus* do evento, a de eliminação do evento do mundo virtual, que será responsável por repôr a lista de acesso da região.

[28]https://tempusdominus.github.io/bootstrap-4/

Editar e eliminar eventos

A página de editar eventos é igual à página de criação de eventos, mas é apresentada pré-preenchida com os dados de um determinado evento. Ao editar um evento que esteja a decorrer é necessário repôr a lista de acesso da regiao e verificar se existe alguma tarefa de criação e eliminação do evento do mundo virtual escalonadas para executar aquando do começo e conclusão do evento e eliminar estas tarefas. Caso o evento esteja planeado para o futuro é apenas necessário eliminar as tarefas escalonadas. Depois disto é apenas necessário editar as informações do evento e escalonar duas novas tarefas que irão executar exactamente como explicado na criação de eventos.

A eliminação de eventos tem de executar exactamente os mesmos passos referidos na edição de eventos, mas eliminando as informações relativas ao evento em vez de as editar e não escalonando as tarefas a executar para a criação e eliminação do evento no mundo virtual.

Controlo de presenças

O controlo de presenças em eventos é uma das funcionalidades mais importantes deste projecto. Um administrador consegue consultar as presenças ao clicar no botão com o indicador 2 na figura 7.13, que representa a listagem de eventos. As presenças nas sessões são constituidas por uma tabela com quatro colunas: o nome do utilizador, o tipo de utilizador, a data e hora em que entrou no evento e a hora em que saiu do evento, como se pode ver na figura 7.15. O administrador pode então ordenar a lista de entradas na tabela pelo nome do utilizador. A decisão de mostrar várias entradas nas tabelas por cada utilizador foi tomada de forma a que não seja considerada uma presença uma entrada no simulador e uma saída 30 segundos após a entrada.

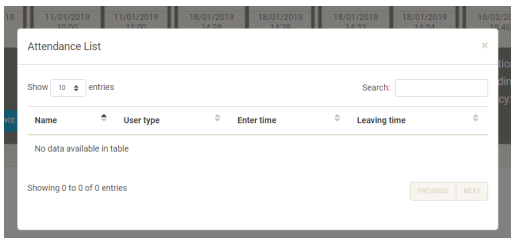

Figura 7.15: Página de apresentação das presenças de um evento.

A implementação do controlo de presenças foi executada em duas fases: o desenvolvimento de uma função no *backoffice* que aceita um pedido HTTP com o conteúdo em JSON e o desenvolvimento de um *script* para o OpenSimulator que envia as informações de presenças de um determinado evento ao *backoffice*. A função que controla as presenças no *backoffice* recebe o pedido e verifica se o IP de origem pertence à lista de simuladores registados na *grid* e, caso esteja, processa o conteúdo JSON e guarda as presenças recebidas na base de dados. Já o *script* desenvolvido necessita de ser adicionado a um objecto presente no mundo virtual de forma a poder ser executado. Para isto é necessário que um administrador o faça através

do *viewer* do OpenSimulator pois não existe nenhuma forma de inserir um objecto numa determinada região em instalações *grid*.Este *script* foi desenvolvido através da linguagem de *scripting* usada no OpenSimulator, a LSL(Linden Script Language), e verifica a cada 30 segundos a lista de utilizadores presentes na região em que está a executar enviando os novos *logins* e os novos *logouts* ao *backoffice* para serem guardados na base de dados. Para o correcto funcionamento do *script* necessitei de alterar algumas configurações do OpenSimulator(explicadas em detalhe no manual de instalação) e necessitei de aprender a linguagem descrita, LSL.

7.3.18 Logs

Esta funcionalidade facilita a identificação de problemas relativos à instalação OpenSimulator sem ter de realizar uma conexão por SSH, por exemplo. Na figura 7.16 podemos ver os ficheiros de logs dos servidores Opensimulator agrupados por IP e divididos entre o servidor R.O.B.U.S.T. e os simuladores(como pode ser visto no lado esquerdo) e o conteúdo do ficheiro(pode ser visto do lado direito), no qual apenas são apresentadas 100 linhas por página de forma a não imputar demasiado poder computacional ao computeador do administrador.

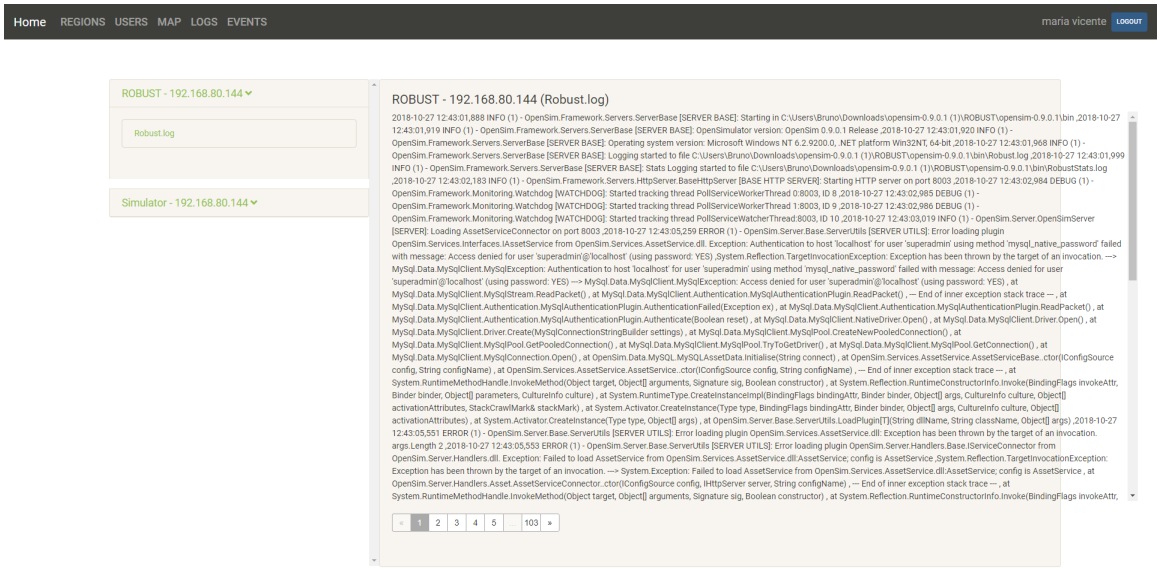

Figura 7.16: Página de visualização dos *logs* dos servidores.

O OpenSimulator não tem qualquer tipo de suporte para a visualização dos ficheiros de logs dos servidores e, por isso, necessitei de encontrar uma forma de implementar esta funcionalidade. A solução encontrada foi disponibilizar os ficheiros através do protocolo HTTP, utilizando o servidor Apache[29] para o efeito.

[29]https://www.apache.org/

7.3.19 Sistema de suporte

O sistema de suporte permite aos utilizadores do OpenSimulator procurar ajuda e permite uma linha de comunicação directa entre os utilizadores e os administradores do sistema, como ilustrado no cenário 3 presente na secção 5.1.

A página de suporte do sistema pode ser vista na figura 7.17. Para enviar uma mensagem aos administradores do sistema o utilizador necessita de adicionar o email para o qual o administrador irá responder à mensagem, o assunto e a mensagem a enviar. Caso o utilizador seja externo à Universidade Aberta o email já estará pré-preenchido com o email com o qual foi efectuado o registo. Ao submeter o formulário o *backoffice* enviará um email para o endereço descrito no ficheiro *support_email.json* de forma a que o(s) administrador(es) do sistema consigam consultar as mensagens e responder ao pedido para o endereço de email referido, o que será automático ao clicar em "Responder"no cliente de correio electrónico utilizado.

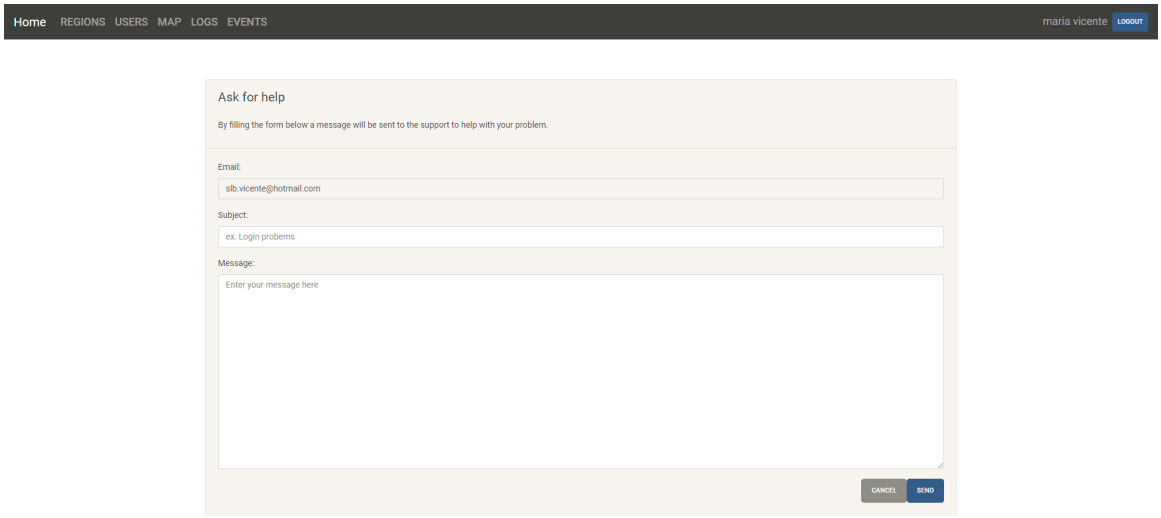

Figura 7.17: Página de suporte a utilizadores.

Capítulo 8

Testes

Neste capítulo é apresentada a metodologia utilizada na fase de testes do projecto. A fase de testes de software é uma das fases mais importantes no desenvolvimento de software e tem como objectivo identificar *bugs* que podem ou não levar a falhas do sistema para, posteriormente, ser possível corrigi-los.

As funcionalidades do sistema apenas foram aceites quando todos os testes efectuados tiveram o resultado e/ou comportamento esperado. Caso algum teste não tivesse o resultado esperado o *bug* era identificado e corrigido e todos os testes eram executados novamente.

Este capítulo está dividido em 4 secções: *black box testing* inclui os testes de robustez, *white box testing* inclui os testes unitários e de integração efectuados ao *backoffice web*, testes à responsividade do sistema onde é testado este atributo de qualidade e testes à segurança do sistema.

8.1 *Black box testing*

Os testes *black box* são testes nos quais não existe acesso ao código fonte do sistema. Desta forma o principal foco deste tipo de testes é verificar se o *output* obtido através da introdução de um *input* é igual ao *output* esperado.

Todos os testes de *black box* efectuados foram automatizados de forma a não ser necessário a indrodução de *inputs* multiplas vezes ao longo desta fase. Esta automatização foi efectuada com recurso ao Selenium[1] e ao Allure[2]. O primeiro é uma tecnologia que permite a automatização de uma selecção de *browsers*(o *browser* utilizado foi o Google Chrome[3]) tornando possível automatizar totalmente a introdução de informações em formulários e o envio destes. O Allure é uma ferramenta que permite observar o resultado dos testes através de uma página HTML, como está ilustrado na figura 8.1. A automatização dos testes tem inúmeras vantagens como a velocidade de execução dos testes pois é mais rápida a introdução de *inputs* por

[1]https://www.seleniumhq.org/
[2]http://allure.qatools.ru/
[3]https://www.google.com/chrome/

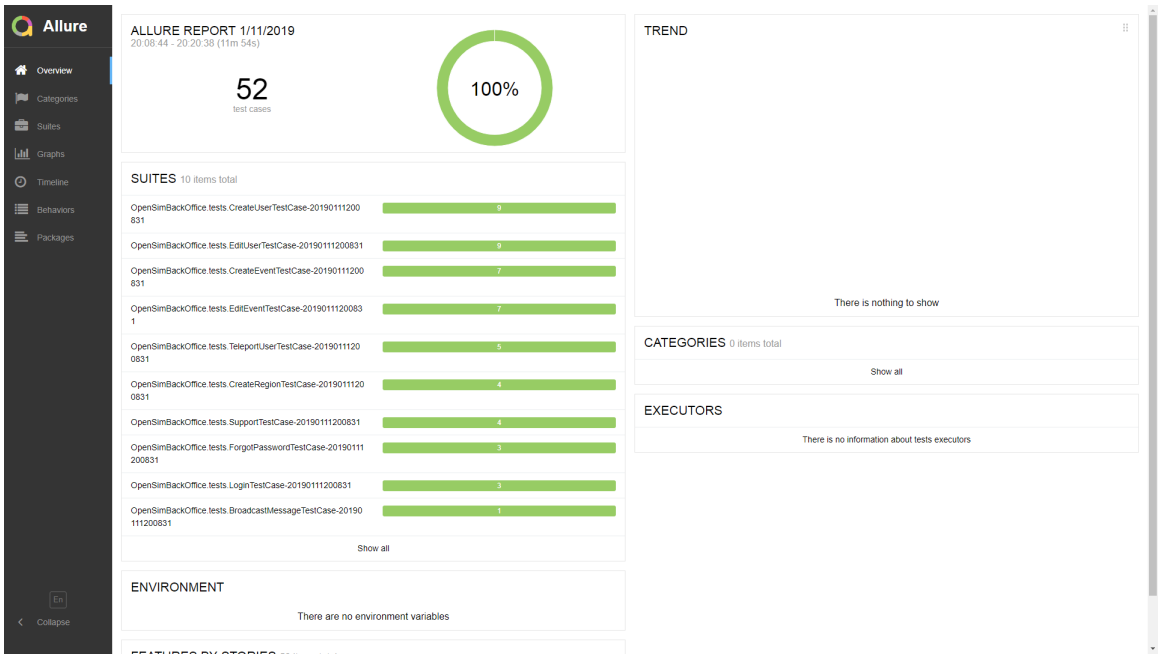

Figura 8.1: Interface *web* do Allure.

parte de *software* do que a introdução manual e a facilidade de repetição dos testes.

8.1.1 Testes de robustez

Os testes de robustez servem para testar a robustez de um sistema quando são introduzidos *inputs* inválidos. Apesar de o sistema ser principalmente direcionado aos administradores da instalação OpenSim foram efectuados testes de robustez a todas as funcionalidades do sistema. De notar que os *inputs* válidos e inválidos variam em cada funcionalidade do *backoffice*.

Autenticação de utilizadores		
Campos		Resultado
username	password	
Vazio	OK	Passou
OK	Vazio	Passou
Mal Formatado	OK	Passou

Tabela 8.1: Resultados dos testes de robustez (Autenticação de utilizadores).

Criar utilizador						
Campos						Resultado
first name	last name	email	avatar	password	confirm password	
Vazio	OK	OK	OK	OK	OK	Passou
OK	Vazio	OK	OK	OK	OK	Passou
OK	OK	Vazio	OK	OK	OK	Passou
OK	OK	OK	Vazio	OK	OK	Passou
OK	OK	OK	OK	Vazio	OK	Passou
OK	OK	OK	OK	OK	Vazio	Passou
Mal Formatado	OK	OK	OK	OK	OK	Passou
OK	Mal Formatado	OK	OK	OK	OK	Passou
OK	OK	Mal Formatado	OK	OK	OK	Passou

Tabela 8.2: Resultados dos testes de robustez (Criar utilizador).

Editar utilizador						
Campos						Resultado
first name	last name	email	avatar	password	confirm password	
Vazio	OK	OK	OK	OK	OK	Passou
OK	Vazio	OK	OK	OK	OK	Passou
OK	OK	Vazio	OK	OK	OK	Passou
OK	OK	OK	Vazio	OK	OK	Passou
OK	OK	OK	OK	Vazio	OK	Passou
OK	OK	OK	OK	OK	Vazio	Passou
Mal Formatado	OK	OK	OK	OK	OK	Passou
OK	Mal Formatado	OK	OK	OK	OK	Passou
OK	OK	Mal Formatado	OK	OK	OK	Passou

Tabela 8.3: Resultados dos testes de robustez (Editar utilizador).

Esquecer password			
Campos			Resultado
first name	last name	email	
Vazio	OK	OK	Passou
OK	Vazio	OK	Passou
OK	OK	Vazio	Passou
Mal Formatado	OK	OK	Passou
OK	Mal Formatado	OK	Passou
OK	OK	Mal Formatado	Passou

Tabela 8.4: Resultados dos testes de robustez (Esquecer password).

Criar região						
Campos						Resultado
region name	persistent	xxx	yyy	privacy	template	
Vazio	OK	OK	OK	OK	OK	Passou
OK	OK	Vazio	OK	OK	OK	Passou
OK	OK	OK	Vazio	OK	OK	Passou
OK	OK	Mal Formatado	OK	OK	OK	Passou
OK	OK	OK	Mal Formatado	OK	OK	Passou

Tabela 8.5: Resultados dos testes de robustez (Criar região).

Enviar mensagem a todas as regiões	
Campos	Resultado
message	
vazia	Passou

Tabela 8.6: Resultados dos testes de robustez (Enviar mensagem a todas as regiões).

Teleportar utilizador				
Campos				Resultado
region	xxx	yyy	zzz	
OK	Vazio	OK	OK	Passou
OK	OK	Vazio	OK	Passou
OK	Letras	OK	OK	Passou
OK	OK	Letras	OK	Passou
OK	OK	OK	Letras	Passou

Tabela 8.7: Resultados dos testes de robustez (Teleportar utilizadores).

Criar evento						
Campos						Resultado
event name	start time	end time	privacy	region	coordinate	
Vazio	OK	OK	OK	OK	OK	Passou
OK	Vazio	OK	OK	OK	OK	Passou
OK	OK	Vazio	OK	OK	OK	Passou
OK	OK	OK	OK	OK	Vazio	Passou
OK	Mal Formatado	OK	OK	OK	OK	Passou
OK	OK	Mal Formatado	OK	OK	OK	Passou
OK	OK	OK	OK	OK	Mal Formatado	Passou

Tabela 8.8: Resultados dos testes de robustez (Criar evento).

Acesso ao evento	
Campos	Resultado
users	
Vazio	Passou

Tabela 8.9: Resultados dos testes de robustez (Acesso ao evento).

Editar eventos						
Campos						Resultado
event name	start time	end time	privacy	region	coordinate	
Vazio	OK	OK	OK	OK	OK	Passou
OK	Vazio	OK	OK	OK	OK	Passou
OK	OK	Vazio	OK	OK	OK	Passou
OK	OK	OK	OK	OK	Vazio	Passou
OK	Mal Formatado	OK	OK	OK	OK	Passou
OK	OK	Mal Formatado	OK	OK	OK	Passou
OK	OK	OK	OK	OK	Mal Formatado	Passou

Tabela 8.10: Resultados dos testes de robustez (Editar evento).

Sistema de suporte			
Campos			Resultado
email	assunto	mensagem	
Vazio	OK	OK	Passou
OK	Vazio	OK	Passou
OK	OK	Vazio	Passou
Mal Formatado	OK	OK	Passou

Tabela 8.11: Resultados dos testes de robustez (Sistema de suporte).

Como podemos observar na última coluna das tabelas apresentadas anteriormente, todas as funcionalidades passaram nos testes efectuados.

8.2 *White-box testing*

Os testes *white box* são vistos complementares em relação aos testes *black box*. Os testes *white box* são desenhados e efectuados com conhecimento do código fonte e tem em conta o comportamento do sistema e o *output*, isto é, tem em conta o *throughput* do sistema. Neste projecto foram efectuados dois tipos de testes: testes unitários e testes de integração.

8.2.1 Testes unitários

Os testes unitários têm como objectivo testar unidades de código separadamente de forma a verificar a existência de *bugs* nessa mesma unidade. Neste projecto foram efectuados testes unitários não só válidos como também testes unitários em que o *input* é aceite no *front-end*, mas é um caso especial que tem de ser tratado no *back-end*.

Estes testes também foram totalmente automatizados de forma a facilitar a repetição destes. O processo de automatização é igual ao descrito na secção 8.1.

Requisito Funcional	Descrição do teste	Resultado esperado	Resultado obtido
RF-1	Criar utilizador com nome já utilizado	Erro	Erro
	Criar utilizador com email já utilizado	Erro	Erro
	Criar utilizador único	OK	OK
RF-2	Editar utilizador OpenSim com password errada	Erro	Erro

	Editar utilizador OpenSim com confirmação de password errada	Erro	Erro
RF-3	Esquecer palavra passe Uab	OK	OK
	Esquecer palavra passe OpenSim	OK	OK
RF-9	Criar região mesmo nome	Erro	Erro
	Criar região coordenadas já usadas	Erro	Erro
RF-14	Mudar lista de acesso de região offline	Erro	Erro
RF-?	Desligar região que está offline	Erro	Erro
RF-20	Criar evento no passado	Erro	Erro
	Criar evento que acabe antes de começar	Erro	Erro
	Criar evento público	OK	OK
RF-21	Editar evento ocurrido	Erro	Erro
	Editar evento a ocorrer	Erro	Erro
	Editar evento para o passado	Erro	Erro
	Editar evento que acabe antes de começar	Erro	Erro
RF-22	Eliminar evento	OK	OK
RF-23	Teleportar utilizador offline	Erro	Erro
	Teleportar utilizador coordenadas maiores que 256	Erro	Erro
	Teleportar utilizador coordenadas menores que 0	Erro	Erro
RF-25	Login com Uab	OK	OK
	Login com conta opensim	OK	OK

Tabela 8.12: Resultados dos testes unitários.

Os testes unitários foram considerados como passados quando o resultado esperado é igual ao resultado obtido. Como se pode identificar na tabela apresentada, todas as funcionalidades passaram nos seus respectivos testes unitários.

8.2.2 Testes de integração

De forma a testar a integração do *backoffice web* com o OpenSimulator foram efectuados testes de integração. Os testes de integração têm como objectivo validar a interação de vários módulos de *software*. Neste caso, este tipo de testes teve como objectivo validar a interacção entre o *backoffice web* e o OpenSimulator.

A execução destes testes foi feita em vários passos. Em primeiro lugar foi necessário ligar toda a instalação OpenSimulator(o servidor a executar os serviços R.O.B.U.S.T. e um simulador OpenSimulator). Depois deste passo foram executadas as acções no *backoffice web* que têm impacto no mundo virtual e, entre cada acção executada no *backoffice*, efectuei o *login* no OpenSimulator através do *viewer* de forma a validar se o impacto no mundo virtual foi o esperado.

Requisito Funcional	Descrição do teste	Resultado esperado	Resultado obtido
RF-1, RF-4	Criar utilizador OpenSim	OK	OK
RF-2	Editar utilizador UAb	Apenas muda o avatar	Apenas muda o avatar
	Editar utilizador OpenSim	OK	OK
RF-7	Ver mapa do mundo virtual	OK	OK
RF-9	Criar região com template	OK	OK
	Criar região cujo ficheiro de conf. é guardado no simulador	OK	OK
	Criar região privada	OK	OK
RF-10	Editar região (template)	OK	OK
RF-11	Recomeçar regiao que está online	OK	OK
	Recomeçar regiao que está offline	OK	OK
RF-12	Guardar template de região	OK	OK

RF-?	Eliminar região que está offline	OK	OK
	Eliminar região que está online	OK	OK
RF-14	Mudar lista de acesso de região online	OK	OK
RF-17	Ver logs do sistema	OK	OK
RF-18	Definir região home	OK	OK
RF-19	Enviar mensagem para todas as regiões	OK	OK
RF-20, RF-27	Criar evento privado	OK	OK
	Criar evento público	OK	OK
RF-21	Editar evento público para privado	OK	OK
	Editar evento privado para público	OK	OK
RF-22	Eliminar evento		
RF-23	Teleportar utilizador para região offline	Erro	Erro
	Teleportar utilizador	OK	OK
	Teleportar utilizador com coordenadas z vazias (permitido)	OK	OK
RF-24	Criar evento automaticamente no mundo virtual	OK	OK
	Terminar evento automaticamente no mundo virtual	OK	OK
RF-28	Controlar presenças do evento	OK	OK
RF-31	Restaurar aparencia utilizador	OK	OK
RF-32	Enviar mensagem de suporte	OK	OK
RF-35	Ver eventos como administrador	Vê todos os eventos	Vê todos os eventos

	Ver eventos como utilizador comum	Apenas vê eventos a que tem acesso	Apenas vê eventos a que tem acesso
RF-36	Banir utilizador Voltar a autorizar utilizador	OK OK	OK OK
RF-37	Desligar região que está online	OK	OK

Tabela 8.13: Resultados dos testes de integração.

Os testes de integração foram considerados como passados quando o resultado esperado é igual ao resultado obtido. Como se pode identificar na tabela 8.13, todas as funcionalidades passaram nos seus respectivos testes de integração.

8.3 Testes à responsividade

A reponsividade de um sistema refere-se à capacidade deste completar tarefas num determinado tempo. Os testes à responsividade do sistema foram efectuados de forma a avaliar se este respeita os cenários apresentados nas tabelas **??** e **??**.

Para validar a responsividade do sistema foram verificadas todas as funcionalidades e foi verificado se o *feedback* após efectuar uma acção é menor do que 3 segundos. Caso o pedido a efectuar para executar essa acção seja assíncrono uma mensagem deve ser mostrada em menos de 3 segundos de forma a indicar ao utilizador que o pedido está a ser efectuado e qual o tempo máximo previsto para a conclusão da acção. Na figura 8.2 podemos ver um exemplo da mensagem apresentada.

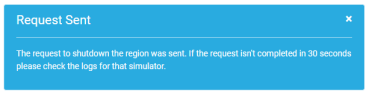

Figura 8.2: Exemplo de mensagem de *feedback* imediato.

Caso o pedido seja síncrono uma mensagem deve ser mostrada ao utilizador em menos de 3 segundos com o resultado da acção. Nas figuras 8.3 e 8.4 são apresentados exemplos das mensagens apresentadas em caso de erro e em caso de sucesso, respectivamente.

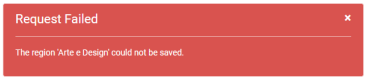

Figura 8.3: Exemplo de mensagem de erro.

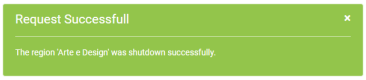

Figura 8.4: Exemplo de mensagem de sucesso.

8.4 Testes à segurança

Apesar da maior parte das funcionalidades do *backoffice* sejam direccionadas aos administradores do sistema existem algumas funcionalidades disponíveis a todos os utilizadores e que necessitam que estes introduzam dados num formulário. Desta forma, a segurança do sistema é extremamente importante uma vez que o utilizador pode conseguir obter informações do sistema como nomes de utilizadores, *hashs* de *passwords*, etc.

Para avaliar a segurança do sistema foram efectuados testes a dois tipos de ataques: *SQL Injection* e *Cross-site scripting*.

8.4.1 *SQL Injection*

De forma a avaliar a resposta do sistema a um ataque *SQL injection* os testes foram efectuadas na página de recuperação de credenciais pois todos os utilizadores têm acesso a esta página.

Na tabela 8.14 são apresentados os ataques efectuados do tipo *SQL Injection* e os seus resultados.

SQL injection		
Ataque	Descrição	Resultado
' OR '1' = '1	A adição de uma condição sempre verdadeira e uma disjunção resulta numa condição sempre verdadeira. Isto possibilita a selecção de todas as entradas na tabela.	Erro
'; INSERT INTO users_table ...	Esta técnica é chamada *Stacked queries* e serve para executar duas *queries* na mesma chamada à base de dados. Esta *query* especificamente serve para inserir um utilizador na tabela de utilizadores possivelmente com permissões de administrador.	Erro

Tabela 8.14: Resultados dos testes de segurança(*SQL injection*).

Estes ataques não têm sucesso devido a todas as operações à base de dados serem efectuadas através de um *Object-relational mapping*(ORM) o que faz com que todas

as *queries* efectuadas à base de dados sejam feitas através de *prepared statements* que são *queries* que sanitizam os dados introduzidos nestas. De notar que a tabela *users_table* não existe na base de dados.

8.4.2 *Cross-site scripting*(XSS)

Os ataques de *corss-site scripting* têm como objectivo executar um *script* no *browser* dos utilizadores quando estes acedem a uma página ou executam uma acção. Os testes a este tipo de ataque foram efectuados na página de criação de eventos e na página de criação de utilizadores. Na tabela 8.15 podem ser verificados os testes efectuados.

SQL injection		
Ataque	Descrição	Resultado
<script>alert("You just got atacked!");</script>	Cria um *popup* com o texto "You just got atacked!"quando um utilizador carrega uma página que contém a string da primeira coluna.	Erro
<body onload=alert('You just got atacked!')>	Cria um *popup* com o texto "You just got atacked!"quando um utilizador carrega uma página que contém a string da primeira coluna. Utiliza uma *tag* HTML diferente do primeiro ataque.	Erro
<b onmouseover=alert('You just got atacked!')>click me!	Cria um *popup* com o texto "You just got atacked!"quando um utilizador passa com o rato em cima do texto "click me!".	Erro

Tabela 8.15: Resultados dos testes de segurança(*Cross-site scripting*).

Capítulo 9

Planeamento e riscos

O planeamento é um elemento essencial para a gestão de um projecto sendo que um planeamento mal efectuado pode levar a um aumento no orçamento do projecto e até levar à não conclusão deste. Também associado ao planeamento estão os riscos presentes no desenvolvimento de um projecto. Estes representam o que pode correr mal ao longo da execução dos planos efectuados, qual a probabilidade de se verificarem, qual o impacto no projecto e qual o plano de mitigação para estes. Assim sendo, nesta secção apresento as estimativas e os planos efectuados para a realização do projecto assim como os riscos associados a este.

9.1 Planeamento

9.1.1 Primeiro semestre

No primeiro semestre o trabalho passou por preparar a fase de implementaçao do sistema que acontecerá no segundo semestre. Como se pode ver na figura 9.1, comecei por estudar a tecnologia OpenSimulator e perceber qual o estado da arte de modo a percepcionar como funciona esta tecnologia e o que é expectável do sistema a desenvolver. Esta tarefa estava prevista ser executada em 4 semanas, mas, devido a muitas dificuldades na instalação das ferramentas de gestão actuais, acabei por executar esta tarefa em 6 semanas. De seguida foi escrito um artigo que estuda as ferramentas de gestão actuais e que as avalia consoante as funcionalidades que estas dsponibilizam. Esta tarefa também demorou mais do que o expectável, na minha opinião, pela falta de experiência na escrita de artigos científicos, tendo demorado mais 1 semana que o previsto. Seguiu-se o levantamento e especificação dos requisitos que deverão estar presentes no sistema e a identificação de atributos de qualidade seguidos do desenho da arquitectura do sistema. Por último, construi os *mockups* da interface *web* do sistema. A escrita do relatório intermédio decorreu ao longo do semestre.

Olhando para a figura 9.1, consigo perceber que as minhas estimativas para as tarefas do primeiro semestre foram demasiado optimistas. A maioria das tarefas que estimei demoraram mais tempo a serem executados do que o estimado.

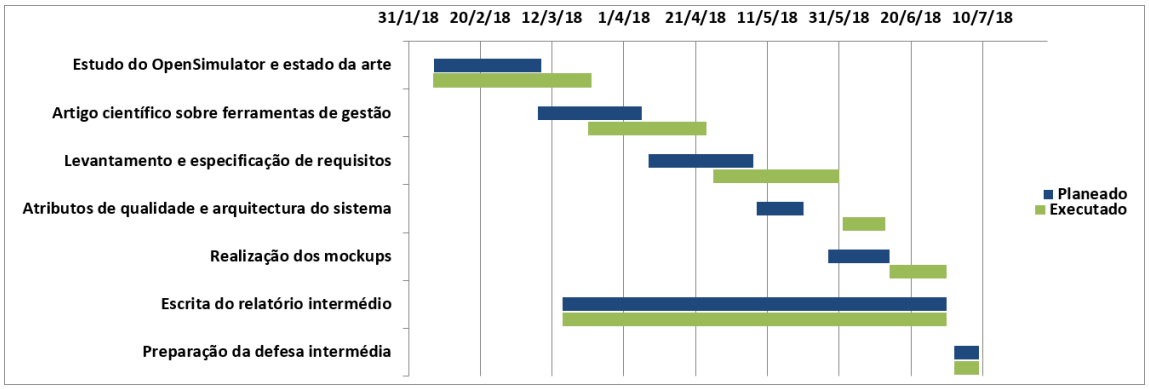

Figura 9.1: Planeamento das tarefas planeadas e executadas no primeiro semestre.

9.1.2 Segundo semestre

O segundo semestre passou por corrigir o relatório consoante o *feedback* recebido na defesa intermédia, corrigir o artigo escrito no primeiro semestre com o objectivo de o submeter a uma conferência, executar os planos de mitigação para os riscos encontrados (fazer tutoriais do Django e também desenvolver um pequeno protótipo que comunicasse com os SI da UAb), implementar o sistema, testá-lo e efectuar as correcções necessárias, escrever o manual de instalação do sistema e escrever o relatório final. Desta forma, a figura 9.2 representa o planeamento do trabalho para o segundo semestre.

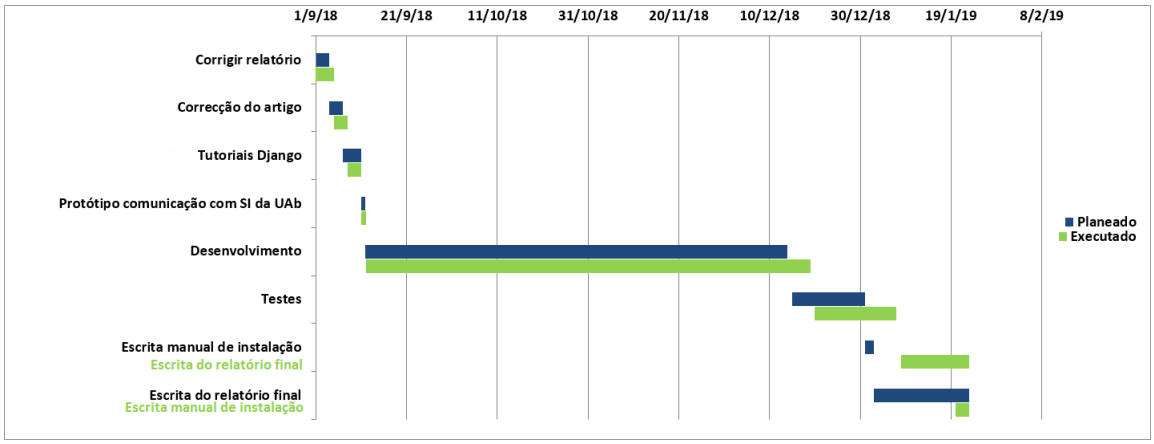

Figura 9.2: Planeamento das tarefas planeadas e executadas no segundo semestre.

Ao analisar o diagrama presente na figura 9.2 podemos verificar que algumas tarefas demoraram mais tempo do que o inicialmente planeado. Isto levou ao atraso das tarefas seguintes e ao pouco tempo que tive para redigir o relatório quando comparado com o planeado o que originou uma muito maior carga de trabalho na fase final do projecto. O manual de instalação não ficou concluido aquando da entrega do projecto devido à ocorrência deste risco.

Para além deste diagrama é possível analisar a relação entre o tempo estimado e o tempo que demorou a executar cada um dos requisitos funcionais no anexo D.

9.2 Riscos

De modo a avaliar os riscos presentes neste projecto, estes serão avaliados quanto a dois factores, o impacto e a probabilidade de ocorrerem. O impacto de um risco num determinado projecto refere-se ao grau de ameaça que um determinado acontecimento pode ter no projecto. A probabilidade de ocorrência representa o grau de possibilidade de um risco ocorrer. Para avaliar estes dois factores decidi dividi-los por 4 níveis, de modo a não cair na tentação de indicar o nível intermédio:

- Muito baixo: Em termos de impacto, este nível representa que os objectivos do projecto são fáceis de atingir mesmo que este se verifique. Em termos de probabilidade de ocorrência, isto indica que existe uma probabilidade de ocorrência entre 0 e 25%.
- Baixo: Em termos de impacto, este nível representa que os objectivos do projecto são atingíveis caso este se verifique, sendo para isso necessário algum esforço extra. Em termos de probabilidade de ocorrência, isto indica que existe uma probabilidade de ocorrência entre 26 e 50%.
- Alto: Em termos de impacto, este nível representa que os objectivos do projecto são difíceis de atingir caso este se verifique, estando em risco a conclusão do projecto. Em termos de probabilidade de ocorrência, isto indica que existe uma probabilidade de ocorrência entre 51 e 75%.
- Muito alto: Em termos de impacto, este nível representa que algum objectivo do projecto é inatingível caso este se verifique, não sendo possível a conclusão do projecto com todos os objectivos bem sucedidos. Em termos de probabilidade de ocorrência, isto indica que existe uma probabilidade de ocorrência superior a 75%.

Uma vez que riscos com probabilidades de ocorrência e impactos baixos ou muito baixos não são um obstáculo para a conclusão do projecto, decidi elaborar planos de mitigação para apenas riscos com uma probabilidade de ocorrência e imapacto entre baixo e muito alto e para riscos com um impacto muito alto na execução do projecto.

A tabela 9.1 apresenta os riscos identificados neste projecto. A maioria dos riscos identificados nesta tabela tem um impacto elevado aliado também a uma grande porbabilidade de ocorrência na execução do projecto, o que leva a que seja necessário um maior controlo na análise destes riscos ao longo do projecto. Desta forma, e seguindo os planos de mitigação, espera-se que a probabilidade de ocorrência dos mesmos baixe, aumentando assim a probabilidade de acabar com sucesso o projecto.

ID	Descrição	Impacto	Probabilidade	Plano de mitigação

Risco 1	Má estimação da duração das tarefas	Baixo	Alta	Ter um maior controlo sobre o plano de trabalhos, tendo reuniões semanais e controlando o tempo gasto em cada tarefa semanalmente.
Risco 2	Mudanças na tecnologia OpenSim	Alto	Muito baixa	
Riscos 3	Dificuldades ao integrar o produto nos sistemas de informação da UAb	Alto	Alta	Elaboração de um protótipo que comunique com os sistemas de informação necessários de forma a ganhar conhecimentos sobre estes.
Risco 4	Falta de conhecimento das tecnologias a utilizar	Alto	Baixa	Estudar profundamente as tecnologias a utilizar, realizando tutoriais e pequenos protótipos com os conhecimentos obtidos.

Tabela 9.1: Riscos identificados no projecto.

Ao longo do projecto apenas um dos riscos identificados ocorreu. A má estimação de algumas tarefas no segundo semestre do projecto teve como consequência a necessidade de trabalhar mais do que o previsto de 40 horas por semana. Isto também levou a que o manual de instalação não estivesse pronto no dia da entrega da dissertação. Apesar disto, o manual de instalação será terminado e entregue à Universidade Aberta, visto que este era uma restrição de negócio deste projecto.

Capítulo 10

Conclusões

Ao longo do primeiro semestre foi efectuada a especificação da solução, realizando o levantamento de requisitos, identificando os atributos de qualidade do sistema, esboçando a arquitectura e estudando a tecnologia OpenSimulator assim como o estado da arte referente ao uso desta. Estas tarefas cimentaram não só o meu conhecimento em relação ao OpenSimulator, como também fortificaram os meus conhecimentos de especificação de uma solução de *software*.

No segundo semestre foi dividido entre o desenvolvimento da solução e a fase de testes. Foram implementados e testados todos os requisitos classificados como *must have* sendo estes testados através de testes de robustez, de testes unitários e de testes de integração do *backoffice* com o OpenSimulator. Todos os testes efectuados tiveram sucesso validando assim o protótipo final.

Após a fase de testes foi possível ter uma reunião com os administradores de sistemas da Universidade Aberta de modo a validar o protótipo desenvolvido assim como perceber qual o contexto em que a solução poderia ser utilizada. Visto o OpenSimulator ser uma tecnologia na qual a maioria dos professores e alunos da Universidade Aberta não têm qualquer experiência o uso globalizado deste sistema para dar aulas no mundo virtual não seria algo realista pois seria necessário uma equipa de vários elementos para criar o espaço virtual como requerido pelo professor, para dar suporte aos estudantes e utilizadores externos e ainda gerir as regiões de forma a utilizar os recursos de forma eficiente. Esta tecnologia poderia ser bastante útil no entanto como ferramenta pedagógica de uso esporádico, sendo assim possível um elemento da equipa de suporte efectuar toda esta gestão. Foi também possível identificar algum trabalho a realizar no futuro como alterar a autenticação de forma a que um utilizador não tenha de efectuar um *login* através do *backoffice* para aceder ao mundo virtual e assim não guardar uma *hash* que representa a *password* do utilizador na base de dados do OpenSimulator. A integração do OpenSimulator com a plataforma de *elearning* também foi identificada como bastante importante para o futuro de forma a ser possível automatizar o processo de autorizar o acesso a uma região dos alunos de uma unidade curricular, como tinha sido já identificado.

Depois do *términos* do projecto concluo que o projecto foi bastante positivo. Por um lado cimentou o meu conhecimento acerca de Engenharia de *Software* desde a especificação da solução até à fase de testes. Por outro lado deu-me um conhecimento

sólido acerca do OpenSimulator e as suas vantagens de utilização como ferramenta pedagógica. Foi também possível contribuir com algumas funcionalidades para facilitar a adopção do OpenSimulator como ferramenta de ensino virtual como o controlo de presenças, a criação de eventos automaticamente no mundo virtual e o controlo do estado dos simuladores.

No futuro é importante realizar testes com utilizadores de forma a validar o *backoffice web* em ambiente pedagógico assim como integrar o *backoffice* com a plataforma de *elearning*. É necessário também testar a solução num ambiente mais próximo de um ambiente de produção, onde existam dezenas ou até centenas de utilizadores a utilizar o mundo virtual. Obviamente também será necessário implementar no futuro os requisitos funcionais levantados neste projecto que não foram possível implementar.

Referências

[1] Leonel Morgado, Hugo Paredes, Benjamim Fonseca, Paulo Martins, Ricardo Antunes, Lúcia Moreira, Fausto de Carvalho, Filipe Peixinho, and Arnaldo Santos. Requirements for the use of virtual worlds in corporate training: perspectives from the post-mortem of a corporate e-learning provider approach of second life and opensimulator. In *iLRN 2016: Immersive Learning Research Network Conference. Workshop, Short Paper and Poster Proceedings from the Second Immersive Learning Research Network Conference*, pages 18–29. Technischen Universität Graz, 2016.

[2] Veljko Potkonjak, Michael Gardner, Victor Callaghan, Pasi Mattila, Christian Guetl, Vladimir M Petrović, and Kosta Jovanović. Virtual laboratories for education in science, technology, and engineering: A review. *Computers & Education*, 95:309–327, 2016.

[3] Moodle website. `https://moodle.org/`. Accessed: 2018-05-22.

[4] Cas protocol. `https://apereo.github.io/cas/4.2.x/protocol/CAS-Protocol.html`. Accessed: 2018-06-05.

[5] How ldap protocol works. `https://www.hack2secure.com/blogs/how-ldap-protocol-works`. Accessed: 2018-05-22.

[6] Secure sockets layer (ssl) protocol overview. `https://www.ibm.com/support/knowledgecenter/en/SSYKE2_7.1.0/com.ibm.java.security.component.71.doc/security-component/jsse2Docs/ssloverview.html`. Accessed: 2018-05-22.

[7] Opensimulator. `http://opensimulator.org/wiki/Main_Page`. Accessed: 2018-03-02.

[8] Server commands. `http://opensimulator.org/wiki/Server_Commands`. Accessed: 2018-03-01.

[9] Moscow method. `https://www.projectmanager.com/training/prioritize-moscow-technique`. Accessed: 2018-03-10.

[10] Opensim web interfaces. `http://opensimulator.org/wiki/Webinterface`. Accessed: 2018-03-13.

[11] Bruno Vicente, Fernando Pais de Sousa, Leonel Morgado, Pedro Furtado, and João Pascoal Faria. A review of management tools for opensimulator. *Video-Jogos*, no prelo.

[12] Mwi. `http://www.hypergridbusiness.com/2013/03/myopengrid-releases-free-web-interface`. Accessed: 2018-03-13.

[13] jopensim. `https://www.jopensim.com/jopensim/the-component.html`. Accessed: 2018-03-14.

[14] Andreas Vilela, Márcio Cardoso, Daniel Martins, Arnaldo Santos, Lúcia Moreira, Hugo Paredes, Paulo Martins, and Leonel Morgado. Privacy challenges and methods for virtual classrooms in second life grid and opensimulator. In *Games and Virtual Worlds for Serious Applications (VS-GAMES), 2010 Second International Conference on*, pages 167–174. IEEE, 2010.

[15] Eric Arnold Anderson and Dave Patterson. *Researching system administration*. Citeseer, 2002.

[16] Alin Moldoveanu, Alexandru Gradinaru, Oana-Maria Ferche, and Livia Ştefan. The 3d upb mixed reality campus: Challenges of mixing the real and the virtual. In *System Theory, Control and Computing (ICSTCC), 2014 18th International Conference*, pages 538–543. IEEE, 2014.

[17] Colin Allison, Anne Campbell, Christopher John Davies, Lisa Dow, Sarah Kennedy, John Philip McCaffery, Alan Henry David Miller, Iain Angus Oliver, and Galhenage Indika Udaya Shantha Perera. Growing the use of virtual worlds in education: an opensim perspective. *Proceedings of the 2nd European Immersive Education Summit*, 2012.

[18] Andreas Konstantinidis, Thrasyvoulos Tsiatsos, Stavros Demetriadis, and Andreas Pomportsis. Collaborative learning in opensim by utilizing sloodle. In *Telecommunications (AICT), 2010 Sixth Advanced International Conference on*, pages 90–95. IEEE, 2010.

[19] Leonel Morgado, Hugo Paredes, Benjamim Fonseca, Paulo Martins, Álvaro Almeida, Andreas Vilela, Bruno Pires, Márcio Cardoso, Filipe Peixinho, and Arnaldo Santos. Integration scenarios of virtual worlds in learning management systems using the multis approach. *Personal and Ubiquitous Computing*, 21(6):965–975, 2017.

[20] Rob Barrett, Eser Kandogan, Paul P Maglio, Eben M Haber, Leila A Takayama, and Madhu Prabaker. Field studies of computer system administrators: analysis of system management tools and practices. In *Proceedings of the 2004 ACM conference on Computer supported cooperative work*, pages 388–395. ACM, 2004.

[21] Security in django. `https://docs.djangoproject.com/en/2.0/topics/security/`. Accessed: 2018-06-19.

Appendices

Esta página foi deixada em branco intencionalmente.

Anexo A

Entrevista semi-estruturada

9 de maio de 2018

1 Bases

Para preparar esta entrevista baseei-me no artigo [1] e na tese [2]. O primeiro descreve o dia-a-dia dos administradores de um sistema complexo. O segundo é um overview da administração de sistemas.

2 Objectivos

O maior objectivo é obter requisitos funcionais e não funcionais para o desenvolvimento da plataforma *web*. Com esta entrevista devo ficar a perceber que sistemas de informação estão presentes na Uab e como interagem entre si. Também devo perceber como os técnicos realizam a manutenção nestes sistemas de modo a perceber os métodos de manutenção com que estão mais confortáveis e como devo desenvolver o *BackOffice*.

3 Guião

1. Apresentação

   Espaço em que me irei apresentar aos técnicos da Uab.

2. Apresentação do projecto

   Irei apresentar aos entrevistandos aquilo que pretendemos realizar. Nesta parte será importante realçar que serão eles a gerir o sistema quando este estiver implementado o que fará com que tenham todo o interesse em ajudar o mais possível pois facilitará o seu trabalho futuro.

3. Sistemas de informação da Uab

   Espaço em que irei perguntar que sistemas de informação estão presentes na Uab e como se relacionam. Devo também perceber quais as principais dificuldades ao instalar e realizar manutenção aos sistemas de informação.

4. Tarefas diárias

Deverei falar com os técnicos sobre as suas tarefas diárias de administração de sistemas. Neste espaço deverei perceber quanto tempo existe diariamente para estes gerirem o OpenSim e perguntar o que os ajudaria a tornar a manutenção mais rápida e fácil. É também igualmente importante perguntar o quão frequentemente existem problemas.

5. Suporte técnico

   É importante aqui perceber como é que os técnicos estão habituados a receber pedidos de ajuda e suporte. É necessário perguntar se utilizam algum sistema de *tickets* na Uab pois caso exista um sistema único será mais fácil para os técnicos receberem os pedidos de todos os sistemas de informação, incluindo do OpenSim, no mesmo local.

6. Espaço aberto

   Este espaço deve servir para falar sobre sugestões que os técnicos possam ter ou aspectos que ache pertinentes depois de ouvir as respostas dos entrevistandos.

4 Possíveis questões

Nesta secção estão possíveis questões a realizar aos técnicos da Uab. De notar que servirão apenas como uma espécie de auxiliar de memória.

1. Que sistemas de informação estão presentes na Uab?
   (a) Poderá(ão) dar-me alguma imagem com o panorama geral dos SI?
2. Como realizam a manutenção destes sistemas de informação?
   (a) Qual a maior dificuldade que encontram ao gerir os SI?
   (b) É importante que as alterações que fazem num SI sejam reflectidas no sistema o mais rápido possível?
   (c) Desenvolveram alguma ferramenta para manter algum SI?
3. Qual a maior dificuldade que encontram ao instalar um novo SI?
   (a) Como ficam a conhecer o sistema? Através de documentação? Ou apenas experiência?
   (b) Caso o SI tenha documentação esta é-vos útil? Dito de outra forma, se uma ferramenta para manter um sistema tivesse uma boa documentação seria útil?
4. Como é o vosso dia-a-dia?
   (a) Executam tarefas de manutenção diariamente?
   (b) Qual as tarefas que ocupam mais tempo?

(c) O que ocupa uma maior percentagem de tempo em termos de manutenção dos SI? Perceber o que é necessário fazer ou fazê-lo?

(d) Quão frequentemente surgem problemas que é necessário resolver? Normalmente são business-critical ou não?

(e) Quando surge um problema business-critical trabalham apenas na resolução desse problema?

(f) Têm alguma sugestão em relação ao sistema de gestão do OpenSim que faça com que seja mais fácil e rápida a sua manutenção?

5. Como tratam das questões de suporte?

   (a) Têm algum sistema único que sirva para todos os SI?

   (b) Respondem manualmente aos "queixosos"?

6. Espaço aberto

   (a) Têm algum sistema que utilize as credenciais e comunique com o sistema de autenticação?

   (b) etc...

References

[1] Barrett, R., Kandogan, E., Maglio, P. P., Haber, E. M., Takayama, L. A., & Prabaker, M. (2004, November). Field studies of computer system administrators: analysis of system management tools and practices. In Proceedings of the 2004 ACM conference on Computer supported cooperative work (pp. 388-395). ACM.

[2] Anderson, E. A., & Patterson, D. (2002). Researching system administration. University of California, Berkeley.

Esta página foi deixada em branco intencionalmente.

Anexo B

ID	Descrição	Prioridade
RF-1	Registo de utilizadores	*Must have*
RF-2	Editar utilizadores	*Must have*
RF-3	Recuperar dados de acesso	*Must have*
RF-4	Definir avatar pré-definido	*Must have*
RF-5	Gerir grupos de utilizadores	*Should have*
RF-6	Estado das regiões	*Must have*
RF-7	Ver mapa do mundo virtual	*Must have*
RF-8	Ver distribuição de utilizadores no mundo virtual	*Should have*
RF-9	Criar regiões	*Must have*
RF-10	Editar regiões	*Must have*
RF-11	Reiniciar regiões	*Must have*
RF-12	Guardar regiões	*Must have*
RF-13	Listagem de utilizadores com acesso a zonas privadas	*Must have*
RF-14	Autorizar e revogar o acesso de utilizadores a uma região privada	*Must have*
RF-15	Autorizar estudantes de uma unidade curricular a aceder a uma região privada	*Could have*
RF-16	Gestão de NPCs	*Should have*
RF-17	Gestão de logs	*Must have*
RF-18	Definir localização base para os utilizadores	*Must have*
RF-19	Enviar mensagens para todas as regiões	*Must have*
RF-20	Criar eventos	*Must have*
RF-21	Editar eventos	*Must have*
RF-22	Eliminar eventos	*Must have*
RF-23	Teleportar utilizadores	*Must have*
RF-24	Criar e eliminar sessões de treino automaticamente	*Should have*
RF-25	Autenticação realizada através dos dados de acesso do LMS	*Must have*
RF-26	Estudantes apenas podem mudar a palavra chave através do LMS	*Must have*
RF-27	Associar uma sessão de treino a um espaço já existente	*Must have*
RF-28	Controlar as presenças nas sessões	*Must have*
RF-29	Poder dividir a comunicação em subgrupos de participantes	*Should have*
RF-30	Fornecer aos participantes avatars pré-preparados	*Should have*
RF-31	Poder restaurar a aparência dos avatars	*Must have*
RF-32	Sistema de suporte	*Must have*
RF-33	Listagem de utilizadores	*Must have*
RF-34	Listagem de regiões	*Must have*
RF-35	Listagem de eventos	*Must have*
RF-36	Banir/Voltar a autorizar utilizadores	*Must have*
RF-37	Desligar regiões	*Must have*
RF-38	Guardar o estado do sistema (regiões)	*Must have*

Tabela 1: Requisitos funcionais do sistema a desenvolver.

Esta página foi deixada em branco intencionalmente.

Anexo C

A review of Management Tools for OpenSimulator

Bruno Vicente[1], Fernando Pais de Sousa[2], Leonel Morgado[3], Pedro Furtado[4] and João Pascoal Faria[5]

[1] Dept. of Informatics Engineering, University of Coimbra, Coimbra, Portugal
brpv@student.dei.uc.pt
[2] Universidade Aberta, Lisbon, Portugal & Centro de Medicina de Reabilitação da Região Centro – Rovisco Pais, Tocha, Portugal
fpsousa@roviscopais.min-saude.pt
[3] Universidade Aberta, Coimbra, Portugal & INESC TEC, CIAC, & LE@D
leonel.morgado@uab.pt
[4] Dept. of Informatics Engineering, University of Coimbra, Coimbra, Portugal
pnf@dei.uc.pt
[5]INESC TEC & Faculty of Engineering, University of Porto, Porto, Portugal
jpf@fe.up.pt

Abstract. To host OpenSimulator virtual world servers at educational institutions, system administrators find at their disposal a diversity of web-based management systems with different sets of features. To support the selection among current management tools and provide a baseline from which to identify subsequent development needs, we installed and evaluated 4 of these systems (*WiFi pages, OSMW, MWI* and *jOpenSim*), analysing and comparing their features. WiFi pages only provides account-management features. MWI has mostly the same features, but also provides systems administrators with the option of creating their own management website. OSMW has account-management and maintenance features, such as log management and editing of configuration files. jOpenSim provides features for account and event management and feature for generating some actions within virtual world, such as broadcasting a message to all regions. From matching the identified features with the literature-reported requirements for virtual world deployment at educational organizations, we conclude that there is no management tool that fulfils all the functional requirements reported in the literature and, therefore, that the adoption of current tools by system administrators will always requires manually performing some of the administrative tasks. We therefore call for development of novel, more encompassing administrative tools for OpenSimulator virtual worlds.

Keywords: OpenSimulator, OpenSim, virtual worlds, system management, system administration.

1 Introduction

There are now many examples of virtual world use in support of learning and training. Their prototyping capabilities enable the rapid, inexpensive, creation and testing of three-dimensional objects, choreographies, and interaction scenarios for virtual actors. These enable concrete (i.e., non-abstract) renderings of theories, concepts, and ideas among students and teachers (Morgado et al., 2010). Also, the use of virtual reality can give the student the opportunity to experiment, through simulation, situations that if lived in the real world could be dangerous, expensive, or otherwise inaccessible. This can also be done in cooperation between simultaneous participants in the virtual world, geographically separated in the physical world (*ibid*.).

Various alternative technologies exist for using virtual worlds in education. Besides third-party hosting such as the well-known Second Life or its purported successor Sansar, there are also several alternatives for hosting at the institutions themselves, which enable institutions to use only local resources, avoid virtual hosting fees, and control network access to the virtual world (Vilela et al.,

2010). Examples include somewhat recent platforms such as High Fidelity and Minecraft, and older solutions such as OpenCobalt and Project Wonderland. However, OpenSimulator (vg. OpenSim) remains the most common virtual world hosting platform in education, or at least the most commonly reported in practitioner and research presentations in conferences focusing on this field (e.g., VWBPE – Virtual Worlds Best Practices in Education, which announced its 12th edition for 2019 - https://vwbpe.org/).

However, being the most referenced does not mean that OpenSim (or any other virtual world) is in widespread use. This has been hindered by a variety of issues (Morgado, 2013), such as lack of integration with existing learning management systems and other organizational information systems, lack of support and knowledge for network profiling, content management at the organizational level (Morgado et al., 2016, December), or user authentication federation (Cruz et al., 2015). The overburdened list of administrative and technical tasks that the systems administrators must deal with, can hamper the OpenSim installations, making it very time consuming to set up and manage activities in these environments (Morgado et al., 2016).

For systems administrators of typical educational organizations, an OpenSim server will be only one among a diversity of computing services that must be managed and serviced. The application structure of organizations, whether educational or otherwise, involves a huge diversity of services, which must be managed and maintained by the systems administrators (Barret et al., 2004). Having to control and manage an OpenSim installation may be seen as yet another unknown server software to be mastered by the people responsible for it. And if usage is not widespread within the institution, it may be a service whose management falls outside daily routines, making it time consuming for the system administrator to review procedures and refresh knowledge about the databases schemas and client-server communication protocols and prone to errors, hence the importance of a system that systematizes and ensures the quality of its management, impacting on the degree of reliability perceived by users (in the case of an educational organization, teachers and students).

In this paper, we present the outcome of studying the features of four OpenSim management systems, to evaluate their applicability. Section 2 provides some background on OpenSim. Section 3 describes the management tools we installed as well as an overview of their features. Section 4 compares the different tools regarding their functionalities. Section 5 is where we draw conclusions regarding the results obtained.

2 OpenSimulator

The OpenSimulator project (also known as OpenSim) is an extensible virtual worlds platform, built to simulate virtual multi-user three-dimensional spaces, where users can create objects (and modify or delete them) and, through scripts, program interactions and behaviours, considering variables representing physical characteristics of materials, as well as light sources defined in these virtual spaces (Fishwick, 2009). The configuration is managed through the text file "opensim.ini", which contains the environment parameters: users starting location coordinates, address for access and authentication, etc. (ibid.). The system is composed by virtual regions simulators and data services, which include user management, inventory storage, and others. A virtual region is a part of the virtual space where users and virtual objects can be located. The starting location is where users are placed by default upon logging in. An inventory is a personal virtual storage of each user where virtual items can be stored. For systems administrators, there is a diversity of such aspects that must be considered. For instance, OpenSim virtual worlds can be configured in Standalone mode, or in Grid mode. If the system is running in *standalone* mode, it executes the simulation of region and data services in a single computational process (opensim.exe) (**Fig. *1***). Thus, system administrators must consider modes and their available physical and virtual machines, when pondering OpenSim installations and configuration.

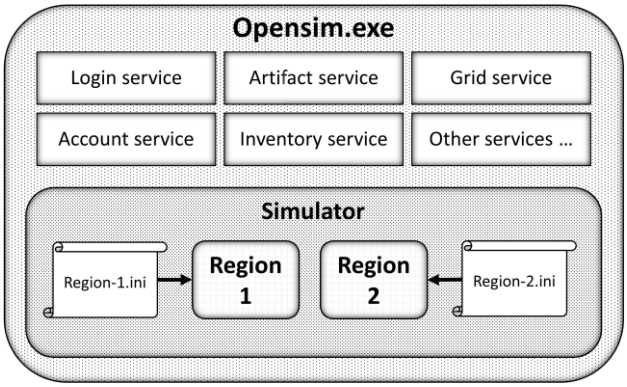

Fig. 1. Standalone configuration with two regions.

Grid mode works differently. Data services and region simulators are separated into two or more computational processes. In this mode, data services are run by the "*Robust.exe*" process (*Redesigned OpenSimulator Basic Universal Server Technology - ROBUST*[1]). Region simulators continue to be run by the "OpenSim.exe" process, but now this process works only for the environment simulation service communicating with data services, running on separate instances (**Fig. *2***).

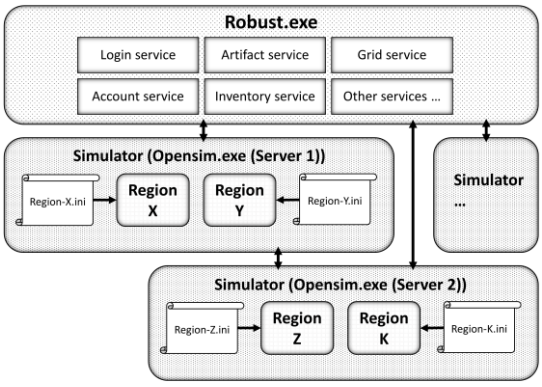

Fig. 2. Grid mode configuration with four regions on two processes.

The reason for having separate processes for regions is that this allows these services to run region simulators (established by users) that connect to other regions, outside the organization, such as public Grids, while keeping data processes isolated from external servers. There is a specific protocol for this, called Hypergrid, with its own settings and management aspects to consider (Lopes, 2011).

OpenSim provides a way to manage the installation via the command line, without providing a graphical interface. The administration commands of the OpenSim servers are entered by typing in the terminal executing the process. These are of two types, those that apply to the simulator and those that apply to grid services. In a standalone installation, both run on the same console because the server is unique. In a grid architecture installation, the commands for the services work on the ROBUST server console and the simulator commands on the consoles of the various simulators (separately). The list of these commands can vary with the version of OpenSimulator used and the best way to find out which ones are available is with the "help" command in the region console.

[1] http://opensimulator.org/wiki/ROBUST/

3 Available management tools

As stated in section 1, managing an OpenSim installation can be time-consuming. To ease this, some management tools have been developed by various parties. To identify existing management tools for this evaluation, we searched them in the following manner:

1. Querying the GitHub website with the search terms "OpenSimulator", "OpenSim", "OpenSimulator web" and "OpenSim web;
2. Querying the Research Gate website with the search terms "OpenSim", "OpenSimulator" and "managing OpenSim";
3. Browsing the OpenSimulator.org website for info on web interfaces for management.

By the first method we found 2 management systems: OSMW and MWI. By the second method we did not encounter any work related to developing an administration tool for OpenSim. By the last method we encountered a web page with a list of management tools for OpenSim[2], but only a few could be installed due to deprecation issues: most existing solutions were based on technologies that are either obsolete, deprecated, or the tool was just developed on a much-outdated version of those technologies, sometimes as early as 9 years ago. We attempted to contact the creators of said tools but failed to get support. Given the prevalence of the use of OpenSim, we were a bit surprised by this obsolescence of management tools, but this actually is consistent with the argument that widespread adoption is hindered by lack of adequate management tools (Morgado, 2013). The tools we were able to install and evaluate from this third source were: jOpenSim, Remote Admin, and OpenSimulator WiFi pages. Alongside the ones found at GitHub, these form our analysis batch.

The goal of the evaluation of the management tools was to be able to identify which features were present in each one and how would they contribute to minimize the effort of a systems administrator conducting daily tasks. The identification of the features was done by reading the tools descriptions and documentation (when available, such as for jOpenSim) and by installing and analysing the web interface of each one after we configured it to work with our OpenSim installation.

Next, we present the tools along with a brief description of each. The following section makes a comparison of the features available in these tools.

3.1 Remote Admin[3]

To access OpenSim remotely, the installation package itself provides the Remote Admin endpoint for the XML-RPC protocol.

Other than providing system administrators with remote access to the OpenSim services specifically (without having to open a remote terminal to the full machine) Remote Admin provides no further support, since its operation is identical to local administration tasks.

3.2 Open Simulator Manager Web[4]

Open Simulator Manager Web, or simply OSMW, is a management tool written in PHP that supplies the administrators of OpenSim installations with a web interface that enables user management, export of some data lists, monitoring and backup of regions, and some management of configuration files (table **1**). This tool was developed 3 years ago and so some of the code needs to be reconstructed due to the changes to PHP consecutively made from version 5.5.29 to version 7.1.14 that deprecated some functions and changed some general semantics on the PHP language.

---

[2] http://opensimulator.org/wiki/Webinterface
[3] http://opensimulator.org/wiki/RemoteAdmin
[4] https://github.com/Nino85Whitman/OpenSim-Manager-WebV5

3.3 MWI[5]

MWI is an open-source web front-end for OpenSim grids, also intending to be a full content management system (CMS).

It has two main parts: a front end for visitors and an administration area for grid managers, where they can change most of the system characteristics. It provides systems administrators with account management, user groups management, and more (table **1**).

It is also possible to use it to automate some tasks, such as user registration and password recovery, and to easily access information about the bound regions on the grid.

3.4 Open Simulator WiFi pages[6]

Wifi pages is a system constituted by a set of components that can be used in both standalone simulators and grid installations to manage OpenSim accounts, updates, passwords and basic aspects of users' inventories. It is said that its properties make it a good fit for small-to-medium OpenSimulator-based virtual worlds. It does not require the installation of a web server since it uses the built-in Open Simulator features.

3.5 jOpenSim Web Interface[7]

jOpenSim is an extension for Joomla!, a free and open-source CMS[8], allowing interaction with an OpenSimulator server and 2 modules (grid status and friends online). This component can interact with grid or standalone mode installations with MySQL as its database engine. Since it is an extension of Joomla!, the rationale is that the administrator can setup a website that suits his needs and design appeal and taste using the features made available by the components and modules provided by jOpenSim. It provides user management, user groups management, event management, etc. (table **1**).

4 Management tools comparison

As shown in table **1**, we can see that three types of features emerge: the ones we consider basic (features that are available on almost every studied tool), the ones that are specific to a certain topic, such as region-management, account management, or configuration and administration related features, and, lastly, features that are only available on a specific solution.

Some tools have a far more extensive range of features than others. Comparing OpenSim WiFi pages with other administrative tools we can see that while it is the only that enables inventory import, beyond that it's the one with the least number of features.

WiFi pages is a very limited management tool since it only provides features related to account management.

Remote Admin, as mentioned earlier, simple enables administrators to manage an OpenSim installation without having to provide full remote access to a machine (i.e., SSH, Remote Desktop). But it remains the only way to access a set of region managements features remotely, that no other tools provide, thus exposing the lack of feature coverage of other tools.

OSMW, along with jOpenSim, is one of the tools with more features. The first has the basic set of features present on every tool and focuses on the maintenance of the installation. It has some unique

---

[5] http://mwi.myopengrid.com/page/home
[6] http://opensimulator.org/wiki/Wif
[7] https://www.jopensim.com/jopensim/thecomponent.html
[8] https://www.joomla.org/

features like the ability to manage non-player characters (agents that are controlled by the computer, usually called bots by OpenSim users), save terrain, manage the system's logs and broadcast a message to all regions. jOpenSim is the only management tool that enables the creation of regions, although the other region-related functionalities are not supported. It was created to manage a grid, i.e. where each region may have its own administrator. It also provides the basic functionalities as well as some unique features like listing virtual money transactions, event management and an in-world search that allows managers to find items within the virtual world.

MWI is one of the two analysed tools that can manage user groups. This functionality may not be important for small OpenSim instalations, but for more complex OpenSim grids it is very important, since groups can be formed by any set of 2 or more avatars. Its other features are like the ones provided by WiFi pages and so we can say that this feature is its only advantage over the latter.

WiFi pages can be used to manage a small OpenSim installation, as it does not need an HTTP server to work and complements the administrator commands with a web interface for account management.

The only tool that provides the ability to authorize, or not, users to access certain parts of the virtual world through a graphical interface is MWI. This feature may be important if there are private sessions occurring somewhere in the virtual space.

With a more thorough analysis of table **1**, we can see that the most adequate management tool for an administrator depends on the overall computational and usage environment of the OpenSim installation. For example, if one uses OpenSim only to explore a virtual form of socialization, with a few people connecting to a server, maybe the only functionalities needed are being able to create, edit, and delete users from the virtual world. If so, WiFi pages are enough to manage this system. On the other hand, if a training corporation or educational institution wants to use an OpenSim installation for training and education purposes, it is likely that none of the management tools explored in this article provides enough features, requiring some of the administrative work to be done at the console.

Table 1. Features available on the management tools.

	Remote Admin	Manager Web	MWI	WiFi Pages	jOpenSim
Remote access to OpenSim	yes	yes	yes	yes	yes
Create users	command line	yes	yes	yes	yes
Edit users	command line	yes	yes	yes	yes
Define default avatar				yes	yes
Export inventory		yes			
Import inventory				yes	
Manage user groups			yes		yes
Region Status	command line	yes	yes	yes	yes
View Map		yes			yes
Create region	command line				yes
Edit regions	command line	yes			
Restart regions	command line				
Get region access list	command line				
Add users to a region's access list	command line				

Remove users from a region's access list	command line				
Save region	command line	yes			
Load region	command line				
Non-player character management		yes			
Save land		yes			
File management		yes			
Log management		yes			
Edit configuration files		yes			
Edit simulators table		yes			
Define home location			yes		yes
Broadcast message to all regions		yes			yes
In-world object search					yes
Define default region			yes	yes	yes
Event management					yes
List money transactions					yes
Terminal management					yes
Get OpenSim version	command line				yes
Teleport agents	command line				

5 Conclusions

As seen in section 4, none of the analysed tools provides a superset of all services. There are always missing features in a tool that are available in others, and features that remain available only via the command line. If one considers the list of requirements identified by Morgado et al. (2016), many aren't available at all. So, it is safe to say that there is not a management tool that fulfils the needs for practical organizational management of OpenSim installations. The analysed tools were likely built with an ad hoc approach, focusing on individual managers' specific needs, and do not reflect an organizational requirements perspective of the management challenges of an OpenSim installation.

We encourage further research on identification and specification of administrative features required of OpenSim management tools, and their subsequent development, towards a more widespread deployment of virtual worlds in education.

Acknowledgements

This research has been partially financed by the European Commission [BEACONING H2020ICT-2015-687676].

References

Barrett, R., Kandogan, E., Maglio, P. P., Haber, E. M., Takayama, L. A., & Prabaker, M. (2004, November). Field studies of computer system administrators: analysis of system management tools

and practices. In Proceedings of the 2004 ACM conference on Computer supported cooperative work (pp. 388-395). ACM.

Cruz, G., Costa, A., Martins, P., Gonçalves, R., & Barroso, J. (2015). Toward educational virtual worlds: should identity federation be a concern?. Educational Technology & Society, 18(1), 27-36.

Fishwick, P. A. (2009, December). An introduction to OpenSimulator and virtual environment agent-based M&S applications. In Winter simulation conference (pp. 177-183). Winter Simulation Conference.

Lopes, C. (2011). Hypergrid: Architecture and protocol for virtual world interoperability. IEEE Internet Computing, 15(5), 22-29.

Morgado, L., Varajão, J., Coelho, D., Rodrigues, C., Sancin, C., & Castello, V. (2010). The attributes and advantages of virtual worlds for real world training. The Journal of Virtual Worlds and Education, 1 (1), 15-35.

Morgado, L. (2013). Technology challenges of virtual worlds in education and training-research directions. In Games and Virtual Worlds for Serious Applications (VS-GAMES), 2013 5th International Conference on (pp. 1-5). IEEE.

Morgado, L., Paredes, H., Fonseca, B., Martins, P., Almeida, Á., Vilela, A., ... & Santos, A. (2016, December). Integrating virtual worlds with learning management systems: the MULTIS approach. In Ubiquitous Computing and Communications and 2016 International Symposium on Cyberspace and Security (IUCC-CSS), International Conference on (pp. 167-172). IEEE.

Morgado, L., Paredes, H., Fonseca, B., Martins, P., Antunes, R., Moreira, L., ... & Santos, A. (2016). Requirements for the use of virtual worlds in corporate training: perspectives from the post-mortem of a corporate e-learning provider approach of Second Life and OpenSimulator. In iLRN 2016: Immersive Learning Research Network Conference. Workshop, Short Paper and Poster Proceedings from the Second Immersive Learning Research Network Conference (pp. 18-29). Technischen Universit¨at Graz.

Vilela, A., Cardoso, M., Martins, D., Santos, A., Moreira, L., Paredes, H., ... & Morgado, L. (2010, March). Privacy challenges and methods for virtual classrooms in Second Life Grid and OpenSimulator. In Games and Virtual Worlds for Serious Applications (VS-GAMES), 2010 Second International Conference on (pp. 167-174). IEEE.

Esta página foi deixada em branco intencionalmente.

Anexo D

Requisitos funcionais	Planeado	Executado
RF-9 - Criar região	7	10
RF-1 - Registo de utilizadores	1	1
RF-25 - Autenticação Uab	13	13
RF-33 - Listagem de utilizadores	1	3
RF-2 - Editar utilizadores	2	2
RF-3 - Recuperar dados de acesso	2	1
RF-4 - Definir Avatar pré-definido	5	4
RF-6 - Estado de regiões	0,5	0,5
RF-7 - Mapa de regiões	5	3
RF-10 - Editar regiões	1	1
RF-11 - Reiniciar regiões	1	1
RF-12 - Guardar regiões	1	3
RF-34 - Listagem de regiões	1	1
RF-37 - Desligar regiões	1	1
RF-38 - Guardar estado do sistema	1	1
RF-13 - listagem utilizadores privadas	1	1
RF-14 - autorizar e revogar acesso regiões	2	2
RF-18 - definir loc. Base utilizadores	1	1
RF-23 - teleportar utilizadores	2	2
RF-26 - mudar pass apenas LMS	0,25	0,25
RF-36 - Banir/Voltar a autorizar utilizadores	1	1
RF-23 - Criar eventos	2	2
RF-24 - Editar eventos	2	2
RF-25 - Eliminar Eventos	0,25	0,25
RF-35 - Listagem de eventos	2	2
RF-27 - Criar e eliminar eventos automaticamente	5	5
RF-30 - Associar evento espaco existente	0,25	0,25
RF-31 - Controlar presenças eventos	4	6
RF-20 - Gestão de logs	2	2
RF-22 - Enviar mensagens para todas as regiões	1	1
RF-35 - Poder restaurar a aparência dos avatars	0,25	0,25
RF-37 - Sistema de suporte	1	1

Tabela 2: Relação entre o tempo planeado e o tempo de execução de cada requisito duncional em dias.