IPG Politécnico da Guarda
Polytechnic of Guarda

RELATÓRIO DE PROJETO

Licenciatura em Engenharia Informática

Rui Miguel Andrês Paredes

setembro | 2018

Escola Superior de Tecnologia e Gestão

Instituto Politécnico da Guarda

PROJETO - LABSECURITY

RUI PAREDES

RELATÓRIO PARA A OBTENÇÃO DO GRAU DE LICENCIADO

EM ENGENHARIA INFORMÁTICA

Set/2018

Elementos Identificativos

Nome: Rui Miguel Andrês Paredes

Número: 1011670

Curso: Licenciatura em Engenharia Informática

Ano Letivo: 2017/2018

Escola: Escola Superior de Tecnologia e Gestão – Instituto Politécnico da Guarda

Orientador: Professor José Carlos Coelho Martins Fonseca

Projeto realizado entre: maio e setembro de 2018

Agradecimentos

Queria começar por agradecer ao meu orientador, Professor José Fonseca, por toda a disponibilidade que mostrou ao longo do desenvolvimento do projeto, por todas as críticas e sugestões que ajudaram e tiveram um papel bastante importante no desenvolvimento deste projeto. Muito Obrigado.

Agradeço ainda aos meus amigos e colegas de turma que me acompanharam ao longo destes três anos de licenciatura, que me ajudaram e apoiaram em todos os momentos ao longo desta minha etapa.

Por fim, agradeço à minha família, pois sem eles nada disto seria possível. Por todo o apoio e ânimo prestado ao longo desta fase difícil e de muita tensão, um Muito Obrigado.

Resumo

O presente documento descreve o projeto realizado no âmbito da unidade curricular de Projeto de Informática, integrada na Licenciatura em Engenharia Informática da Escola Superior de Tecnologia e Gestão do Instituto Politécnico da Guarda.

O projeto desenvolvido consiste numa plataforma *web* focada no tema da *cybersecurity*, onde o principal objetivo da mesma é o de preparar melhor jovens programadores aos vários riscos aos quais os seus projetos poderão um dia vir a estar vulneráveis. De forma a alcançar este objetivo, a plataforma *web* foi focada num ambiente competitivo, constituído por desafios de *cybersecutiy*, onde os vários utilizadores poderão participar em competições de *hacking* entre eles e por à prova as suas capacidades e conhecimentos ao nível de *cybersecurity*.

Para o desenvolvimento do projeto foi usada a metodologia de desenvolvimento ágil, SCRUM. A aplicação web foi desenvolvida usando as *frameworks* ReactJS e NodeJS com recuso à base de dados Mysql.

Apesar do projeto não ter alcançado todos os objetivos previstos, os dois objetivos no qual o projeto se baseava foram alcançados com sucesso, sendo estes a secção de competições entre os vários utilizadores e a funcionalidade de sugestões de desafios pelos utilizadores.

Palavras-chave: Aplicação *Web, Cybersecurity, Web Services, ReactJS, NodeJS, Plataforma Competitiva*

Abstract

This report describes the project carried out within the scope of the final project of the Computer Science Engineering degree at the School of Higher Education in Technology and Management of the Polytechnic of Guarda.

The project developed consists of a web platform focused on the cybersecurity theme, where the main objective is to prepare young programmers to the variety of risks that their future projects may face. To accomplish this objective, the web platform was focused on a competitive environment, composed of cybersecurity challenges, where users can participate in hacking competitions against other users and put to test their skills and knowledge about cybersecurity

This project was developed using an agile methodology, SCRUM. The web platform was developed using frameworks as ReactJS and NodeJS combined with MySQL database.

Although some of the foresee objectives to the project weren't accomplished, the two primary objectives were accomplished successfully: cybersecurity competitions between users and challenges suggestions provided by the users.

Keywords: Web Aplication, Cybersecurity, Web Services, ReactJS, NodeJS, Competitive Platform

Índice

Índice de Figuras

Índice de Tabelas

Lista de Acrónimos

API	Application Programming Interface
CSRF	Cross-Site Request Forgery
CSS	Cascading Style Sheets
CTF	Capture The Flag
DOM	Document Object Model
ER	Entity- Relationship
FS	File System
HTML	HyperText Markup Language
JSON	JavaScript Object Notation
LAMP	Linux, Apache, Mysql, PHP
NPM	Node Package Manager
PHP	HyperText Preprocessor
RUP	Rational Unified Process
SQL	Structured Query Language
SQLi	Structured Query Language Injection
SSH	Secure Shell
UML	Unified Modeling Language
XML	Extensible Markup Language

1 Introdução

Este documento descreve o projeto realizado pelo aluno Rui Paredes no âmbito da unidade curricular Projeto de Informática, da Licenciatura de Engenharia Informática.
O projeto desenvolvido consiste numa plataforma *web* de programação e segurança, onde podem ser obtidas informações sobre vários ciberataques, realizar *quizzes* sobre as várias vulnerabilidade, os ataques às mesmas e forma de defender, e competir contra outros utilizadores em competições de *hacking*. Dadas as semelhanças de objetivos, a aplicação CodeFlex – Aplicação *Web* de Programação Competitiva [1], serviu de inspiração no desenvolvimento deste projeto.

1.1 Motivação

Todos os anos, alunos da UC de Programação e Segurança da Licenciatura de Engenharia Informática do Instituto Politécnico da Guarda desenvolvem exemplos que ilustram vulnerabilidades, ataques e defesas na área da cibersegurança. Porém, após a apresentação dos mesmos para avaliação, estes exemplos tipicamente não voltam a ser usados para o estudo dessas vulnerabilidades.

Desta situação surgiu a ideia de desenvolver um laboratório de programação e segurança, que consiste numa plataforma web, onde se poderá aprender mais acerca de vulnerabilidades, ataques e defesas ao nível da segurança informática, contando com vários exemplos desenvolvidos por alunos da UC de Programação e Segurança, servindo ainda desta forma como um repositório para utilização futura.

Este laboratório de programação e segurança para além de disponibilizar informações acerca de várias vulnerabilidades, ciberataques existentes e como se defender dos mesmos, permite ainda realizar *quizzes* e competições de *hacking* contra outros utilizadores, servindo assim como uma plataforma de aprendizagem e de desenvolvimento das capacidades dos seus utilizadores.

1.2 Objetivos da Plataforma

Esta plataforma tem como principais objetivos:

- Realizar desafios de cibersegurança
- Gerar e gerir competições de cibersegurança
- Verificação automática das respostas a *quizzes* ou desafios.
- Sistema de Classificação por pontos e tempo
- Realizar *Quizzes*
- Ver lições teóricas sobre vulnerabilidades/ataques
- Inserção de Propostas de desafios e questões para os *quizzes*

1.3 Estrutura do Documento

Este documento encontra-se dividido em 7 capítulos. Neste primeiro capítulo é feita uma introdução ao projeto, descrevendo os motivos que levaram ao desenvolvimento do mesmo e os objetivos pretendidos. No segundo capítulo é feito um estudo e análise a várias soluções que se enquadram nesta área com o objetivo de encontrar pontos que possam servir de apoio ao desenvolvimento da aplicação *web*. No terceiro capítulo é descrita a metodologia de trabalho usada neste projeto. No quarto capítulo são apresentados os elementos usados no planeamento e *design* da aplicação. No quinto capítulo é feita uma descrição do processo de implementação da solução, desde as tecnologias usadas no desenvolvimento da aplicação, até à arquitetura do sistema e algumas das funcionalidades implementadas. No sexto capítulo é feita uma descrição de como foram realizados os testes à solução desenvolvida. Por fim, no sétimo capítulo são apresentadas as conclusões e sugestões de trabalho futuro.

2 Estado da Arte

Definidos os objetivos propostos para o desenvolvimento do projeto, foi realizada uma pesquisa de forma a saber quais as soluções atualmente existentes.

Desta pesquisa foram encontradas várias plataformas em que alguns dos objetivos são idênticos aos definidos inicialmente para este projeto, e de onde foram retiradas informações acerca do seu funcionamento que poderão servir de apoio para o desenvolvimento da plataforma *web* do LabSecurity.

Estas plataformas web, embora com alguns objetivos idênticos aos propostos, não chegam a completá-los todos ou divergem em alguns pontos importantes dos mesmos. Algumas delas focam -se em jogos CTF (*Capture the flag*) entre equipas ou indivíduos, ou apenas no ataque a cada vulnerabilidade sem competição direta com outros utilizadores.

2.1 Plataformas existentes

Das várias aplicações encontradas na fase de pesquisa foram selecionadas para análise duas que demonstraram ter um maior reconhecimento na comunidade, sendo estas a Hacking Lab e a CTF365 e ainda outras três também bastante populares tendo sido mencionadas pelo orientador, sendo estas o Reversing.kr, RingZer0Team e WebGoat.

2.1.1 Hacking Lab

O Hacking-Lab [2] consiste numa plataforma de *hacking* ético onde podem ser realizados desafios de segurança e que contem várias competições que vão desde criptografia, a engenharia-reversa, entre outras, e que variam ainda dentro de cada categoria em dificuldade. Pode-se observar na Figura 1**Erro! A origem da referência não foi encontrada.** como estes desafios se encontram apresentados.

O Hacking-Lab apresenta uma variedade de desafios, aproximadamente 23 gratuitos sendo que ainda possui desafios pagos aos quais só se tem acesso após feito o pagamento, pelo qual não se sabe o número exato. Alguns destes desafios requerem que o utilizador use software adicional para resolver determinados desafios.

O Hacking-Lab dispõe ainda de uma secção de divulgação de conferências de cibersegurança. De forma a ser possível aos utilizadores resolver os desafios é necessário que estes instalem uma máquina virtual com um sistema fornecido pela plataforma e se conectem a ela através de VPN.

Após resolvido e submetido um desafio, a solução apresentada pelo utilizador é enviada para ser analisada por uma equipa, e caso esta se encontre correta, o mesmo obtém pontos, sendo estes adicionados à classificação do utilizador.

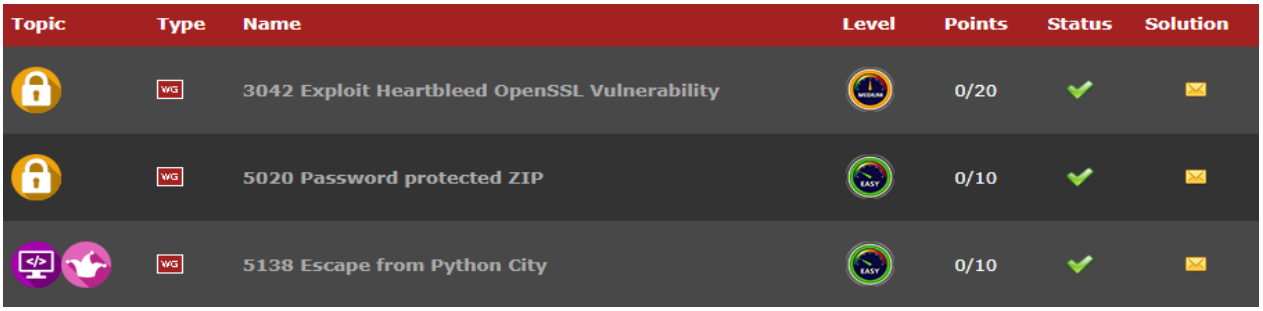


Figura 1- Desafios Hacking- Lab

2.1.2 CTF365

O CTF365 [3] consiste numa plataforma com jogos CTF (*Capture the flag*), em que são simulados ataques a vulnerabilidades aos serviços que cada equipa possui, e em que o objetivo principal consiste nas equipas tornarem os seus serviços o mais invulneráveis que conseguem enquanto simultaneamente atacam os serviços de equipas inimigas. Os serviços de cada equipa encontram -se nos vários "*Fortress*" que possuem, sendo que um *fortress* corresponde a uma máquina virtual atribuída à equipa.

Na realização de ataques a um *fortress* de outra equipa, os utilizadores podem usar uma grande variedade de ferramentas, não havendo muitas regras que os utilizadores devem respeitar nos seus ataques. A criação de um *fortress* é exclusiva a equipas que façam parte de um plano que requer pagamento, porém, qualquer utilizador que esteja registado na plataforma, com ou sem equipa ou plano, consegue realizar ataques.

Para as equipas submeterem as vulnerabilidades encontradas durante os ataques e ganharem pontos é necessário na secção de submissão da resposta indicar qual o tipo de vulnerabilidade, a

quem pertence o *fortress* inimigo, e qual o endereço do mesmo, podendo este processo ser verificado na Figura 2**Erro! A origem da referência não foi encontrada.**. Após a vulnerabilidade ser submetida, é analisada por uma equipa do CTF365 que verificará se a vulnerabilidade realmente existe ou não no *fortress* indicado, adicionando pontos à classificação da equipa que detetou a vulnerabilidade.

Figura 2- CTF365 Submeter Vulnerabilidade

2.1.3 Reversing.kr

O Reversing.kr [4] é uma plataforma de desafios de *cracking* e de uso de engenharia reversa. Os desafios que esta plataforma possuí consistem em aplicações para diferentes ambientes (Windows, Linux, Windows Phone, …), podendo ser alguns destes observados na Figura 3**Erro! A origem da referência não foi encontrada.**, e em que o ataque é executado nesses mesmos ambientes. Quando um utilizador resolve um desafio e chega a uma solução correta, de forma a ganhar pontos na plataforma, necessita submeter a solução numa secção própria na plataforma do Reversing.kr. Após obtenção dos pontos, estes passam a ser visíveis na tabela de Classificações da plataforma.

Figura 3- Reversing.kr - Challenges

2.1.4 RingZer0 Team Online CTF

O RingZer0 Team Online CTF [5] consiste numa plataforma onde se encontram disponíveis uma grande variedade de desafios de cibersegurança em jogos *capture the flag*. Estes desafios vão desde criptografia, análise de *malwares*, SQLi, entre outros. Alguns deles podem ser resolvidos diretamente na plataforma, enquanto que noutros a resolução é feita no sistema do utilizador, por exemplo, uma aplicação com vulnerabilidade, ou então através de SSH para uma máquina onde a vulnerabilidade estará presente. Independentemente do tipo de vulnerabilidade, após realizado o ataque com sucesso, é devolvida uma *flag* que pode ser usada seguidamente na página onde o desafio realizado se encontra, obtendo pontos de acordo com o desafio realizado, ilustrado na Figura 4.

A plataforma disponibiliza ainda algumas ferramentas que poderão ser usadas de forma a resolver certos desafios. Possuí ainda um sistema de classificações onde são apresentados os 50 utilizadores com mais pontos na plataforma.

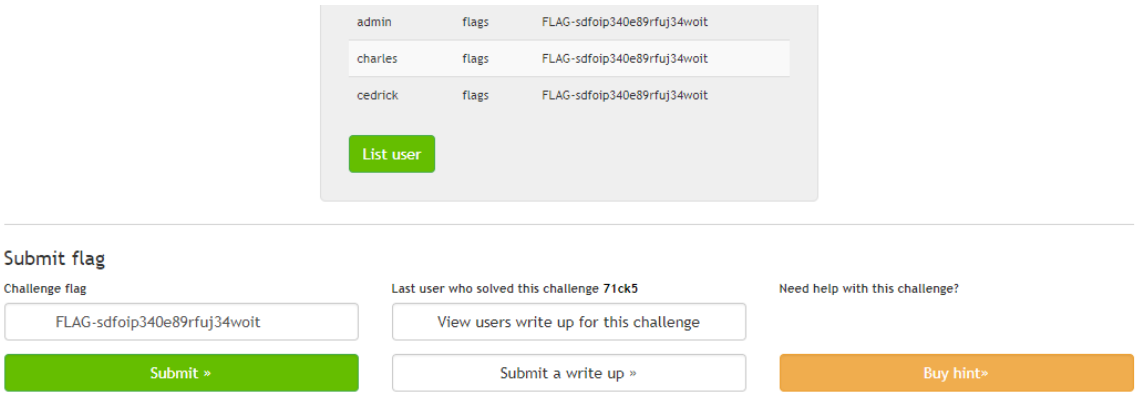


Figura 4- RingZer0Team - Desafio SQLi após ataque

2.1.5 WebGoat

A WebGoat [6] consiste numa aplicação onde se pode encontrar uma variedade de vulnerabilidades comuns em aplicações desenvolvidas em Java e componentes *open source*. Esta plataforma, para além de ter vários desafios sobre diversas vulnerabilidades, que vão desde falhas que podem ser exploradas através de injeção de código malicioso, ataques de SQLi ou de CSRF, entre outros, que podem ser observados na **Erro! A origem da referência não foi encontrada.**, t em ainda lições teóricas acerca dos vários ataques que podem ser testados na plataforma, ilustrados na Figura 5. Nenhum dos desafios presentes na WebGoat necessita de ferramentas adicionais para a sua resolução.

Os desafios encontram-se divididos em dois tipos: desafios para testes das vulnerabilidades nos quais, durante a sua resolução, os utilizadores podem obter dicas de como realizar o ataque, sendo que da sua resolução não resulta pontuação adicional ao utilizador, e desafios competitivos, nos quais não há qualquer tipo de dicas para chegar à resposta e de onde resulta uma *flag* usada para obter pontos pelo desafio realizado.

O servidor para a plataforma WebGoat corre como servidor local, pelo qual se encontra disponível para download na internet.

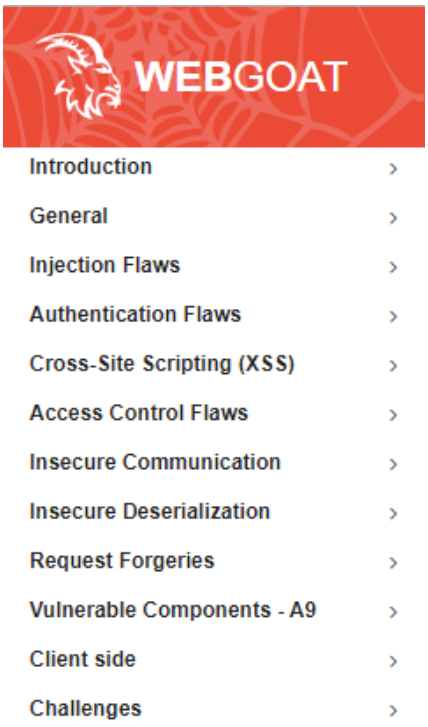


Figura 5- WebGoat - Menu Principal

2.2 Análise crítica das soluções existentes

Todas as aplicações apresentadas reúnem um conjunto de características que vão ao encontro dos objetivos do LabSecurity.

O Hacking-Lab, tem pontos fortes que devem ser referidos, como uma grande variedade de desafios a explorar e que analisam uma grande diversidade de vulnerabilidades, que vão desde desafios fáceis a difíceis. Para além disso, a plataforma online é de fácil uso e possui as informações necessárias à realização de cada desafio. A plataforma apresenta ainda uma secção onde são divulgados eventos de cibersegurança, que apesar de não ir ao encontro dos objetivos especificados para o LabSecurity, não deixa de ser um ponto forte já que esses eventos são uma forma de divulgar informação na área da cibersegurança.

Como pontos fracos o Hacking-Lab apresenta características como a necessidade de ter de instalar a máquina virtual disponibilizada pela plataforma do Hacking-Lab e o acesso através de VPN à plataforma de forma a ter acesso a grande parte dos desafios. Para além disso existem ainda características como o uso de ferramentas adicionais e a forma como a solução encontrada para cada desafio é avaliada.

Ao nível do CTF365, esta plataforma tem como principais pontos fortes a ótima apresentação da plataforma e a variedade de vulnerabilidades que podem ser exploradas devido aos ataques serem executados em *sandboxes*. Tem ainda características positivas como a liberdade na criação dos *fortress* de cada equipa e da defesa dos mesmos, bem como a situação de ataque ao *fortress* de outras equipas. Como pontos fracos apresentam-se características como a falta de informação acerca das várias vulnerabilidades a serem exploradas, a dificuldade que a plataforma apresenta a utilizadores com pouco conhecimento na área da programação e segurança e o método de verificação da vulnerabilidade que exige verificação por parte da gestão da plataforma.

Ao analisar a plataforma Reversing.kr verificaram-se características positivas como a variedade de desafios disponíveis, o método de obtenção automática de uma *flag* após chegar a uma solução correta, e o uso das *flags* para confirmação que a resposta encontrada está correta e obtenção de pontos.

Como características negativas detetou-se que apesar de haver uma boa variedade de desafios disponíveis, todos eles estão relacionados com vulnerabilidades de aplicações, sendo que encontram -se estão ainda limitadas a diversos tipos de sistemas operativos e linguagens de programação o que faz com que seja necessário executar essas aplicações nos sistemas correspondentes. Existe ainda a falta de informação acerca dos vários desafios disponíveis.

O RingZer0 Team Online CTF é uma plataforma que apresenta alguns aspetos positivos como uma vasta variedade de desafios disponíveis, o uso de *flags* para validação e obtenção de pontos por desafio realizado com sucesso, a disponibilização de ferramentas adicionais que podem ser usadas na resolução de certos desafios e a facilidade de uso da plataforma. Como aspetos negativos verificou-se a falta de informação acerca dos desafios disponíveis e vulnerabilidades a explorar e a necessidade de usar SSH para certas máquinas de forma a poderem ser realizados determinados desafios.

O WebGoat, apesar de funcionar como um servidor local, mostra ser uma plataforma bastante completa que apresenta características positivas como a variedade de ataques disponíveis, constituídos por informação acerca dos mesmos, desafios de treino e algumas competições. Como ponto positivo existe ainda o uso de *flags* para a validação e obtenção de pontos para o *ranking*. Esta plataforma apresenta ainda como pontos negativos o facto de não ser uma plataforma online e de ter muitas poucas competições disponíveis.

No geral, todas as plataformas analisadas têm alguns pontos que a plataforma web a desenvolver tem como objetivos, no entanto nenhuma delas tem um sistema de *Quizzes* ou de Propostas de ataques feitas pelos utilizadores e sendo que a única que fornece informação teórica acerca das várias vulnerabilidades a serem exploradas é a plataforma WebGoat que apresenta ser a mais completa para um ambiente de aprendizagem.

Apesar da variedade entre as várias plataformas analisadas, todas elas têm funcionalidades a ter em conta no desenvolvimento da plataforma web LabSecurity.

Como resumo à etapa de análise das soluções já existentes podem ser observadas na Tabela 1 o conjunto de características que serviram como comparação das várias soluções. As características analisadas em cada solução foram se possuem um Sistema de Classificação, ou seja, se cada vez que um utilizador resolve um desafio se o mesmo recebe pontos pelo mesmo e se estes são apresentados em comparação com a pontuação que o resto dos utilizadores obtiveram, o uso de *flags*, que consiste numa característica usada em jogos CTF e que tem como objetivo atribuir um código de solução, normalmente único ao utilizador ou equipa, que é introduzido na aplicação em campos próprios para o mesmo, de forma a receber os pontos pela resolução de determinado desafio. Foi ainda analisado se a verificação das respostas aos diferentes desafios era feita de forma automática pelo sistema, ou se era realizada por membros da parte administrativa da aplicação.

Outra das características analisadas foi se todas os desafios eram resolvidos ao nível da plataforma *web*, ou seja, na própria interface da aplicação, ou se haviam alguns que tinham de ser resolvidos fora da plataforma (por exemplo, através do descarregamento de um ficheiro ao qual seria realizado o ataque e seguidamente usada a *flag* obtida para obter pontos na plataforma *web*). Foram ainda verificadas se cada uma das soluções permitia aos utilizadores da plataforma poderem

fazer propostas de ataques/desafios, se estas continham uma secção de lições teóricas acerca das várias vulnerabilidades (Ataques e Defesas) e ainda se tinham *quizzes* com perguntas com base nas várias vulnerabilidades existentes nas mesmas.

Tabela 1- Aplicações existentes e objetivos propostos

Aplicação	**Sistema de Classificação**	**Uso de Flags**	**Auto-Verificação da Resposta**	**Ataques na Plataforma Web**	**Propostas de Ataques**	**Vulnerabilidades – Lições Teóricas**	**Quizzes**
Hacking-Lab	Sim	Não	Não	Não	Não	Não	Não
CTF365	Sim	Não	Não	Sim	Não	Não	Não
Reversing.kr	Sim	Sim	Sim	Não	Não	Não	Não
RingZeroTeam	Sim	Sim	Sim	Sim	Não	Não	Não
WebGoat	Sim	Sim	Sim	Sim	Não	Sim	Não

Seguidamente apresenta -se a metodologia de desenvolvimento de software escolhida e que se pretende usar no processo de desenvolvimento da plataforma web.

3 Metodologia

De forma a minimizar os riscos de erros e falhas no decorrer do desenvolvimento de uma plataforma *web*, é aconselhado usar metodologias de desenvolvimento de *software*. Apesar de existir uma grande variedade de metodologias, estas encontram-se divididas em dois tipos: metodologias tradicionais e metodologias ágeis, tendo como tradicionais a metodologia em espiral, RUP, entre outras, e como metodologias ágeis o Scrum e o eXtreme Programming.

Para o desenvolvimento deste projeto foi definida como metodologia de desenvolvimento de software o Scrum devido a ser uma metodologia focada em desenvolvimento rápido e iterativo de software. Além isso, o Scrum é fácil de usar e cria a oportunidade de recolher de forma mais flexível os requisitos necessários para o desenvolvimento da plataforma web e permitir que esse processo se desperte ainda numa fase inicial do desenvolvimento do projeto.

Metodologia de desenvolvimento ágil: Scrum

O Scrum [7] é uma estrutura da metodologia de desenvolvimento de software ágil conhecida por ser focada em produzir e entregar produtos de grande qualidade. É um *framework* simples de entender, apesar de ser complexo de colocar em prática, e é usado em trabalhos complexos onde não se consegue prever futuras dificuldades.

Ao nível do projeto em questão, esta metodologia foi usada de forma a ajudar na organização e gestão do mesmo, desde a definição de cada fase do projeto, o que seria desenvolvido ou corrigido em cada uma, às várias reuniões ao longo do desenvolvimento do projeto juntamente com o orientador, bem como a ajudar a perceber o que podia ser melhorado em cada fase ou na tentativa de prever futuros erros.

O Scrum é constituído por um conjunto de cargos, eventos e regras interligados que juntos permitem abordar problemas complexos e desenvolver produto de qualidade ao mesmo tempo.
Para lá disso, permite ainda saber se as técnicas e método de manutenção do projeto são eficazes, fazendo com que seja possível fazer melhorias continuas ao nível do produto a entregar, ao nível da equipa que se encontra a trabalhar no projeto e no ambiente de trabalho.

O Scrum é baseado no controlo empírico de processos e usa um método iterativo e incremental de forma a tentar aumentar a previsão de erros e controlo de riscos.

Em todas as implementações de processos de controlo empírico existem três pilares essenciais a seguir: Transparência, Inspeção e Adaptação.

A Transparência consiste num conjunto de aspetos relevantes no processo de controlo que devem ser visíveis aos responsáveis pelos resultados obtidos.

Este conjunto de aspetos devem ser definidos de forma a que os participantes partilhem um mesmo conhecimento do que lhes é apresentado. Um exemplo desses aspetos é saber quando se pode dizer que determinada funcionalidade é dada como terminada.

A Inspeção consiste em análises frequentes ao trabalho que se está a desenvolver de forma a verificar se não existem variações do que é pedido e do que se está a desenvolver, e são normalmente efetuadas por alguém com experiência na execução dessas análises

A Adaptação consiste na reorganização do projeto caso se detetem desvios nos objetivos do mesmo, sendo que esta deve ser realizada o mais breve possível de forma a minimizar os riscos causados.

O Scrum é constituído por quatro fases que definem o decorrer de uma *Sprint*, sendo estas, por ordem de realização, *Sprint Planning*, *Daily Scrum*, *Sprint Review* e *Sprint Retrospective*.

Uma *Sprint* consiste numa fase do projeto que é iniciada e que tem um espaço de tempo já definido para a conclusão da mesma, não podendo este exceder um mês. Uma nova *Sprint* só começa após a finalização da *Sprint* que se encontra a decorrer.

O uso de *sprints* pelo Scrum é bastante vantajoso já que estas permitem fazer uma previsão do estado do projeto através da inspeção e adaptação pelo menos a cada mês.

Uma *sprint* inicia-se por uma *Sprint Planning* que consiste numa reunião em que participam o proprietário do produto, o gestor do projeto e a equipa de desenvolvimento, em que é decidido o que será entregue no final da sprint e como é que o trabalho realizado vai ser gerido. Nesta reunião é analisado um documento, designado de *Product Backlog*, que contém todas as funcionalidades que se pretende que o produto tenha no final do seu desenvolvimento, e no final da mesma deve resultar um documento, designado de *Sprint Backlog*, que consiste numa lista de todas as funcionalidades a que a equipa de desenvolvimento, juntamente com o gestor do projeto se comprometem a entregar no final da *sprint*.

Ao longo do desenvolvimento do produto são realizadas várias *Daily Scrum*, que consistem em reuniões diárias onde se verifica que trabalho foi realizado desde a última *Daily Scrum* e tentar prever que trabalho será feito até à próxima, ajudando a perceber qual a probabilidade de a equipa de desenvolvimento alcançar o objetivo da *sprint*, e até mesmo aumentar a mesma através de ajustes a vários níveis.

No Final da *sprint* é feita a *Sprint Review* que consiste numa reunião onde participa o proprietário do produto, o gestor de projeto e equipa de desenvolvimento, onde é revisto o trabalho realizado durante a *sprint* e apontadas informações valiosas abordadas durante a reunião.

Entre a *Sprint Review* e a nova *Sprint Planning*, realiza -se a *Sprint Retrospective*, que consiste numa reunião entre o gestor do projeto e a equipa de desenvolvimento com o objetivo de melhorar o rendimento da próxima *sprint* tendo em conta a última.

Todo este processo pode ser observado na Figura 6**Erro! A origem da referência não foi encontrada.**.

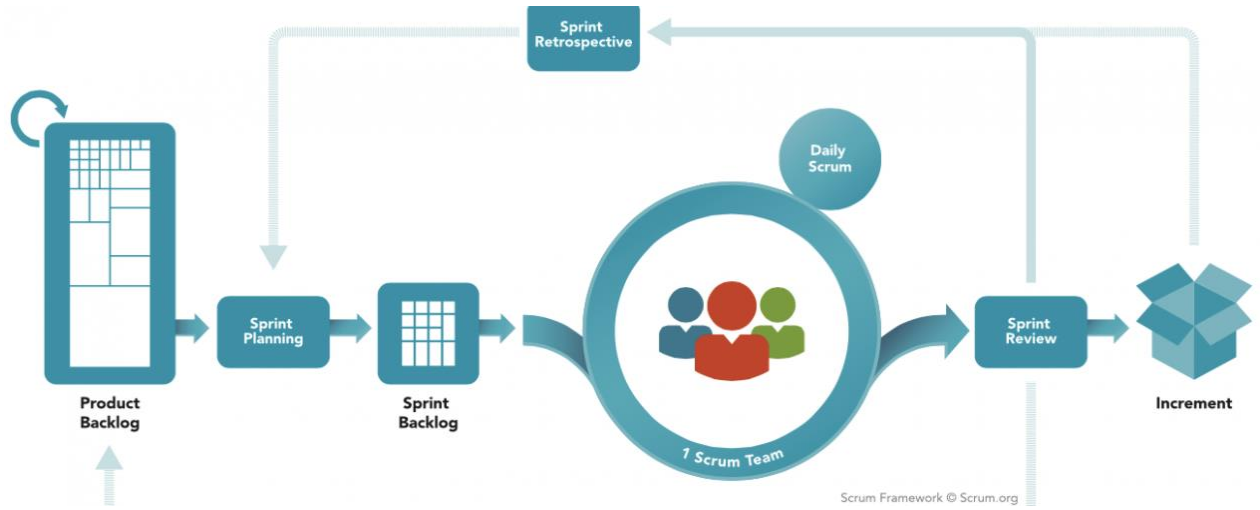


Figura 6- Processo de uma Sprint [7]

4 Análise de Requisitos

Os requisitos de um sistema consistem num conjunto de dados que descrevem as várias funcionalidades que a aplicação faz ou que permite aos seus utilizadores fazer na mesma. Tendo em conta a importância dos requisitos de um sistema e os principais objetivos descritos inicialmente, foi feita uma análise recorrendo à linguagem UML de forma a planear os alguns casos de uso da aplicação.

Feita uma análise aos requisitos do sistema, segue-se abaixo um conjunto de práticas que são importantes a ter em conta durante o desenvolvimento da aplicação, que vão desde os objetivos previsto, atores e respetivos casos de uso, aos vários diagramas UML como diagrama de contexto e diagramas de sequência, entre outros.

4.1 Objetivos Previstos

Após uma análise aos requisitos do sistema, foram definidos os objetivos que se pretende que a aplicação faça:

- Gerir competições da aplicação
    - Criar, editar, eliminar e consultar dados das competições;
- Gerir Desafios
    - Criar, eliminar e consultar dados dos desafios;
- Gerir Classificações (dos desafios)
    - Criar, eliminar e consultar classificações
- Gerir Sugestões de Desafios
    - Consultar, aceitar, rejeitar sugestões.
- Participar, consultar torneios
- Consultar classificações das competições
- Consultar desafios
- Submeter solução para desafio
- Submeter sugestões de desafios

4.2 Atores e respetivos casos de uso

Tendo em conta que a aplicação foi pensada em ser desenvolvida para um meio escolar, ao fazer a análise aos possíveis atores da mesma surgiram os seguintes elementos:

- **Gestor da Plataforma –** Pessoa responsável por gerir a plataforma.
- **Aluno –** Utilizador que pode participar nas várias componentes que a plataforma disponibiliza e às quais está autorizado, desde participar nas competições, realizar desafios e sugerir novos desafios.

De forma a que a plataforma funcione corretamente, existem funcionalidades que são específicas para o gestor da plataforma, sendo todas as outras comuns para alunos e gestor, sendo estas apresentadas na Tabela 2.

Tabela 2- Atores e respetivos casos de uso

Atores	Casos de Uso
Gestor da Plataforma	• Criar, Consultar, Editar, Eliminar Competições • Criar, Consultar, Eliminar Desafios • Ver Sugestões • Aceitar Sugestões • Rejeitar Sugestões
Aluno	• Criar Conta • Ver, Participar Competições • Ver tabela de Classificações de uma competição • Ver, Realizar Desafio • Submeter Sugestões de desafios

4.3 Diagrama de casos de uso

Seguidamente, na Figura 7**Erro! A origem da referência não foi encontrada.**, pode-se observar o diagrama de casos de uso, onde se encontram representados os atores que interagem com a plataforma e os seus respetivos casos de uso generalizados. Todos os casos de uso que irão ser desenvolvidos no decorrer no projeto encontram-se dentro da fronteira.

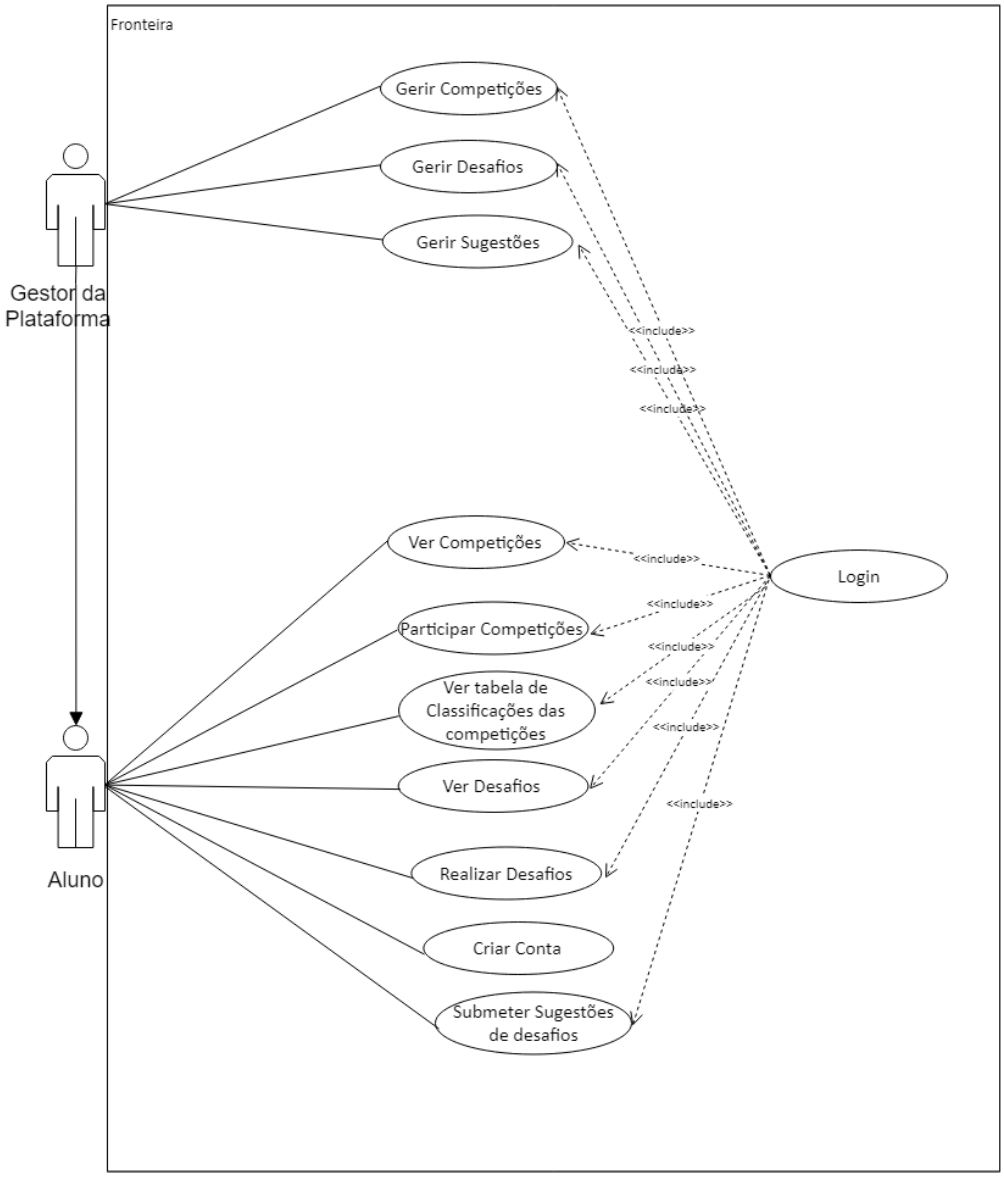


Figura 7- Diagrama de casos de uso

4.4 Diagrama de Contexto

O diagrama de contexto consiste num diagrama de fluxo de dados que representa os processos entre os atores e o sistema. Este diagrama é uma forma de representar o objeto em estudo, o projeto e a sua relação com o ambiente.

Apresenta -se seguidamente, na Figura 8**Erro! A origem da referência não foi encontrada.**, o diagrama de contexto da aplicação, onde se pode observar como entidades externas o gestor da plataforma e o aluno.

Figura 8 - Diagrama de Contexto

4.5 Descrição dos casos de uso e Diagramas de Sequência

A descrição de um caso de uso é um processo indispensável no desenvolvimento e na visão de como a aplicação irá interagir com os atores em diversas funcionalidades, permitindo desta forma fazer um planeamento de como a interface deverá ser, e a lógica por detrás da funcionalidade.

A descrição de um caso de uso é caracterizada por um conjunto de pontos a serem analisados, sendo estes:

- Descrição: Deve ser uma descrição curta e precisa do caso de uso a ser descrito. Deve descrever o objetivo do caso de uso.
- Pré-Condição: Condição ou condições iniciais necessárias a que o caso de uso possa decorrer.
- Caminho Principal: Descreve o caminho que o utilizador deve percorrer para que tudo decorra com sucesso. Descreve ainda a resposta do sistema às ações do utilizador.
- Caminhos Alternativos: Descreve caminhos para os quais poderá ter ocorrido algo errado em determinado passo do caminho principal.

Tendo em conta os vários passos da descrição de casos de uso, são agora realizadas algumas descrições dos casos de uso mais importantes da aplicação.

4.6 Submeter resposta para desafio de uma competição

Na Tabela 3 pode-se observar o desenvolvimento do caso de uso em questão.

Tabela 3 - Descrição do caso de uso "Submeter resposta para desafio de uma competição"

Nome	Submeter resposta para desafio de uma competição
Descrição	Este caso de uso descreve o processo de submeter uma resposta para um desafio de uma competição
Pré-Condição	*Login* válido e registo na competição
Caminho Principal	1. O ator clica para entrar na competição em que o desafio se encontra; 2. O sistema mostra a página da competição com as várias informações sobre a mesma (nome da competição, estado, pontuação total, os desafios com as suas informações (nome, pontuação, tempo, dificuldade e tipo)); 3. O ator clica no desafio que pretende resolver; 4. O Sistema devolve a página com o desafio e as suas características (nome, descrição); 5. O ator introduz e submete a sua resposta; 6. O sistema verifica a resposta dada, avalia, adiciona pontos ao utilizador, calcula o tempo que o mesmo demorou a resolver o desafio e redireciona o ator para a página da competição já atualizada com os pontos e tempo.
Caminhos Alternativos	6.a. Caso a resposta não seja a correta, o Sistema informa o ator que a resposta fornecida está incorreta.

Na Figura 9**Erro! A origem da referência não foi encontrada.** apresenta-se o diagrama de sequência que representa o caminho principal do caso de uso “Submeter resposta para um desafio”.

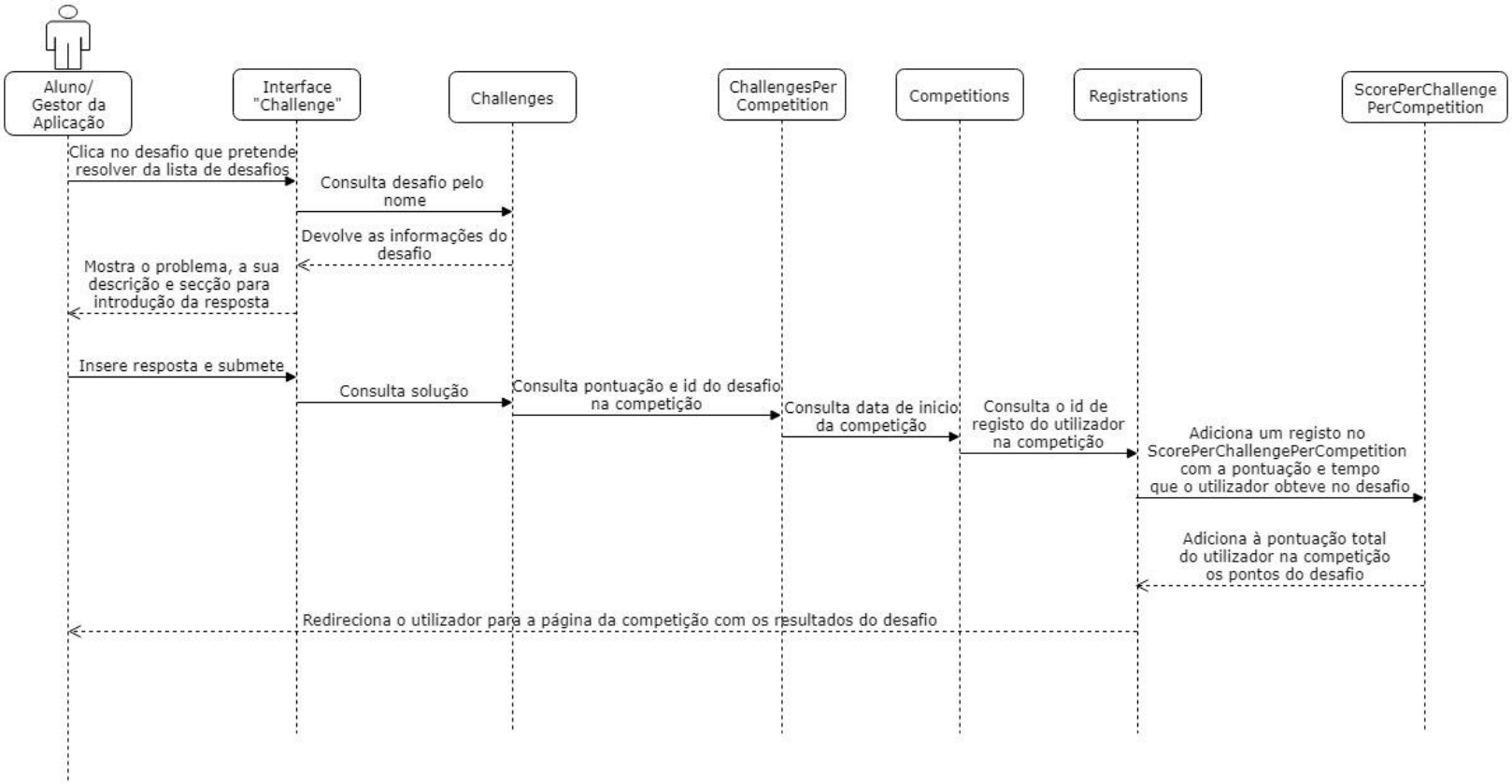


Figura 9 - Diagrama de Sequência "Submeter resposta para um desafio"

4.7 Criar Desafio

Descreve-se seguidamente na Tabela 4 o caso de uso “Criar Desafio”. Este processo passa por definir as várias características do desafio (nome, descrição, pasta do desafio, entre outras).

Tabela 4 - Descrição do Caso de Uso "Criar Desafio"

Nome	Criar Desafio
Descrição	Este caso de uso consiste no processo de criação de um desafio
Pré-Condição	*Login* Válido
Caminho Principal	1. E ator clica em “Add New Challenge” 2. O Sistema mostra uma página com formulário e com os vários campos a preencher para a criação do desafio (Nome, descrição, pasta do desafio, ficheiro principal, solução, classificação e dificuldade); 3. O ator preenche os vários campos e submete a criação do desafio clicando em “Create Challenge”. 4. O sistema adiciona o desafio, informa o ator do desafio ter sido criado com sucesso e redireciona o mesmo para a página de gestão de desafios já com o novo desafio.
Caminhos Alternativos	3.a. O Sistema alerta o ator que existem campos por preencher. 3.b. O Sistema alerta o ator que o nome do desafio ou da pasta em que se encontra já existe.

Na Figura 10**Erro! A origem da referência não foi encontrada.** apresenta-se o diagrama de sequência relativamente ao caminho principal do caso de uso “Criar Desafio”.

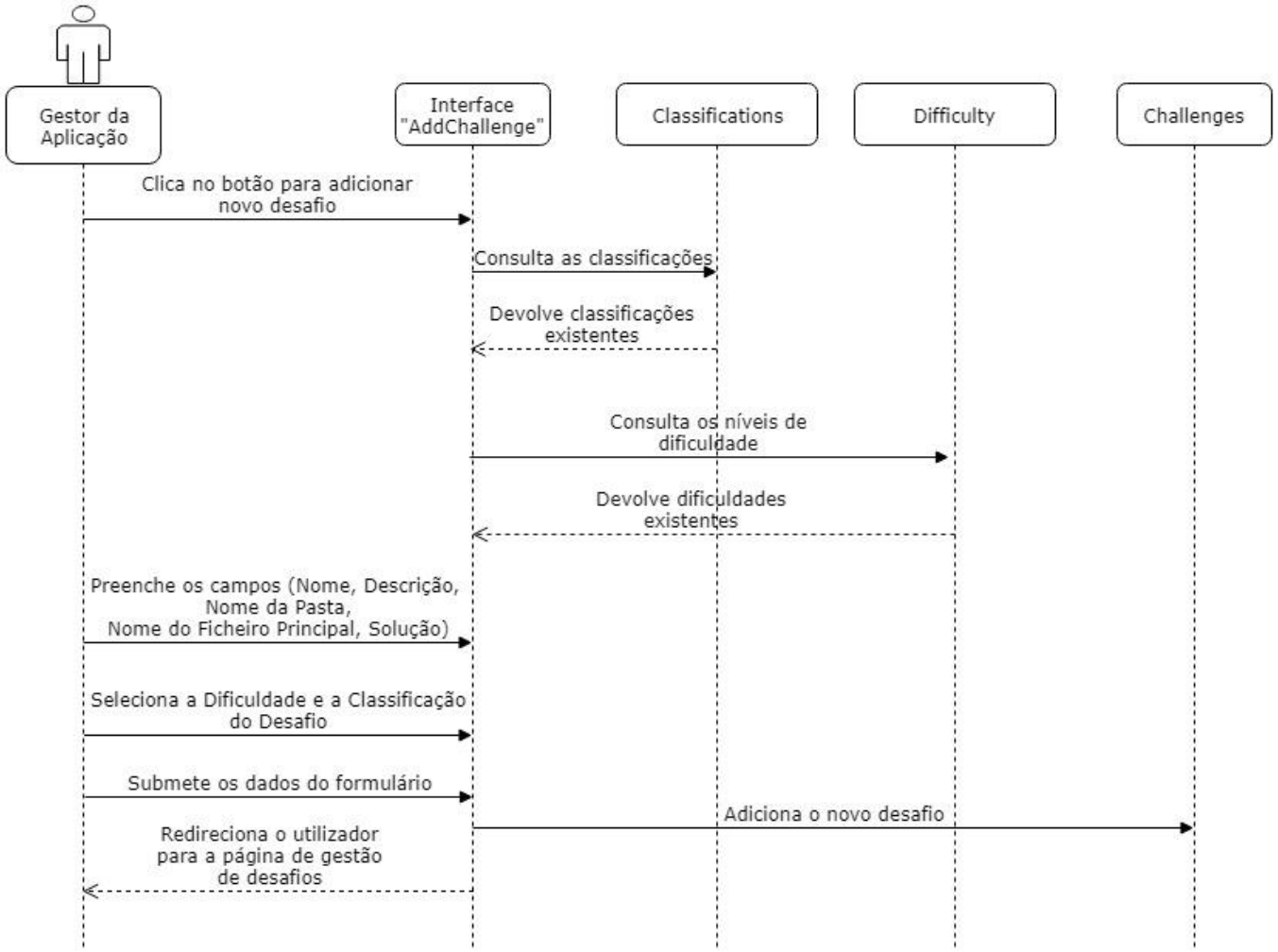


Figura 10 - Diagrama de Sequência do Caso de Uso "Criar Desafio"

4.8 Diagrama de Classes

O Diagrama de classes tem como principal objetivo demonstrar as relações entre as várias classes que fazem parte do sistema. Cada classe é constituída por um nome (o que representa), atributos (informações que são armazenadas e analisadas resultado das várias funcionalidades da aplicação) e por fim as operações que representa podem ser realizadas em cada uma delas pelos atores.

De forma a obter um melhor conhecimento de algumas das classes mais importantes da aplicação, são seguidamente descritas algumas delas.

Na Figura 11**Erro! A origem da referência não foi encontrada.Erro! A origem da referência não foi encontrada.** é possível observar o diagrama de classes da aplicação.

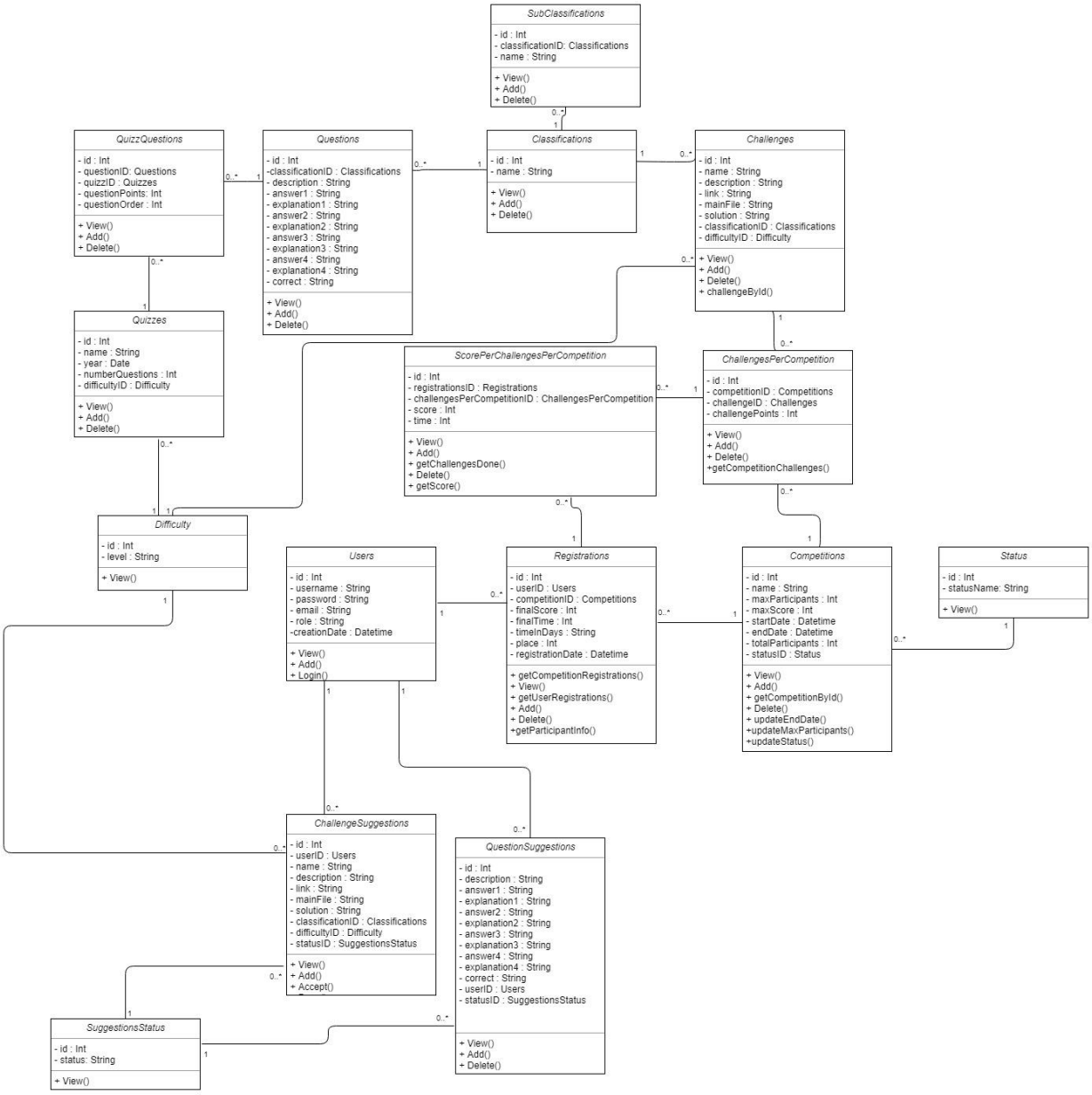


Figura 11 - Diagrama de Classes

A classe *ScorePerChallengePerCompetition* é uma das classes mais importantes da aplicação já que esta interage com duas outras classes bastante importantes, sendo estas a classe *ChallengesPerCompetition*, que faz o armazenamento dos vários desafios e respetivos scores em determinada competição, e a classe *Registrations*, onde são armazenados o número de identificação do utilizador e a Competição em que está a participar, o *Score* Final e o Tempo Final com que o mesmo acabou no final da competição, sendo que estes são ambos calculados através da classes *ScorePerChallengePerCompetition.* Esta classe é umas das classes mais consultadas tendo em conta que o objetivo principal da aplicação é a resolução de desafios de forma competitiva.

Outra classe bastante importante na aplicação é a classe *ChallengeSuggestions*. Apesar de ser uma classe que interage apenas com algumas classes simples, como a classe *Users*, de forma a armazenar quem sugeriu o desafio, a classe *Difficulty*, que permite atribuir um grau de dificuldade ao desafio que se está a sugerir e a classe *Classifications*, de forma a conceder um uma classificação do tipo de desafio que se está a sugerir, esta classe desempenha uma das funcionalidades que distingue a aplicação das estudadas anteriormente, permitindo aos atores fazerem sugestões de novos desafios para serem usados nas competições, podendo estes ser rejeitados ou aceites pelo gestor da plataforma.

Tendo em conta que o ponto fulcral da aplicação é ser competitiva, a classe *Competitions* permite fazer a gestão dos dados associados às competições. Esta classe tem uma ligação com a classe *Registrations*, onde se encontram os registos dos vários utilizadores nas várias competições, bem como o Score e o Tempo com que acabaram no final da competição, sendo depois usados numa Leaderboard. Têm ainda uma ligação com a tabela *challengesPerCompetition*, onde se encontram registados todos os desafios da competição e respetivas pontuações que um participante recebe na competição caso conclua o desafio.

Segue-se o modelo ER, ilustrado na Figura 12**Erro! A origem da referência não foi encontrada.**, baseado no diagrama de classes apresentado.

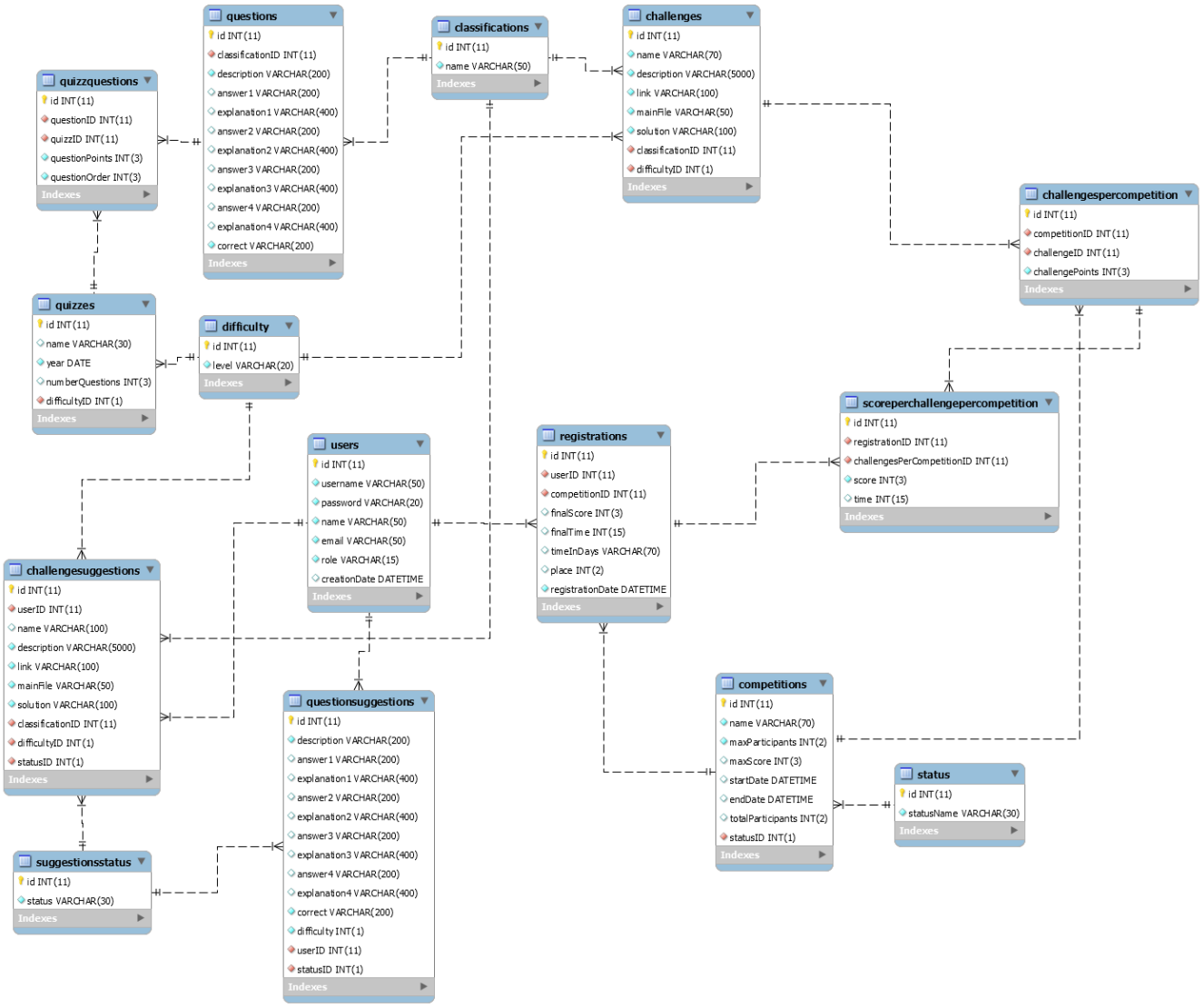


Figura 12 - Modelo ER

5 Implementação da Solução

Após realizado o estudo às várias soluções já existentes no mercado e feito o planeamento das funcionalidades que a aplicação deverá ter através da recolha e análise dos requisitos do sistema, prossegue-se com o desenvolvimento da aplicação *web*.

Para tal, foi necessário escolher a um grupo de tecnologias que permitam o um desenvolvimento da aplicação de forma eficiente.

5.1 Tecnologias Utilizadas

Para o desenvolvimento da aplicação LabSecurity foi feita uma análise às várias tecnologias mais usadas no desenvolvimento de aplicações *web* tendo em conta a sua popularidade, performance e interação entre as mesmas. Feita esta análise, foram definidas inicialmente como tecnologias a usar o AngularJS como *frontend*, o NodeJS como *backend*, tendo em que ambas usam a linguagem de programação JavaScript, sendo usadas ao nível das interfaces o HTML e CSS. Porém, feita uma análise mais aprofundada às mesmas verificou -se que ReactJS é um *framework* de *frontend* mais fácil e cómodo de usar do que AngularJS, permitindo ainda que em pouco tempo se aprenda a trabalhar com o mesmo e a perceber a grande variedade de funcionalidades ao dispor do programador, para lá de permitir uma interação com uma maior variedade de APIs de *backend* do que o AngularJS, passando a ser usado ReactJS no desenvolvimenta do *frontend* da aplicação.

Tendo em conta as boas práticas de desenvolvimento de software, foi usado o sistema de gestão de versões Git de forma a armazenar a aplicação e registar o desenvolvimento da mesma.

São agora apresentadas abaixo todas as tecnologias usadas no desenvolvimento do projeto.

5.1.1 JavaScript

Javascript consiste numa linguagem de programação dinâmica e interpretada [8] (não necessita ser compilada) bastante usada no desenvolvimento de aplicações *web*. Hoje em dia a maioria dos *sites* usa Javascript, principalmente devido a esta permitir interagir com o DOM, resultando numa maior comunicação entre o utilizador e as páginas da aplicação. Esta linguagem

é bastante conhecida e usada ainda ao nível da programação orientada a objetos. Apesar de JavaScript ser bastante usada ao nível do *browser*, não significa que não possa ser usada como *backend*, como por exemplo *NodeJS*.

Nesta aplicação são usadas duas bibliotecas JavaScript, o *NodeJS* para o desenvolvimento do *backend* e o ReactJS para o *frontend*.

5.1.2 NodeJS

O NodeJS consiste numa *framework* de JavaScript designado para *backend*. Este recorre ao motor V8 da Google que faz com que a velocidade de processamento e execução de código no NodeJS seja extremamente rápido. Além disso, o *NodeJS* usa funções do tipo *callback* de forma a realizar processamentos *single-threaded*, ou seja, processamento assíncrono de dados, fazendo com que esta seja uma *framework* de alta performance para implementação do *backend* de uma aplicação Web. As funções de *callback* consistem em funções que são passadas para dentro de outras de forma a serem chamadas assim que essa função acaba [9]. Ao nível da organização de código, tal como outras *frameworks*, o NodeJS usa um sistema de módulos. Cada ficheiro é um módulo diferente e, de forma a poderem ser usados, é necessário existir uma exportação dos mesmo no seu ficheiro específico e uma importação para o ficheiro no qual se quer usar determinado módulo.
Os módulos encontram -se divididos em três categorias diferentes:

- Módulos Nativos: São aqueles que já vem instalados juntamente com o NodeJS. Um exemplo deste tipo é o FS, módulo usado para lidar com ficheiros.
- Módulos de Terceiros: Tal como o módulo diz, são módulos de terceiros e podem ser instalados através de pacotes que se encontram em repositórios. Como exemplo temos o *npm* (*Node Package Manager*) que é um dos módulos mais usados devido a servir como um gestor de pacotes de JavaScript permitindo aos programadores descobrir pacotes de código que pode ser reutilizado pelos mesmos [10].
- Módulos Locais: Módulos criados pelo próprio programador dentro da aplicação.

5.1.3 ReactJS

O ReactJS consiste numa *framework* de JavaScript *OpenSource* criada pelo *Facebook* designada para *frontend* de aplicações *web.* Além da versão *ReactJS* existe ainda uma outra versão para desenvolvimento *mobile* denominada React Native.

Como sistema de módulos, o ReactJS recorre ao uso de componentes para o desenvolvimento da interface da aplicação. Estes componentes, para além de possuírem a parte daquilo que o utilizador vai ver quando usa a aplicação, possuem ainda a lógica daquilo que vai ser ou não mostrado ao utilizador. Pode ainda conter dados que poderão ser usados no desenho da aplicação [11].

O React foi pensado com a ideia de que a manipulação do DOM é um processo bastante dispendioso e que deve ser minimizado, fazendo com que a fluidez da aplicação seja muito maior.

De forma a resolver o problema com o DOM, o React permite ao programador usar um *Virtual* DOM para voltar a desenhar as páginas às quais são feitas alterações, em vez do DOM. Este *Virtual DOM* faz com que não seja necessário desenhar novamente todo o componente ou componentes, mas sim apenas as partes que foram alteradas, fazendo com o que número de operações ao nível do DOM diminua bastante, tornando a aplicação bastante mais fluida [11].
Para entender um pouco mais este processo, apresenta-se na Figura 13**Erro! A origem da referência não foi encontrada.** a diferença entre *Virtual* DOM *e* DOM.

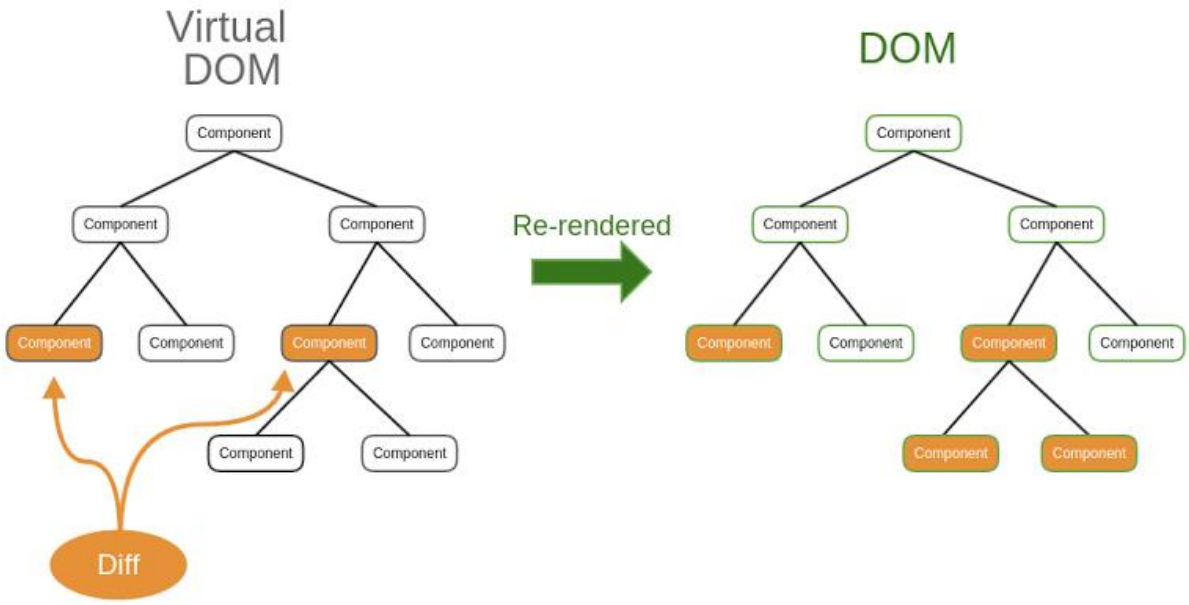


Figura 13 - Virtual DOM vs DOM

Fonte: https://medium.com/naukri-engineering/naukriengineering-virtual-dom-fa8019c626b

5.1.4 HTML

HTML [12] é a linguagem mais usada no desenvolvimento para páginas *web*. Usa -se para apresentar informações ou dados numa página *web* através de *tags* que definem a estrutura das páginas, desde cabeçalhos a parágrafos, entre outros elementos, que são interpretados pelo *browser* e mostrados ao utilizador e é normalmente usada em conjunto com CSS e JavaScript.

5.1.5 CSS

CSS [13] é um mecanismo que permite ao programador definir estilos para os vários elementos usados em linguagens de marcação, como o HTML, através de regras. Apesar de poderem ser aplicados estilos diretamente aos elementos HTML, não é aconselhado, sendo que os estilos devem ser criados num ficheiro à parte onde são usados seletores de forma a indicar a que elemento HTML vai ser aplicado certo estilo, e declarações, que permitem identificar qual a propriedade a ser alterada desse mesmo elemento, fazendo desta forma com que se obtenha uma melhor organização e como resultado um código mais transparente.

5.1.6 MySQL

O MySQL [14] é um sistema de gestão de bases de dados relacionais *open-source* da Oracle e é baseado em *SQL*. Este sistema, apesar de abranger uma grande variedade de tipos de aplicações, é normalmente associada a aplicações *web*.

O sistema é bastante conhecido e popular devido a ser um componente muito importante ao nível da *stack* LAMP.

O MySQL é bastante reconhecido pela sua velocidade, fácil uso, capacidade *multi-threading*, segurança, devido ao uso de SSL, entre outras características que o levam a ser dos sistemas de gestão de bases de dados relacionais mais usado.

5.1.7 Git

O Git [15] é um sistema de gestão de versões *open-source* e grátis. Este permite ao utilizador, ao longo do desenvolvimento de um projeto, criar versões através de *commits* aos quais pode aceder, ou em caso de ocorrem problemas, reverter para uma versão estável. Este sistema permite ainda uma maior facilidade de desenvolvimento em equipa, já que possui funcionalidades como criação de *branches*.

Uma das características únicas do Git é que ao contrário de outros sistemas de gestão de versões, sempre que são feitas alterações ao projeto, após feito o *commit* apenas são atualizados os ficheiros que sofreram alterações, e não todo o projeto em si, fazendo com que o processo de adição de uma nova versão seja bastante rápido.

Para este projeto foi usada a plataforma GitHub para armazenar o projeto e registar o progresso do mesmo. Este pode ser consultado em https://github.com/ruiparedes/ProjetoFinal.

5.2 Arquitetura do Sistema

A arquitetura de um sistema consiste na descrição dos vários elementos que fazem parte do sistema, como estão organizados ou integrados e como interagem entre si de forma a criar todo o sistema.

Para arquitetura deste projeto, foi usada uma arquitetura de *web services*. Esta é uma arquitetura flexível e modular conhecida por ser usada em páginas *web* mais interativas onde existe a comunicação entre vários sistemas/aplicações. Esta arquitetura permite adicionar elementos que aumentam a qualidade e segurança da aplicação a ser desenvolvida, como mecanismos de autenticação, *tokens*, entre outros.

Uma arquitetura de *web services* permite ainda fazer uma separação do *frontend* e do *backend*, fazendo com que seja fácil melhorar a organização do projeto, e ter um melhor controlo em relação à lógica do negócio e ao que é mostrado ao utilizador através do consumo de dados. Este consumo e exposição de dados é realizada através de linguagens preparadas para tal, sendo as principais JSON e XML.

Neste projeto foi usada JSON, tendo esta características como ser de fácil compreensão, completamente independente de qualquer linguagem de programação, e ideal como uma linguagem de troca de informação ou consumo/exposição de dados.

Um dos pontos mais importantes pelo uso desta arquitetura é que a lógica feita no *backend* pode ser consumida para diversas aplicações, desde *web*, a *mobile*, entre outras.

Na Figura 14 pode observar -se a arquitetura do sistema do projeto, que consiste entre a interação dos utilizadores com o *frontend*, sendo este constituído pelo ReactJS e este com o *backend*, constituído pelo NodeJS e pela base de dados MySQL.

O processo passa-se da seguinte forma:

1. O utilizador faz um pedido ao *frontend* para, por exemplo, lhe fornecer informações acerca das competições. Estes pedidos são realizados através de *fetchs*. Um *fetch* permite ao utilizador aceder e manipular partes da tecnologia HTTP como *requests* e o *responses*.
2. Por sua vez, o ReactJS efetua o pedido ao NodeJS indicando-lhe qual a *Route* que pretende aceder.
3. O NodeJS recebe o pedido e verifica a *Route* que lhe é enviada. Após verificar a mesma, o NodeJS realiza a lógica que essa *Route* possuí, caso existam operações a realizar antes de

ser feito um pedido à base de dados, e por fim o pedido é enviado para a base de dados MySQL através de uma *query*.

4. Por fim, a base de dados recebe a *query*, executa -a, e envia o resultado para o NodeJS, que por sua ver devolve os dados ao ReactJS onde estes serão apresentados ao utilizador.

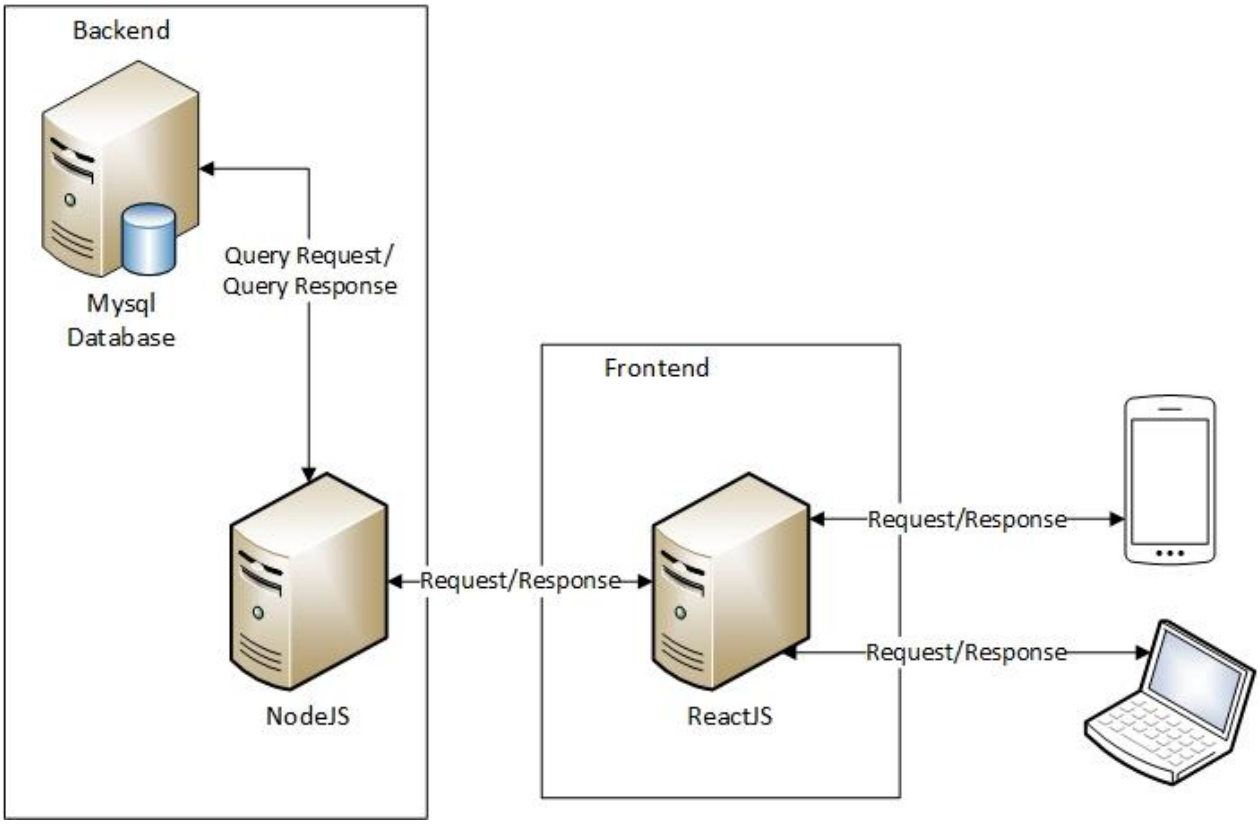


Figura 14 - Arquitetura do Sistema

É seguidamente apresentado na Figura 15 um organigrama dos componentes implementados na aplicação e componentes que serão implementados num trabalho futuro (a vermelho).

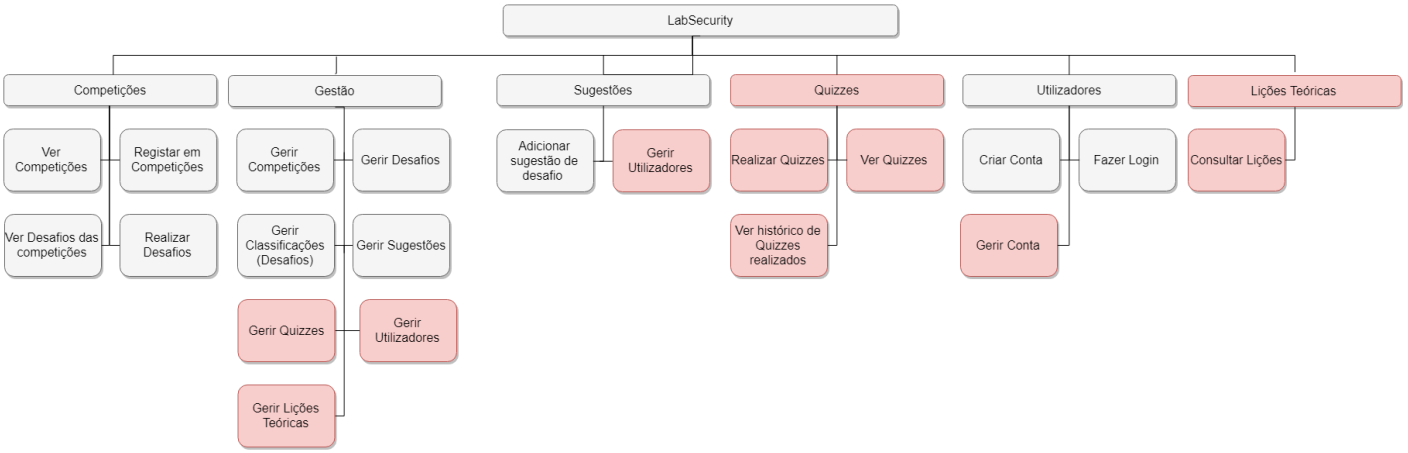


Figura 15 - Organigrama dos Componentes da Aplicação

De seguida são apresentadas alguns processos e funcionalidades que desempenham um papel fundamental ao bom funcionamento da aplicação.

5.3 Carregamento da interface de um desafio

A funcionalidade de carregamento da interface de um desafio consiste em conseguir mostrar ao utilizador, através de uma única rota, qualquer desafio ao qual tenha autorização para aceder. De forma a entender melhor o porquê desta funcionalidade ser tão importante e perceber melhor como este processo se realiza, é feita uma descrição abaixo de todo o processo.

Devido ao ReactJS funcionar através de componentes, e tendo em conta que cada componente necessita ser importado como uma *route* do mesmo, de forma a se poder ter acesso aos vários desafios, houve a necessidade de criar um componente onde os *imports* dos mesmos seriam realizados dinamicamente.

Este processo consiste em, para cada desafio, em vez de cada um deles possuir uma *Route* própria (exemplo: /desafio1, /desafio2), todos os desafios acedem a uma mesma rota dinâmica (exemplo: /desafios/: nomeDoDesafio) com o componente em questão e que tem como principal objetivo conseguir mostrar os componentes de qualquer desafio sem a necessidade de serem criadas rotas específicas para cada um.

Isto não seria necessário se não existisse a funcionalidade de Sugerir Desafios, já que todos os desafios que seriam apresentados aos utilizadores seriam aqueles que o gestor da plataforma criou e implementou na aplicação, podendo estes terem rotas especificas para cada um.

Desta forma, foi criada a rota "/Challenge/:name", como se pode observar na Figura 16,**Erro! A origem da referência não foi encontrada.** que é usada para desenhar qualquer componente relativo aos desafios dinamicamente, onde ":name" será o nome do desafio que se quer mostrar ao utilizador.

```
<Route exact path="/challenge/:name/" component={Challenge} />
```

Figura 16 - Route Dinâmica para os desafios

Dentro do componente dinâmico, de forma a importar o desafio desejado, é usado o campo "link" do desafio, que representa a pasta e nome do ficheiro do desafio, sendo esta uma das informações passadas do componente que mostra todos os desafios de uma competição quando o utilizador clica no desafio que quer resolver.

Pode-se observar na Figura 17**Erro! A origem da referência não foi encontrada.** a forma como os desafios estão a ser importados no componente dinâmico.

```
var challengeLink = this.props.location.state.challengeLink;

const myImportFunction = () => import('../challenges/'+link);
const Component = importedComponent(myImportFunction);
return <Component challengeID={challengeID}  challengeName={challengeName} competitionID={competitionID} value={values}/>;
```

Figura 17 - Import dinâmico de desafios

5.4 Encerramento de uma competição e criação da tabela de Classificações da mesma

Tendo em conta que a aplicação foi pensada para ter uma componente competitiva, houve a necessidade de criar um processo que fizesse com que uma competição fechasse assim que a sua data de fim fosse atingida e para lá disso, gerasse uma tabela com a classificação de cada participante na competição. Sem esta funcionalidade, a competição teria de ser fechada manualmente pelo gestor da aplicação e criada a tabela de classificações da mesma posteriormente.

O processo de encerramento de uma competição consiste em detetar quando a data do sistema é maior ou igual à data de fim registada para cada competição. Para fazer essa deteção é usada ao nível do *backend*, no NodeJS a função "setInterval()" onde é designada uma função e de quanto em quanto tempo se quer que aquela mesma função seja chamada. Foi configurada para ser chamada de 1 em 1 segundo uma função que vai buscar todas as competições existentes e que estão a decorrer de momento e verifica se a data de fim da competição já foi alcançada ou não. Caso esta se confirme, são executadas sequencialmente um conjunto de funções em conjunto com *promises*, que consistem num tipo de funções assíncronas, e ciclos *for* de forma a alcançar os seguintes objetivos:

1. Fechar a competição
2. Verificar quais os desafios que foram realizados com sucesso pelos participantes
3. Calcular o Score Final e o Tempo Final com que cada participante acabou no final da competição
4. Inserir ao registo do participante o *Score* Final, o Tempo Final e a Posição em que ficaram na competição.

A forma como é calculado o *Score* Final de cada participante consiste na simples soma do score obtido na realização de cada desafio da competição, no caso do Tempo Final este é obtido verificando o maior valor de tempo registado na realização de um desafio, já que o tempo registado na conclusão de um desafio é a diferença entre a data do sistema no momento de conclusão e a data de inicio da competição, obtendo um valor em milissegundos. A posição de cada participante na competição é calculada usando o *Score* Final e o Tempo Final,

Quanto maior for o *Score* Final e quanto menor for o Tempo Final, mais alto ficará na tabela de Classificações o participante.

Dado que o valor do Tempo Final está em milissegundos, de forma a ser apresentado de uma forma mais "amigável" aos utilizadores, foi feita uma conversão de milissegundos para dias, horas, minutos e segundos. A fórmula usada para esta conversão pode ser observada na Figura 18**Erro! A origem da referência não foi encontrada.**.

```
finalParticipantTime = participantScoreOnChallenge[0].time;
var ms = finalParticipantTime;
var days, hours, mnts, seconds;
seconds = Math.floor(ms / 1000);
mnts = Math.floor(seconds / 60);
seconds = seconds % 60;
hours = Math.floor(mnts / 60);
mnts = mnts % 60;
days = Math.floor(hours / 24);
hours = hours % 24;
var timeInDays = days + " day(s) " + hours + " hour(s) " + mnts + " Minute(s) " + seconds + " Second(s)";
```

Figura 18 - Conversão de milissegundos para Dias, horas, minutos e segundos

5.5 Interfaces da aplicação *web*

Nesta fase vão ser abordadas as várias interfaces que fazem parte do projeto de forma a obter uma melhor compreensão das várias funcionalidades presentes em cada uma delas e da interação que os utilizadores irão ter com as mesmas.

5.5.1 Homepage (Interface Inicial)

Primeira interface apresentada ao utilizador assim que acede à aplicação, mesmo que ainda não tenha feito *login* ou *signup* na mesma, podendo ser observada na Figura 19.

O objetivo principal desta interface é dar as boas vindas ao utilizador à aplicação, sendo ainda apresentadas possibilidades de realizar *Login*, caso este já possua conta na aplicação, ou registar-se na mesma, podendo depois aceder à secção das competições ou às sugestões de desafios através da mesma.

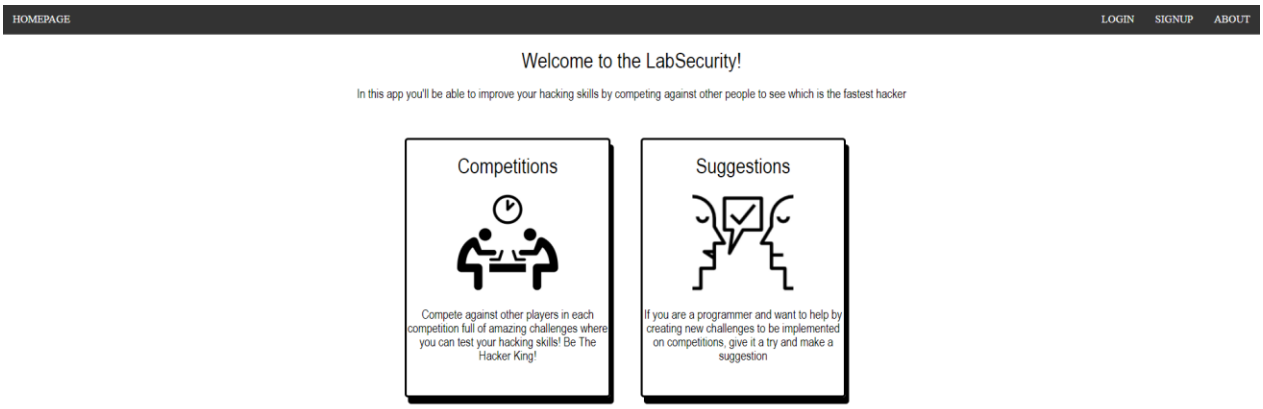


Figura 19 - Interface Inicial

5.5.2 Interface de Registo

Caso o utilizador se sentir tentado a testar a aplicação, este pode dirigir-se à página de registo, ilustrada na Figura 20**Erro! A origem da referência não foi encontrada.**, onde poderá criar a conta que irá usar para poder aceder às várias funcionalidades que a aplicação o autoriza.

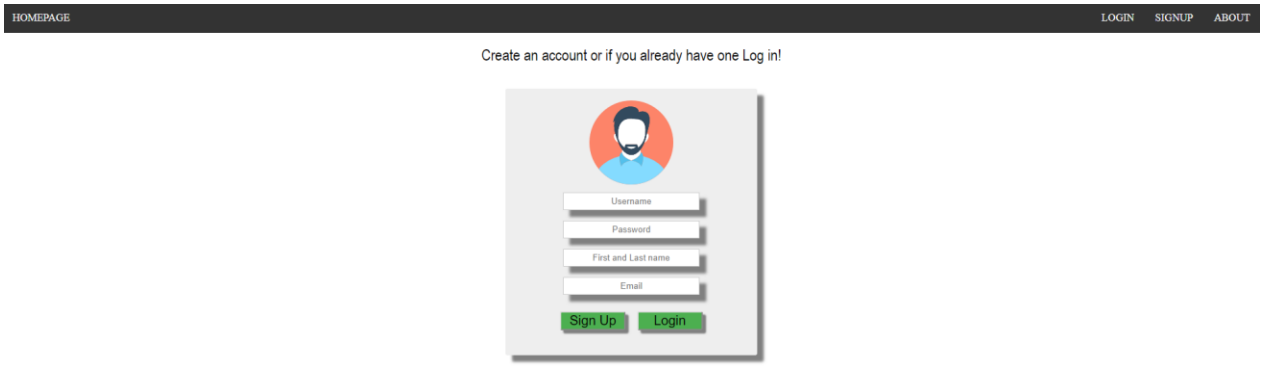


Figura 20 - Interface de Registo

5.5.3 Interface de Login

No caso o utilizador já possua uma conta na aplicação, este poderá realizar o *Login* com a mesma e começar a explorar a aplicação. A interface de *Login* pode ser observada na Figura 21 .

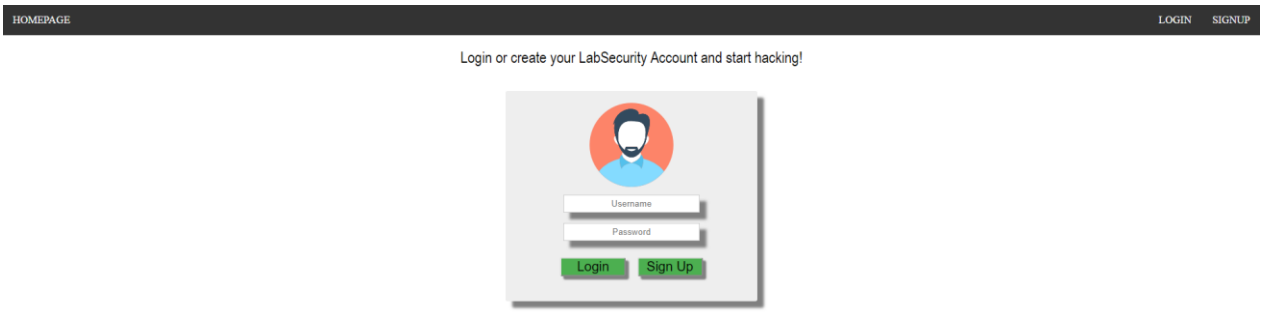


Figura 21 - Interface de Login

5.5.4 Interface de competições

Na interface de competições são apresentadas ao utilizador as várias competições disponíveis e as várias informações sobre a mesma (nome, número de participantes, datas de início e fim e estado), e em que se poderá registar caso ainda existam vagas para entrar e que a competição ainda esteja a decorrer. Ao registar-se numa competição passa a aparecer um botão que lhe permite entrar na competição e verificar os vários desafios da mesma.

Quando uma competição termina, passa a ser mostrado um botão que permite ao utilizador aceder à tabela de classificações daquela competição, independentemente se participou ou não na mesma.

Na Figura 22**Erro! A origem da referência não foi encontrada.** pode observar-se a interface de competições, onde o utilizador com Login efetuado já participou numa competição e a mesma já terminou.

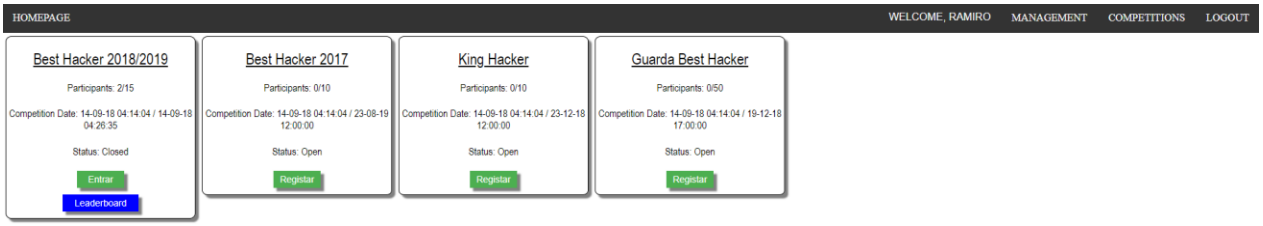


Figura 22 - Interface das Competições

5.5.5 Interface dos desafios de uma competição

Na interface dos desafios de uma competição são apresentados ao utilizador os vários desafios que a competição possuí, o nome, estado e score total da competição, como ainda o score total na competição que o mesmo tem. Para cada desafio são apresentadas várias informações como o nome do mesmo, o score do desafio e se o utilizador já resolveu o mesmo, o tempo que demorou a resolver o mesmo em milissegundos, a dificuldade e o tipo de desafio. Quando uma competição termina, o utilizador pode continuar a aceder a esta interface, porém não será capaz de voltar a resolver os desafios da mesma. Isto também acontece caso o utilizador já tenha resolvido determinado desafio da competição, o mesmo não poderá voltar a resolvê-lo. A interface dos desafios de uma competição pode ser observada na Figura 23 **Erro! A origem da referência não foi encontrada.**.

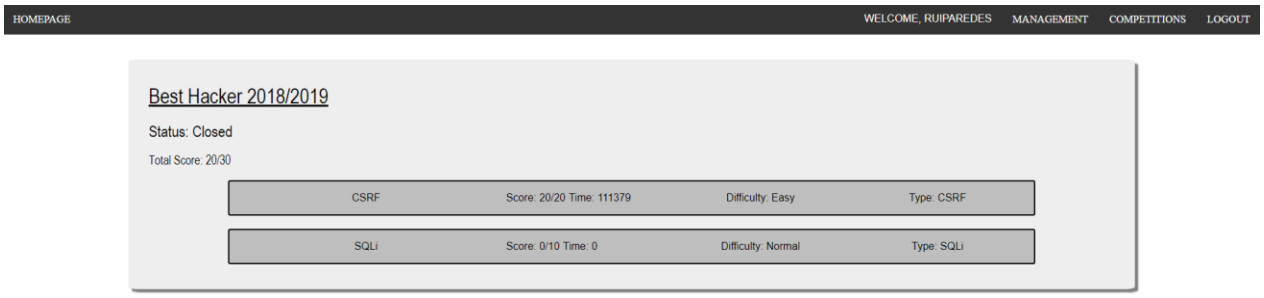


Figura 23 - Interface dos desafios de uma competição

5.5.6 Interface do desafio

Tal como referido anteriormente, devido ao componente que desenha as interfaces dos desafios ser dinâmico, a interface do desafio pode ser diferente dependendo do desafio em si. Por norma, a interface de cada desafio é constituída pelo nome do desafio, descrição e secção de *input* da resolução.

Na Figura 24**Erro! A origem da referência não foi encontrada.** pode observar -se a interface do desafio CSRF.

Figura 24 - Interface do desafio

5.5.7 Interface da tabela de classificações de uma competição

A interface da tabela de classificações é apenas acessível ao utilizador assim que uma competição acabar.

Nesta poderá verificar os vários utilizadores que participaram na competição, o lugar em que ficaram, o resultado final obtido e o total de tempo que demoraram a obter aquele resultado total.

Encontra-se ilustrado na Figura 25**Erro! A origem da referência não foi encontrada.** a interface da tabela de classificações de uma competição, em que neste caso, apenas dois utilizadores participaram, obtiveram o mesmo resultado final, mas em que um ganhou ao outro no tempo demorado.

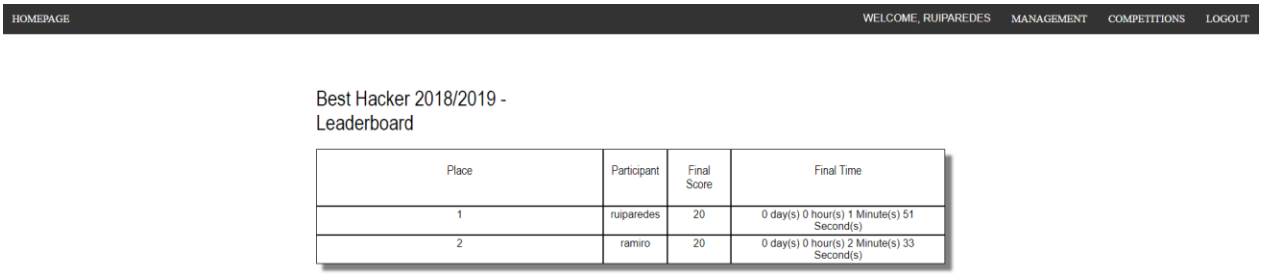


Figura 25 - Interface de Classificações

5.5.8 Interface de Gestão da Aplicação

A interface de gestão da aplicação tem acesso restrito, pelo que apenas o gestor da plataforma pode aceder à mesma.

Esta interface permite aceder aos vários elementos da aplicação que podem ser geridos, e realizar operações como adicionar, editar ou eliminar, entre outras relativamente às competições, desafios, classificações de desafios e sugestões.

É apresentada na Figura 26**Erro! A origem da referência não foi encontrada.** a interface relativamente à gestão de elementos da aplicação.

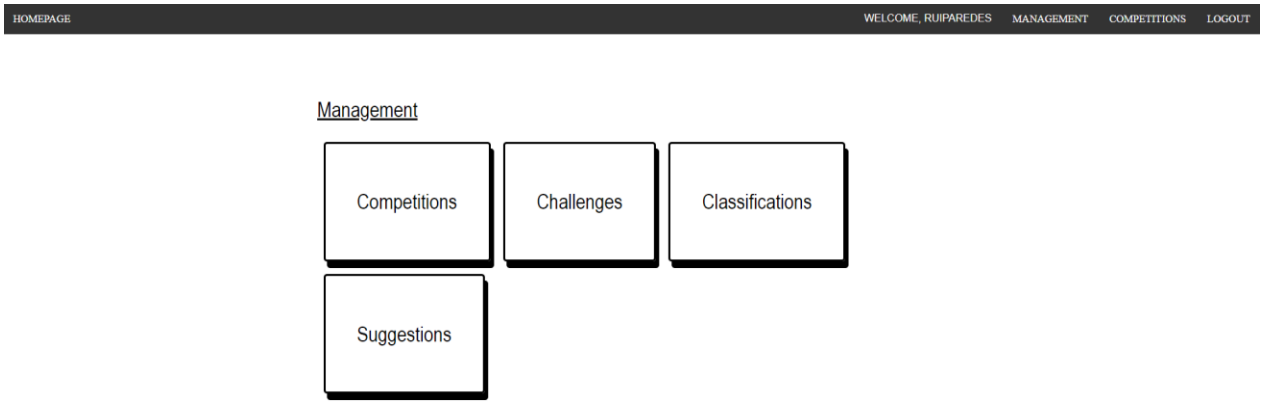


Figura 26- Interface de Gestão da Aplicação

6 Testes

Devido ao processo de desenvolvimento de software ser um processo disposto a erros, foi necessário usar uma metodologia de forma a verificar as várias funcionalidades e verificar se os resultados obtidos são os esperados ou não.

Optou-se então pela utilização do Postman [16], um software bastante conhecido e que é usado para realizar pedidos de HTTP, e realização de testes, facilitando o processo de desenvolvimento de *APIs*.

Através de coleções, o Postman permite aos seus utilizadores uma fácil organização dos vários pedidos que quer realizar e sendo depois possível realizar testes a conjuntos de coleções de forma fácil e eficiente.

O Postman permite ainda a criação de *workspaces* que permite a organização de coleções por equipas, podendo estas ser partilhadas entre workspaces.

Pode ser verificado na Figura 27**Erro! A origem da referência não foi encontrada.** um exemplo bastante simples de um teste realizado de forma a saber se o pedido efetuado para adicionar um novo desafio foi realizado com sucesso ou não. Este tipo de testes foi o mais realizado, de forma a verificar se as informações enviadas estavam a ser inseridas com sucesso ou não.

```
1 pm.test("response is ok", function () {
2     pm.response.to.have.status(200);
3 });
```

Figura 27 - Bloco de código de um teste

Neste caso pode-se verificar através da Figura 28 que o pedido não foi realizado com sucesso, tendo -se obtido um erro de código 400, em vez do código de sucesso 200.

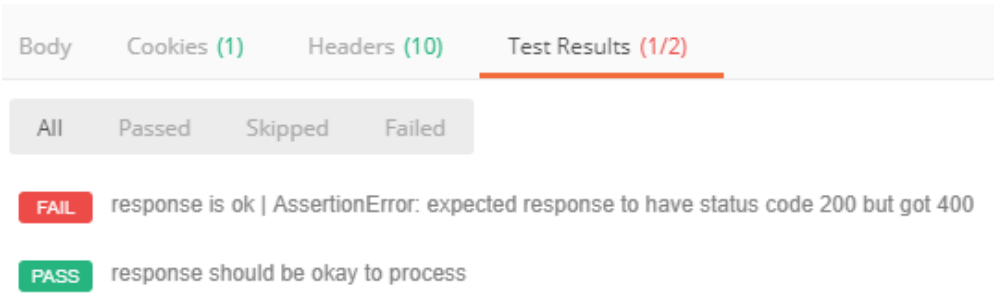


Figura 28 - Resultado de um teste

7 Conclusão

Ao longo do projeto foi desenvolvida uma aplicação *web* com o principal objetivo de permitir aos utilizadores desenvolverem as suas capacidades ao nível da programação e segurança através de competições de *hacking* entre os mesmos. Para além disso, um dos principais objetivos era também servir como repositório aos desafios desenvolvidos pelos alunos da UC de Programação e Segurança, de forma aos mesmos poderem ser usados em várias competições e não serem desperdiçados após entrega e apresentação dos mesmos.

Neste projeto foram realizados com sucesso dos objetivos propostos inicialmente a implementação, gestão e realização de competições, bem como dos desafios que integram as mesmas. Foram ainda implementados dois desafios, ambos com verificação automática da resposta, e sistema de classificação por pontuação e tempo para as várias competições.

Além disso, foi ainda implementada a funcionalidade de sugerir desafios, bem como a zona de gestão dos mesmos, de forma a poder aceitar ou rejeitar os mesmos.

Porém, ficaram por ser realizados alguns dos objetivos propostos inicialmente, como a implementação de *quizzes*, onde seriam também usadas as perguntas semanais desenvolvidas pelos alunos da UC de Programação e Segurança e a secção de lições teóricas onde poderiam ser consultadas várias informações acerca dos diversos tipos de ciberataques reconhecidos pela sociedade, bem como formas de prevenir ou diminuir os riscos causados pelos mesmos.

O desenvolvimento deste projeto apresentou um nível de dificuldade bastante elevado devido à utilização de tecnologias recentes e que não tinham sido abordadas com detalhe durante a licenciatura, pelo qual teve de ser realizado um estudo mais aprofundado das mesmas, das várias funcionalidades que as estas disponibilizam e do uso de módulos de terceiros para facilitar o desenvolvimento de certas funcionalidades na aplicação.

Como trabalho futuro existem várias funcionalidades e correções que poderão ser aplicadas à aplicação *web*. De forma a tornar a aplicação mais amigável à aprendizagem de programação e segurança, seriam implementadas as interfaces de realização de *quizzes*, bem como a secção de lições teóricas. Para além disso, seriam adicionadas algumas validações e notificações às várias interfaces da aplicação, tendo em conta que existem ainda algumas falhas a esse nível.

De forma a tornar a aplicação mais segura, seriam implementados *Tokens* de forma a não ser possível a utilizadores sem autorização para tal ter acesso a informações provenientes do *backend*

através das suas rotas. Por fim, e de forma aos alunos a terem acesso à aplicação fora da escola, seria efetuado o *deploy* usando um serviço *cloud*.

8 Bibliografia

[1] M. Brito, “Codeflex - Aplicação Web de Programação Competitiva,” IPG - ESTG, Guarda, Julho 2018.

[2] “Hacking-Lab,” [Online]. Available: https://www.hacking-lab.com/index.html. [Acedido em 12 Junho 2018].

[3] “CTF365,” [Online]. Available: https://ctf365.com. [Acedido em 12 Junho 2018].

[4] “Reversing.kr,” [Online]. Available: http://reversing.kr/. [Acedido em 13 Junho 2018].

[5] “RingZer0 Team Online CTF,” [Online]. Available: https://ringzer0team.com. [Acedido em 13 Junho 2018].

[6] “OWASP WebGoat Project,” [Online]. Available: https://www.owasp.org/index.php/Category:OWASP_WebGoat_Project. [Acedido em 13 Junho 2018].

[7] “SCRUM,” [Online]. Available: https://www.scrum.org/resources/what-is-scrum. [Acedido em 17 Junho 2018].

[8] D.Flanagan, “Introduction to JavaScript,” em *JavaScript : The Definitive Guide*, Agosto 1996, p. 1.

[9] A. Nandaa, “Introduction to Node.js,” em *Beginning API Development with NodeJS*, Julho 2018, p. 1.

[10] A. Nandaa, “Module Categories,” em *Beginning API Development with Node.js*, Julho 2018, p. 1.

[11] P. S. Vipul A M, “What is React?,” em *ReactJS by Example - Building Modern Web Applications with React*, 2016, p. 1.

[12] J. Duckett, “Introducing HTML,” em *Beginning Web Programming with HTML, XHTML, and CSS*, 2004, p. 4.

[13] J. Duckett, “Introducing CSS,” em *Beginning Web Programming with HTML, XHTML, and CSS*, 2014, p. 10.

[14] P. DuBois, “Why Choose MySQL?,” em *MySQL*, 2008, pp. 4-6.

[15] B. S. Scott Chacon, “A Short History of Git,” em *Pro GIT*, 2005, pp. 5-6.

[16] “Postman,” [Online]. Available: https://www.getpostman.com/. [Acedido em 14 Setembro 2018].

9 Anexos

9.1 Código

9.1.1 Conjunto de funções para o processo de encerrar uma competição e gerar a tabela de classificação da mesma

```
function inicialFunction() {
    var goCheckCompetitionData = checkCompetitionDate();
    goCheckCompetitionData.then(function (result) {
        var competitions = result;
        doLoopThroughCompetitions(competitions);
    })
}

function checkCompetitionDate() {
    return new Promise(function (resolve, reject) {
        const SELECT_ALL_COMPETITIONS_QUERY = 'SELECT * FROM competitions';
        con.query(SELECT_ALL_COMPETITIONS_QUERY, (err, results) => {
            if (err) {
                reject(err);
            }
            else {
                resolve(results);
            }
        });
    })
}

async function doLoopThroughCompetitions(competitions) {
    for (var i in competitions) {
        if ((competitions[i].endDate <= Date.now()) && competitions[i].statusID == 2) {
            await closeCompetition(competitions[i].id);
        }
    }
}

function checkParticipantsPlaces(competitionClosed) {
    return new Promise(function (resolve, reject) {
        const SELECT_PARTICIPANTS_ORDERBY_FinalScore_FinalTime = `SELECT id, finalScore,
finalTime from registrations where competitionID = ${competitionClosed} ORDER BY finalScore
DESC, finalTime ASC`;
```

```
        con.query(SELECT_PARTICIPANTS_ORDERBY_FinalScore_FinalTime, (err, participantsOrderBy)
=> {
            if (err) {
                reject(err);
            }
            else {
                resolve(participantsOrderBy);
            }
        })
    })

}

function closeCompetition(competitionToClose) {
    return new Promise(function (resolve, reject) {
        const ALTER_COMPETITION_STATE_QUERY = `UPDATE competitions SET statusID = 4 WHERE id =
${competitionToClose}`;
        con.query(ALTER_COMPETITION_STATE_QUERY, (err, competitionAltered) => {
            if (err) {
                console.log(err);
            }
            else {
                var goCheckCompetitionChallenges =
checkCompetitionChallenges(competitionToClose);
                goCheckCompetitionChallenges.then(function (competitionChallenges) {
                    var selectedChallenges = competitionChallenges;
                    var competitionParticipants = checkAllParticipants(competitionToClose);
                    competitionParticipants.then(function (participants) {
                        var allParticipants = participants;
                        doLoopThroughParticipantsAndChallenges(allParticipants,
selectedChallenges, competitionToClose);
                    })
                })
            }
        })
    })
}

function checkCompetitionChallenges(competitionID) {
    return new Promise(function (resolve, reject) {
        const SELECT_ALL_COMPETITION_CHALLENGES_QUERY = `SELECT * from ChallengesPerCompetition
where competitionID = ${competitionID}`;
        con.query(SELECT_ALL_COMPETITION_CHALLENGES_QUERY, (err, competitionChallenges) => {
            if (err) {
                reject(err);
```

```
            }
            else {
                resolve(competitionChallenges);
            }
        })
    })
}
function checkAllParticipants(competitionID) {
    return new Promise(function (resolve, reject) {
        const SELECT_ALL_PARTICIPANTS_QUERY = `SELECT * FROM registrations where competitionID
= ${competitionID}`;
        con.query(SELECT_ALL_PARTICIPANTS_QUERY, (err, participants) => {
            if (err) {
                reject(err);
            }
            else {
                resolve(participants);
            }
        })
    })

}
async function doLoopThroughParticipantsAndChallenges(participants, challenges,
competitionToClose) {
    for (var participant in participants) {
        let finalParticipantScore = 0
        let finalParticipantTime = 0;
        for (var challenge in challenges) {
            await checkParticipantScoreChallenge(participants[participant].id,
challenges[challenge].id).then(function (participantScoreOnChallenge) {
                if (participantScoreOnChallenge.length != 0) {
                    finalParticipantScore += participantScoreOnChallenge[0].score;
                    if (finalParticipantTime < participantScoreOnChallenge[0].time) {
                        finalParticipantTime = participantScoreOnChallenge[0].time;
                        var ms = finalParticipantTime;
                        var days, hours, mnts, seconds;
                        seconds = Math.floor(ms / 1000);
                        mnts = Math.floor(seconds / 60);
                        seconds = seconds % 60;
                        hours = Math.floor(mnts / 60);
                        mnts = mnts % 60;
                        days = Math.floor(hours / 24);
                        hours = hours % 24;
                        var timeInDays = days + " day(s) " + hours + " hour(s) " + mnts + "
Minute(s) " + seconds + " Second(s)";
```

```
                        timeinDays = timeInDays.toString()
                    }
                    createScoreboard(participants[participant].id, competitionToClose,
finalParticipantScore, finalParticipantTime, timeInDays);
                }

            });
        }
    }
    checkParticipantsPlaces(competitionToClose).then(function (participantsOrderBy) {
        var pos = 1;
        for (var rank in participantsOrderBy) {
            insertParticipantPlace(participantsOrderBy[rank].id, pos);
            pos++;
        }
    })
}
function insertParticipantPlace(participantID, place) {
    return new Promise(function (resolve, reject) {
        const INSERT_PARTICIPANT_PLACE_QUERY = `UPDATE registrations SET place = ${place} where
id= ${participantID}`;
        con.query(INSERT_PARTICIPANT_PLACE_QUERY, (err, updatedParticipantPlace) => {
            if (err) {
                reject(err);
            }
            else {
                resolve(updatedParticipantPlace);
            }
        })
    })
}

function checkParticipantScoreChallenge(participantID, challengeID) {
    return new Promise(function (resolve, reject) {
        const CHECK_PARTICIPANT_SCORE_CHALLENGE_QUERY = `SELECT score, time from
scorePerChallengePerCompetition where registrationID = ${participantID} and
challengesPerCompetitionID = ${challengeID}`;
        con.query(CHECK_PARTICIPANT_SCORE_CHALLENGE_QUERY, (err, participantScoreOnChallenge)
=> {
            if (err) {
                console.log(err);
            }
            else {
                resolve(participantScoreOnChallenge);
            }
```

```
        })
    })
}
function createScoreboard(participantID, competitionID, totalScore, totalTime, timeInDays) {

    return new Promise(function (resolve, reject) {
        console.log('Time in Days');
        console.log(timeInDays);
        const GENERATE_SCOREBOARD_COMPETITION_QUERY = `UPDATE registrations SET finalScore=
${totalScore}, finalTime = ${totalTime}, timeInDays ='${timeInDays}'  where id=
${participantID} and competitionID = ${competitionID} `;
        con.query(GENERATE_SCOREBOARD_COMPETITION_QUERY, (err, scoreboard) => {
            if (err) {
                reject(err);
            }
            else {
            }
        })
    })
}
```

9.1.2 Método para adicionar pontos ao utilizador após realizar com sucesso um desafio

```
app.post('/api/scorePerChallengePerCompetition/getScore', (req, res) => {
        const userID = req.body.userID;
        const competitionID = req.body.competitionID;
        const challengeID = req.body.challengeID;

        const SELECT_COMPETITION_CHALLENGE_QUERY = `SELECT * from
challengesPerCompetition where competitionID = ${competitionID} and challengeID=
${challengeID}`;
        con.query(SELECT_COMPETITION_CHALLENGE_QUERY, (err, results) => {
            if (err) {
                return res.send(err);
            }
            else {
                const challengeScore = results[0].challengePoints;
                const competitionChallengeID = results[0].id;

                const SELECT_COMPETITION_QUERY = `SELECT * from competitions
where id = ${competitionID}`;
                con.query(SELECT_COMPETITION_QUERY, (err, competitionInfo) => {
                    if (err) {
                        return res.send(err);
                    }
```

```
                    else {
                        const competitionStartDate =
competitionInfo[0].startDate;
                        var time = Date.now() - competitionStartDate;

                        const SELECT_PARTICIPANT_ID = `SELECT * from
registrations where userID = ${userID} and competitionID = ${competitionID}`;
                        con.query(SELECT_PARTICIPANT_ID, (err, participantInfo)
=> {
                            if (err) {
                                return res.send(err);
                            }
                            else {
                                var registrationID = participantInfo[0].id;
                                var participantFinalScore =
participantInfo[0].finalScore;
                                const ADD_SCORE_PARTICIPANT_QUERY = `INSERT INTO
scorePerChallengePerCompetition (registrationID, challengesPerCompetitionID,
score, time) VALUES(${registrationID}, ${competitionChallengeID},
${challengeScore}, ${time})`;
                                con.query(ADD_SCORE_PARTICIPANT_QUERY, (err,
insertedInfo) => {
                                    if (err) {
                                        return res.send(err);
                                    }
                                    else {
                                        participantFinalScore =
participantFinalScore + challengeScore;
                                        console.log(participantFinalScore);
                                        const UPDATE_PARTICIPANT_TOTAL_SCORE =
`UPDATE registrations SET finalScore = ${participantFinalScore} where
id=${registrationID}`;
                                        con.query(UPDATE_PARTICIPANT_TOTAL_SCORE,
(err, updatedParticipantFinalScore) => {
                                            if (err) {
                                                return res.send(err);
                                            }
                                            else {
                                                return res.status(200).json({
                                                    message: 'Points were added
to the Participant!'

                                                })
                                            }
                                        })
```

```
                                    }
                                })
                            }
                        })

                    }
                })


            }
        })
    })
```

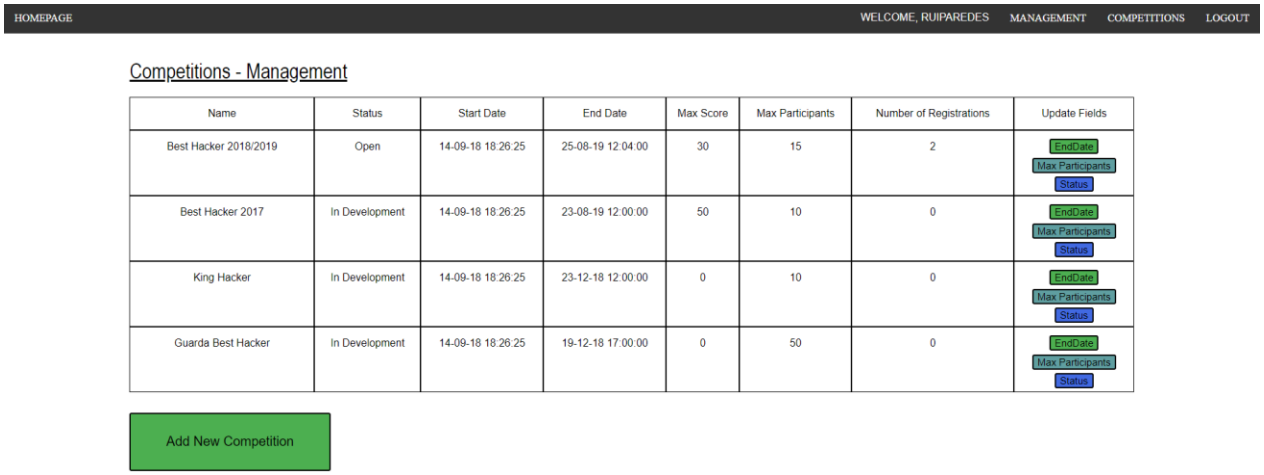
9.1.3 Método de sugerir um novo desafio

```
var multer = require('multer');
var storage = multer.diskStorage({
    destination: (req, file, cb) => {
        cb(null, './uploads/')
    },
    filename: (req, file, cb) => {
        cb(null, file.fieldname + '-' + Date.now() + '.zip');
    }
});


var upload = multer({ storage: storage }).single('file');

    app.post('/api/challengeSuggestions/Add', (req, res) => {

        upload(req, res, (err) => {
            if (err) {
                console.log("Erro no upload do ficheiro")
                res.send(err);
            }
            else {
                console.log(req.file);
                const userID = req.body.userID;
                const name = req.body.name;
                const description = req.body.description;
                var folderName = req.file.filename;
                folderName = folderName.substring(0,
folderName.lastIndexOf('.'));
                const link = req.file.destination + folderName;
                const mainFile = req.body.mainFile;
```

```
                const solution = req.body.solution;
                const difficultyID = req.body.difficultyID;
                const statusID = 1;
                const classificationID = req.body.classificationID;
                const ADD_CHALLENGESUGGESTION_QUERY = `INSERT INTO
challengeSuggestions (userID, name, description,  link, mainFile, solution,
classificationID,difficultyID, statusID) VALUES( ${userID}, '${name}',
'${description}', '${link}', '${mainFile}', '${solution}',
${classificationID},${difficultyID}, ${statusID})`;
                con.query(ADD_CHALLENGESUGGESTION_QUERY, (err, results) => {
                    if (err) {
                        res.send(err);
                    }
                    else {
                        console.log("Challenge suggested:", results);
                        return res.send({
                            "code": 200,
                            "success": "Challenge Added to the suggestions"
                        });

                    }
                })
            }
        })
    });
```

9.2 Interfaces

9.2.1 Gerir Competições



9.2.2 Sugerir Desafio