PROJETO - LabSecurity

Rui Paredes

RELATÓRIO PARA A OBTENÇÃO DO GRAU DE LICENCIADO

EM ENGENHARIA INFORMÁTICA

Jul/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 julho de 2018

## Agradecimentos

## Resumo

## Abstract

## Índice

## Siglário

# Introdução

Este documento desenvolvido pelo aluno Rui Paredes no âmbito da unidade curricular Projeto de Informática, da Licenciatura de Engenharia Informática, descreve o projeto realizado para aprovação à mesma.

O Projeto desenvolvido consiste numa plataforma web de programação e segurança, onde podem ser obtidas informações sobre vários ciberataques, realizar *quizzes* e competir contra outros utilizadores em competições de *hacking*.

## Motivação

Todos os anos, alunos da UC de Programação e Segurança desenvolvem exemplos que ilustram vulnerabilidades, ataques e defesas na área da cibersegurança. Porém, após a apresentação dos mesmos para avaliação, estes exemplos provavelmente não voltam a ser usados para o estudo de determinadas vulnerabilidades.

Por estas razões 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.

Este laboratório de programação e segurança para lá de disponibilizar informações acerca de várias vulnerabilidades, ciberataques existentes e como 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.

## Objetivos da Plataforma

Esta plataforma tem como principais objetivos:

* Realizar desafios de cibersegurança
* Participar em competições de cibersegurança
* Verificação automática das respostas a quizzes ou desafios.
* Sistema de Ranking por pontos
* Plataforma online
* Realizar Quizzes
* Ver lições teóricas sobre as vulnerabilidades/ataques
* Inserção de Propostas

## Fases do Trabalho

## Estrutura do Documento

# 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 completar todos os objetivos propostos 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.

## 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 três sugeridas pelo Orientador, sendo estas o Reversing.kr, RingZer0Team e WebGoat.

Na Tabela 1 são apresentados os objetivos definidos inicialmente para a plataforma web do LabSecurity e se cada solução existente contém essas características ou não.

Tabela 1- Aplicações existentes e objetivos propostos

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Aplicação** | **Ranking** | **Uso de Flags** | **Auto- Verificação da Resposta** | **Ataques na Plataforma Web** | **Propostas de Ataques** | **Vulnerabilidades - Lessons** | **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 |

### Hacking Lab

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

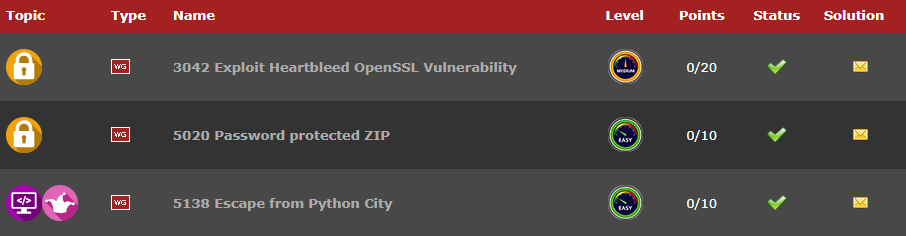
Ao nível dos desafios, o Hacking-Lab apresenta uma variedade de desafios. Algumas destas competições requerem que o utilizador use software adicional para resolver determinados desafios.

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 através de VPN a esta.

Após resolvido um desafio e submetido, 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 ao *rank* do utilizador.

Figura 1- Desafios Hacking- Lab

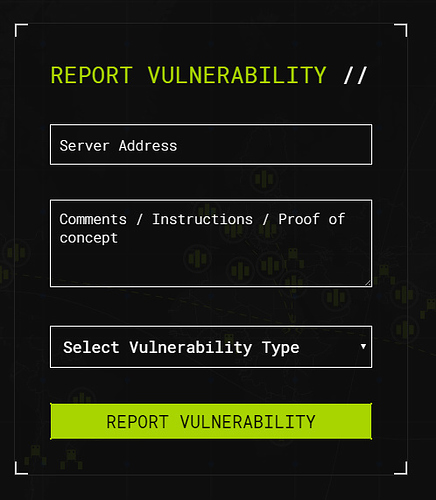


### CTF365

O CTF365 [2] 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 atacam os serviços de equipas inimigas. Os serviços de cada equipa encontram -se nos vários “*Fortress*” que estas possuem, sendo que um *fortress* corresponde a uma máquina virtual atribuída para a equipa. Na realização de ataques a um *fortress* de outras equipa, os utilizadores podem usar uma grande variedade de ferramentas, não havendo uma grande quantidade de 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.

De forma às 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. 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 ao *rank* da equipa que detetou a vulnerabilidade.

Figura 2- CTF365 Submeter Vulnerabilidade



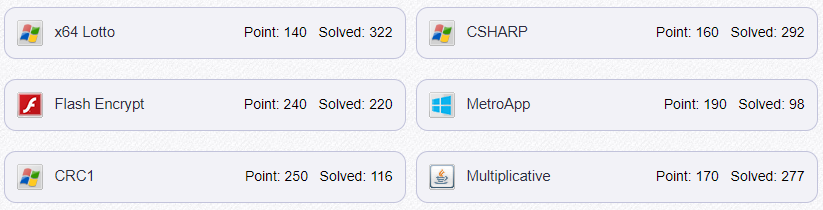
### Reversing.kr

O Reversing.kr [3] 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, 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 mesma numa secção própria na plataforma do Reversing.kr. Após obtenção dos pontos, estes passam a ser visíveis na tabela de *Ranking* da plataforma.

Possuí ainda uma secção onde se podem encontrar plataformas semelhantes a esta.

Figura 3- Reversing.kr - Challenges

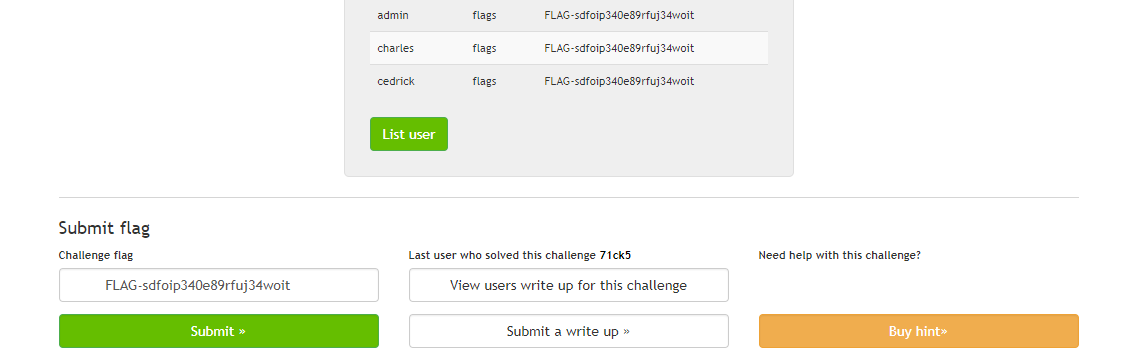


### RingZer0 Team Online CTF

O RingZer0 Team Online CTF [4] 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, onde alguns deles podem ser resolvidos diretamente na plataforma, enquanto que outros 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. A plataforma disponibiliza ainda algumas ferramentas que poderão ser usadas de forma a resolver certos desafios. Possuí ainda um *Scoreboard* onde são apresentados os 50 utilizadores com mais pontos na plataforma.

Figura 4- RingZer0Team - Desafio SQLi após ataque

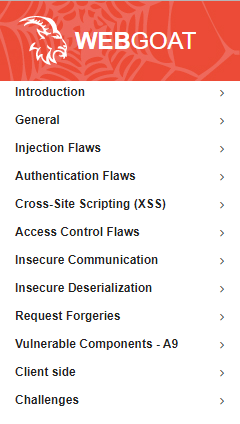


### WebGoat

A WebGoat [5] consiste numa aplicação onde se pode encontrar uma variedade de vulnerabilidades comuns em aplicações desenvolvidas em Java e componentes *open source*. Esta plataforma, para lá de ter vários desafios sobre diversas vulnerabilidades, que vão desde falhas que podem ser exploradas através de injeção de código malicioso, ataques de SQLi por exemplo, a ataques de CSRF, entre outros que podem ser observados na Figura 5, tem ainda lições teóricas acerca dos vários ataques que podem ser testados na plataforma. Todos os desafios presentes na WebGoat não necessitam de ferramentas adicionais para a sua resolução, e encontram -se divididas 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



## 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 lá 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 entrada a grande parte dos desafios. Para lá 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 verificou -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 verificou -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 a característica de esta 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 de todas as exploradas. Apesar da variedade entre as várias plataformas analisadas, todas elas têm funcionalidades a ter em conta no desenvolvimento da plataforma web LabSecurity.

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

# 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 (XP).

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 [6] 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.

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*.

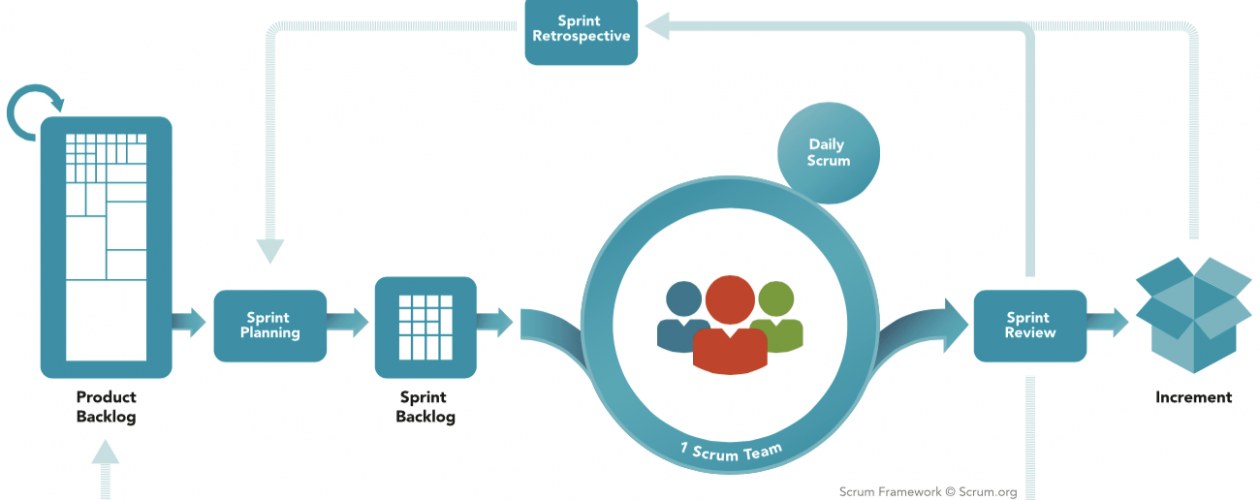
Ao longo do desenvolvimento do produto são realizadas várias *Daily Scrum*, que consistem em reuniões diárias onde se verifica que trabalho foi realizado desde a última *Daily Scrum* e tentar prever que trabalho será feito até à próxima, ajudando a perceber qual a probabilidade da equipa de desenvolvimento 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.

Figura 6- Processo de uma Sprint [6]



# 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.

## Objetivos Previstos

Após uma análise aos requisitos do sistema, foram definidos os objetivos que são previstos que a aplicação permita fazer:

* 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 sugestão.
* Participar, consultar torneios
* Consultar classificação do torneio
* Consultar desafios
* Submeter solução para desafio
* Submeter sugestão de desafio

## 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(as) responsável por gerir a plataforma.
* **Aluno –** Utilizadores que podem participar nas várias componentes que a plataforma disponibiliza e às quais está autorizado, desde participar nas competições, realizar desafio 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.

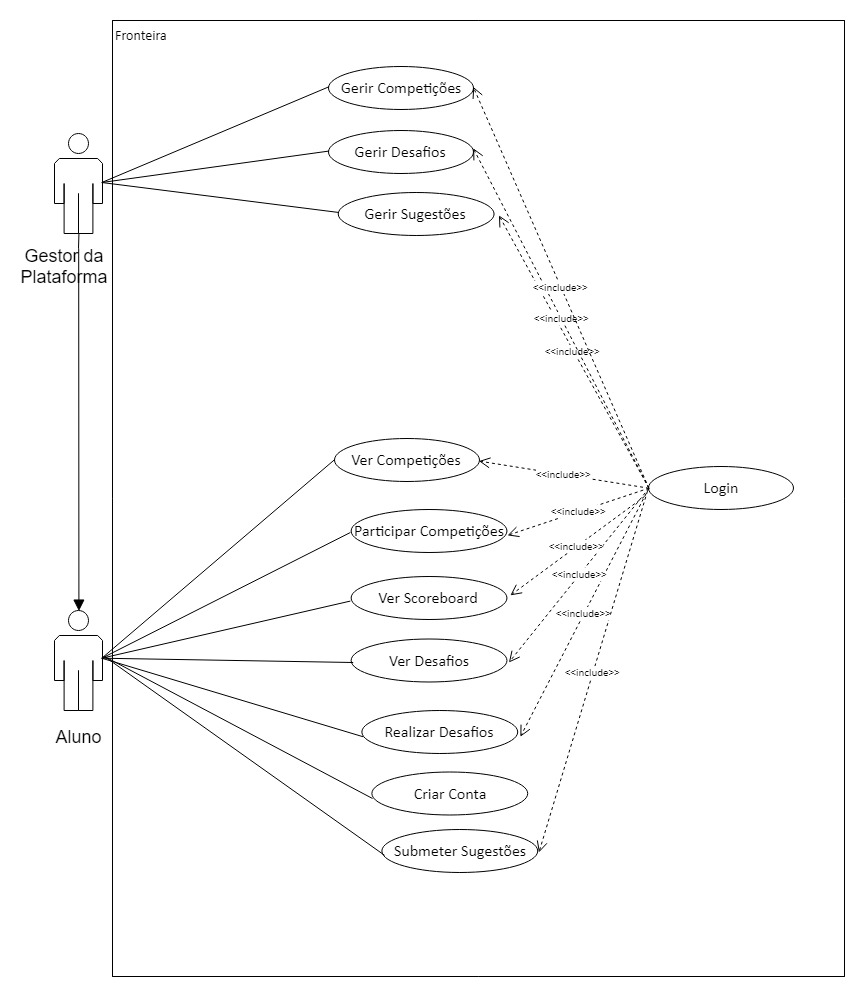
Tabela 2- Atores e respetivos casos de uso

|  |  |
| --- | --- |
| **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 Leaderboard * Ver, Realizar Desafio * Submeter Sugestões |

## Diagrama de casos de uso

Seguidamente 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

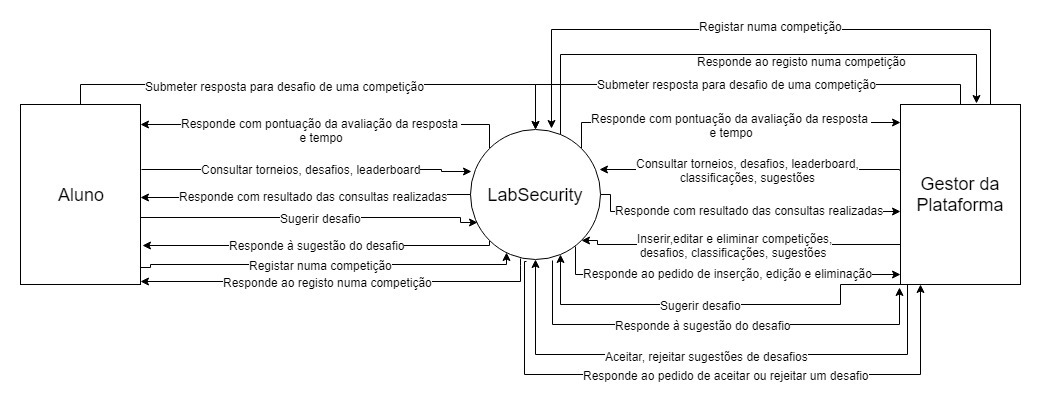


## Diagrama de Contexto

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, 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



## 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.

### Submeter resposta para desafio de uma competição

Na 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 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"

Uma imagem com texto, mapa

Descrição gerada com confiança muito alta

### 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 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"

Uma imagem com captura de ecrã

Descrição gerada com confiança alta

## 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.

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 ID 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 registadas todos os desafios da competição e respetivos scores que um participante recebe na competição caso conclua o desafio.

Na Figura 11 é possível observar o diagrama de classes da aplicação.

Figura 11 - Diagrama de Classes

Uma imagem com texto

Descrição gerada com confiança muito alta

Segue -se o modelo ER, ilustrado na Figura 12 , tendo este sido obtido através de Engenharia Reversa, opção disponibilizada pelo Mysql Workbench.

Figura 12 - Modelo ER

Uma imagem com captura de ecrã

Descrição gerada com confiança muito alta

# 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 ao desenvolvimento da aplicação *web*.

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

## Tecnologias Utilizadas

### JavaScript

Javascript consiste numa linguagem de programação dinâmica e interpretada [7] (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*, sendo estas o *NodeJS* como *backend* e o ReactJS como *frontend*.

### NodeJS

O NodeJS consiste numa *framework* de *JavaScript* designado para *backend*. Este recorre ao V8 engine 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 de *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 [8]. Ao nível da organização de código, tal como outras *frameworks*, o *NodeJS* dá uso a um sistema de módulos. Cada ficheiro é um módulo diferente e de forma a poderem ser usados é necessário existir um *export* dos mesmo no seu ficheiro específico e um *import* no 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, *npm* (*Node Package Manager*), 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 [9].
* Módulos Locais: Módulos criados pelo próprio programador dentro da aplicação.

### ReactJS

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

Como sistema de módulos, o *ReactJS* recorre ao uso de componentes para o desenvolvimento das várias interfaces da aplicação. Estes componentes, para lá 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, podendo ainda conter dados que poderão ser usados na renderização da aplicação [10].

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

De forma a resolver o problema com o *DOM*, o *React* permite ao programador usar um *Virtual DOM* para renderizar, em vez do *DOM.* Este *Virtual DOM* faz com que não seja necessário renderizar 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 [10].

Para entender um pouco mais este processo, apresenta -se na Figura 13 a diferença entre *Virtual DOM e DOM.*

Figura 13 - Virtual DOM vs DOM

Uma imagem com texto, mapa

Descrição gerada com confiança muito alta

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

### HTML

*HTML,* ou *Hypertext Markup Language,* é a linguagem de marcação padrão para páginas *web*. Usa -se para descrever e definir o conteúdo de uma página *web* através de *tags,* que são interpretadas pelo *browser* e mostrados ao utilizadore é normalmente usada em conjunto com *CSS* e *JavaScript.*

### CSS

*CSS*, ou *Cascading Style Sheets,* é um mecanismo que permite ao programador definir estilos para os vários elementos usados em linguagens de marcação, como o *HTML*. 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, fazendo desta forma com que se obtenha uma melhor organização e como resultado um código mais transparente.

### MySQL

O *MySQL* 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*.

### Git

O Git 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*.

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>.

## 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,* 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.

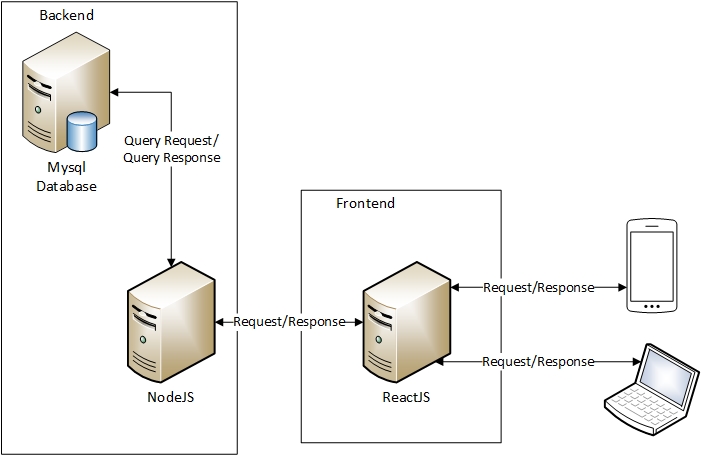
Um dos pontos mais importantes pelo uso desta arquitetura é que a lógica feita no *backend* pode ser consumida para diversas aplicações, desde *web*, a *mobile*, entre outras.

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 a* 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



## Carregamento dos desafios

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 mesmos, houve a necessidade de criar um componente onde os *imports* iriam ser feitos dinamicamente, ou seja, para o caso de cada desafio, em vez de cada um deles possuir uma *Route* própria (exemplo: /desafio1, /desafio2), todos os desafios acedem a uma mesma rota dinâmica com o componente em questão e que tem como principal objetivo conseguir renderizar 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 que é usada para renderizar qualquer componente relativo aos desafios dinamicamente, onde “:name” será o nome do desafio em questão para renderizar.

Figura 15 - Route Dinâmica para os desafios



Dentro do componente dinâmico, de forma a importar o desafio desejado, é usado o campo “link” 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 16 a forma como os desafios estão a ser importados no componente dinâmico.

Figura 16 - Import dinâmico de desafios





## Encerramento de uma competição e criação de um Leaderboard

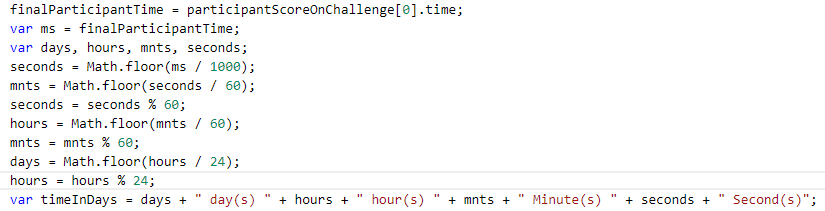
O processo de encerramento de uma competição consiste em detetar quando a data do sistema é maior ou igual à data de fim registada para cada competição. Para fazer essa deteção é usada ao nível do *backend*, no *NodeJS* a função “setInterval()” onde é designada uma função e de quanto em quanto tempo 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* 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 Leaderboard 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 17.

Figura 17 - Conversão de milissegundos para Dias, horas, minutos e segundos



## 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.

# Bibliografia

|  |  |
| --- | --- |
| [1] | “Hacking-Lab,” [Online]. Available: https://www.hacking-lab.com/index.html. [Acedido em 12 Junho 2018]. |
| [2] | “CTF365,” [Online]. Available: https://ctf365.com. [Acedido em 12 Junho 2018]. |
| [3] | “Reversing.kr,” [Online]. Available: http://reversing.kr/. [Acedido em 13 Junho 2018]. |
| [4] | “RingZer0 Team Online CTF,” [Online]. Available: https://ringzer0team.com. [Acedido em 13 Junho 2018]. |
| [5] | “OWASP WebGoat Project,” [Online]. Available: https://www.owasp.org/index.php/Category:OWASP\_WebGoat\_Project. [Acedido em 13 Junho 2018]. |
| [6] | “SCRUM,” [Online]. Available: https://www.scrum.org/resources/what-is-scrum. [Acedido em 17 Junho 2018]. |
| [7] | D.Flanagan, “Introduction to JavaScript,” em *JavaScript : The Definitive Guide*, Agosto 1996, p. 1. |
| [8] | A. Nandaa, “Introduction to Node.js,” em *Beginning API Development with NodeJS*, Julho 2018, p. 1. |
| [9] | A. Nandaa, “Module Categories,” em *Beginning API Development with Node.js*, Julho 2018, p. 1. |
| [10] | P. S. Vipul A M, “What is React?,” em *ReactJS by Example - Building Modern Web Applications with React*, 2016, p. 1. |