UNIVERSIDADE
LUSÓFONA

Trabalho Final de Curso

Yash Jahit - a21705201

Lisboa, Portugal

outubro 2019

Universidade Lusófona de Humanidades e Tecnologias

Licenciatura em Engenharia Informática - Trabalho Final de Curso

Orientador: Bruno Cipriano / Coorientador: Pedro Alves

www.ulusofona.pt

Direitos de cópia

Resumo

A plataforma Drop Project permitiu simplificar o processo de submissão de projetos, fichas e trabalhos de casa efetuados pelos alunos e a posterior validação dos mesmos feita pelos professores. Mas isto também introduziu ansiedade aos alunos, pois o processo de submissão exigia a realização de alguns passos para que a submissão fosse válida na plataforma antes de correr qualquer teste unitário. Daqui surgiu a ideia da criação e implementação de um plugin acessível através do *marketplace* da ferramenta IntelliJ, este plugin permitirá maior facilidade aos alunos, pois facilitará a submissão do projeto, tudo através da interface do IntelliJ. O projeto é *open-source* e está disponível no Github[1].

Palavras-chave: Drop Project, *Plugin*, Kotlin, Java, IntelliJ, Submissão de Projetos, Visualização de relatórios, Testes Unitários, *Automatic Assessment Tools.*

[1] https://github.com/yashjahit-21705201/Plugin-Drop-Project/

Abstract

The Drop Project platform made it possible to simplify the process of submitting projects, worksheets and homework by students and their subsequent validation by teachers. But this also introduced anxiety for students, as the submission process required some steps to be taken to make the submission valid on the platform before running any unit tests. From here came the idea of creating and implementing a plugin accessible through IntelliJ's marketplace, this plugin will make it easier for students, as it will facilitate project submission, all through the IntelliJ interface. The project is *open-source* and is available on Github[2].

Keywords: Drop Project, plugin, Kotlin, Java, IntelliJ, Submissão de Projetos, Visualização de relatórios, Testes Unitários, *Automatic Assessment Tools.*

[2] https://github.com/yashjahit-21705201/Plugin-Drop-Project/

Índice

Índice de Figuras

Índice de Tabelas

1. Identificação do problema

O Drop Project é uma plataforma que serve para apoiar aos alunos na submissão de projetos, fichas e defesas finais dos projetos, nas cadeiras de Fundamentos de Programação, Algoritmia e Estrutura de Dados e Linguagens de Programação II. Foi criada com recurso ao *Spring Framework* e foi usada a linguagem de programação *Kotlin*. Foi desenvolvida pelo Professor Pedro Alves na Universidade Lusófona. A plataforma permite aos alunos submeterem projetos (respeitando um conjunto de requisitos) e obterem um feedback sobre os erros existentes no programa, a má escrita de código (código não está de acordo com as convenções de escrita em certas linguagens de programação) e permite acesso ao histórico de submissões e erros que tinham. A plataforma não só tem funcionalidades para os alunos como inclui também um conjunto de ferramentas que permite facilitar aos professores no momento da avaliação das submissões como também filtrar alunos com acesso a plataforma, comparar todas as submissões efetuadas cada uma com o *time stamp* do último *commit* ou upload, download dos ficheiros ZIP que são submetidos pelos alunos, entre outros. Esta plataforma trouxe muitas vantagens aos alunos e aos professores como por exemplo evitar que os professores tenham de estar a receber imensos projetos e estar constantemente a configurar os projetos para os testar ou até permitir aos alunos terem acesso a submissões efetuadas e a evolução de erros no projeto, mas isto também trouxe um sentimento de ansiedade aos alunos.

Tal como referido anteriormente, cada submissão precisa de cumprir com alguns requisitos de modo a que a submissão não seja rejeitada: estes podem variar desde a criação de um ficheiro de texto com nomenclatura correta e corpo de ficheiro com conteúdo válido, como podem obrigar a criação de packages com nomenclatura predefinida. Isto trouxe alguns problemas de ansiedade aos alunos, mas não só, por exemplo no ato de mini fichas ou defesas de projeto, os alunos devem de estar constantemente a criar ficheiros ZIP para a sua posterior transferência a plataforma, isto consome tempo precioso aos alunos e aumenta o nível de ansiedade dos mesmos. Neste momento o processo de submissão e validação de projetos consiste nos seguintes passos:

- É efetuado o *login* na plataforma, através de uma autenticação federada da Universidade, usando para tal as credenciais de aluno, professor ou de administrador.

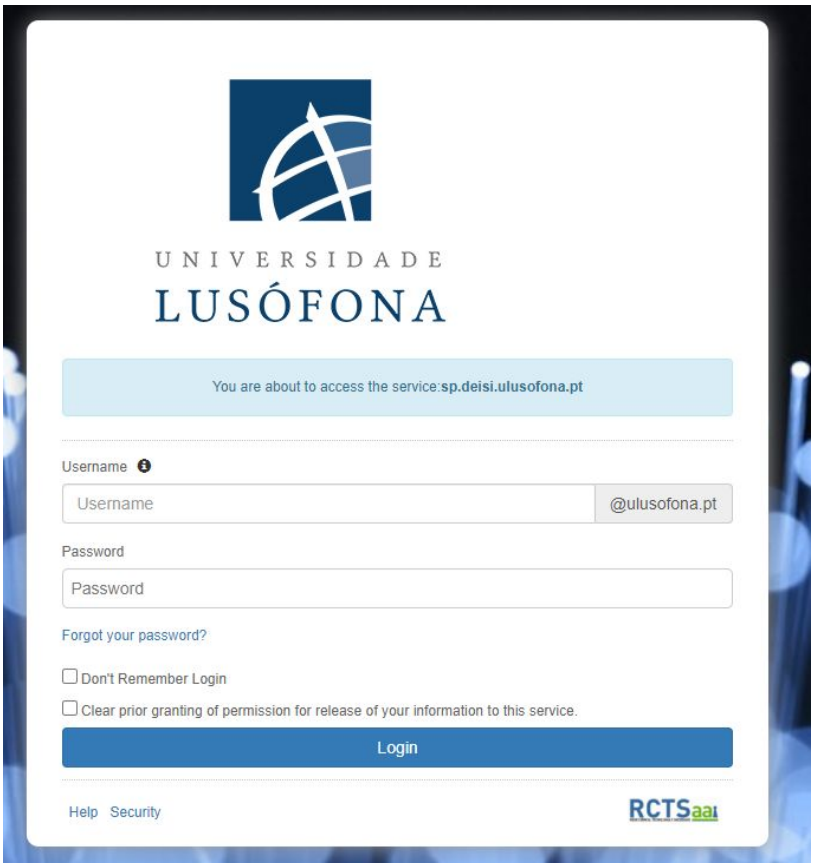


Figura 1. Página de *login* da plataforma

- Caso as credenciais fornecidas sejam de aluno, é apresentada uma página em que o aluno consegue escolher o *assignment* desejado (o aluno pode chegar diretamente à página da submissão a partir de um link fornecido pelos professores). Após ter escolhido o assignment desejado, é redirecionado para a página onde pode fazer a submissão do ficheiro ZIP, nesta página também é descrita uma breve descrição do *assignment*, instruções na implementação do código e a estrutura necessária para garantir que o projeto consiga ser testado. No caso de projetos da cadeira, as instruções da estrutura do

projeto, funções e classes obrigatórias (p.ex: “fun lerFicheiro(), class Main()”) estão descritas no enunciado do projeto.

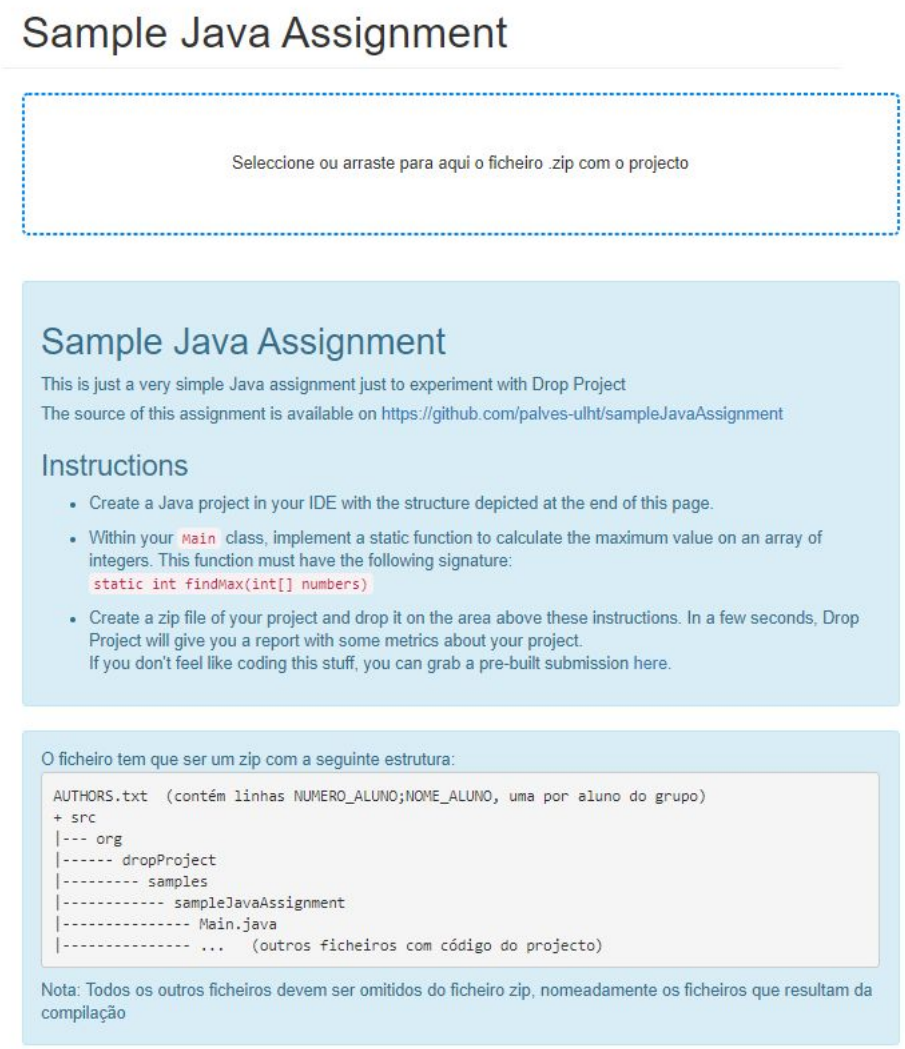


Figura 2. *Landing page* depois de fazer o *login* e escolher *assignment*

- Após o aluno ter submetido o seu projeto, o Drop Project mostra um relatório com todos os testes de professor que passaram com sucesso e os que não passaram, neste caso os alunos têm acesso a stack do programa, de modo a permitir perceber que teste devolveu o resultado errado.

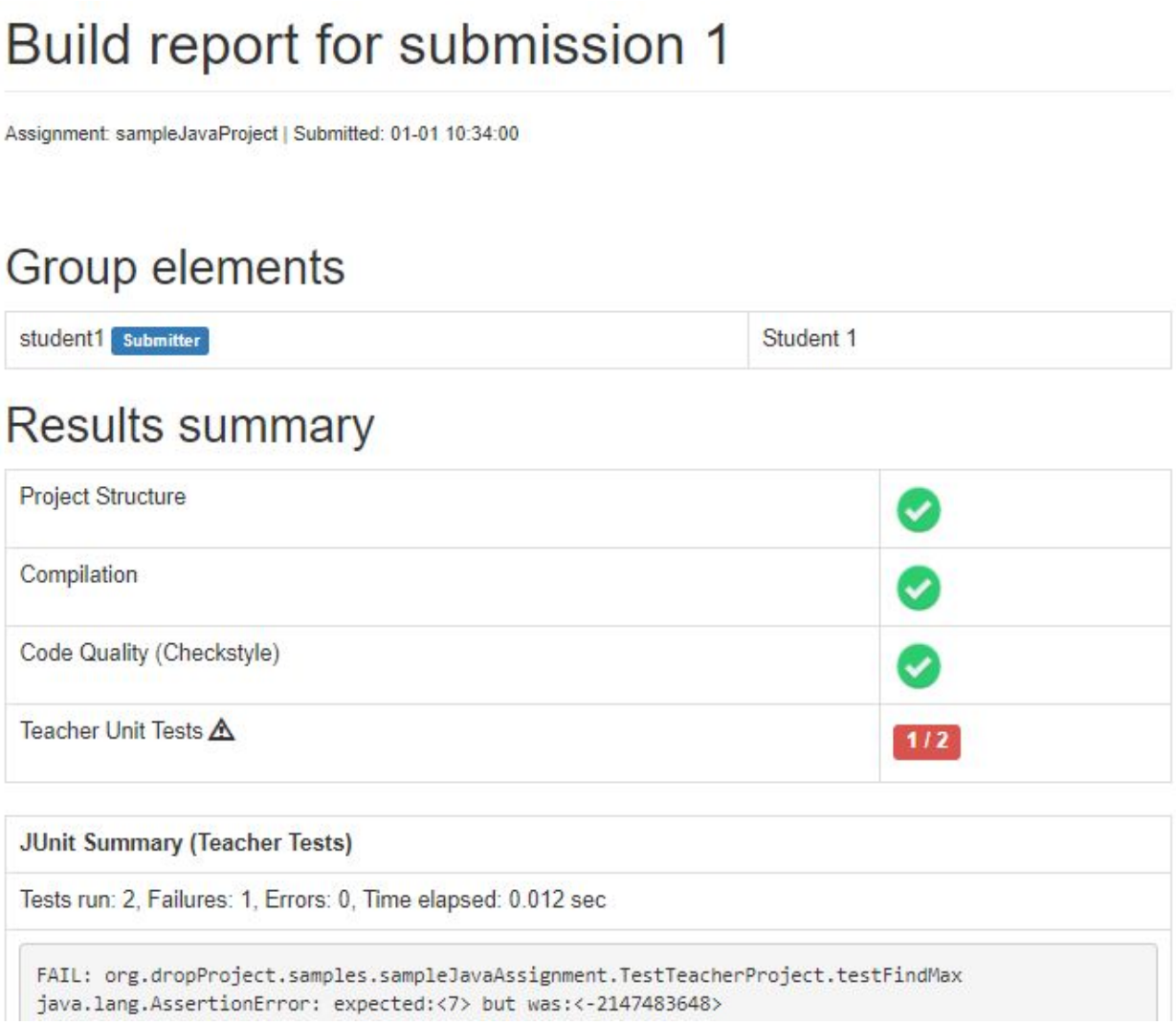


Figura 3. Ecrã com resultado dos testes

Estes passos todos consomem tempo aos alunos que poderiam ser gastos a programar ou a fazer mais pesquisa sobre o exercicio. O objetivo do plugin é permitir a diminuição de passos necessários para que uma submissão seja feita na plataforma e acreditamos que a automatização deste processo irá permitir reduzir a ansiedade dos alunos, não obstante, existem passos que são essenciais para o correto funcionamento da plataforma, como por exemplo a criação de funções obrigatórias.

O plugin a ser desenvolvido será acessível através do *marketplace* da IntelliJ e poderá ser usado em qualquer versão superior a 2020.1 do IDE (Community Edition/Ultimate)

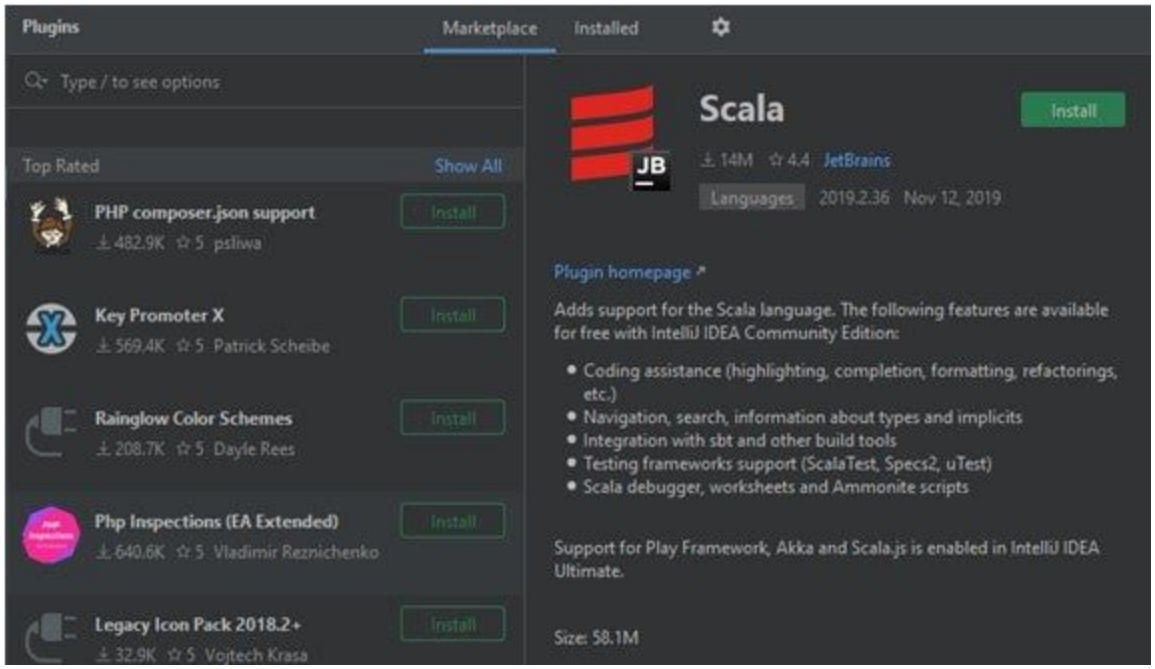


Figura 4. *Marketplace* acessível através do IntelliJ

2. Viabilidade e Pertinência

O projeto foi realizado com a ajuda da ferramenta IntelliJ e a documentação disponível para o desenvolvimento de plugins [1], será usada a linguagem de programação Kotlin por isso o projeto não terá custos associados e o código-fonte do projeto será *open-source* com licença Apache 2.0, sem custos nenhum exceto para seu uso em fins comerciais.

Para ser possível avaliar a pertinência do projeto, foi realizado um questionário aos alunos do 1º Ciclo de Engenharia Informática, Informática de Gestão e Engenharia Informática de Redes e Telecomunicações da Universidade Lusófona de Humanidades e Tecnologia e obteve-se um número total de 31 respostas, sendo a maioria dos alunos do 2º ano.

Figura 5. Gráfico de distribuição de resposta por ano

Também foi necessário perceber em que disciplinas se fez uso do Drop Project, pois desta forma é mais fácil perceber se com o passar de ano existe uma redução da ansiedade na plataforma.

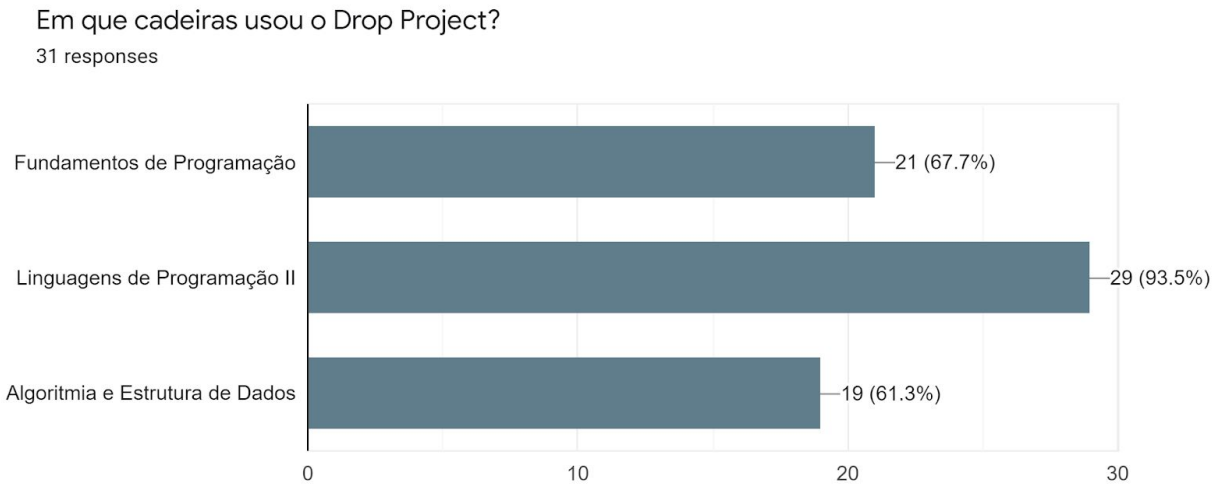


Figura 6. Uso do Drop Project nas cadeiras do curso

O Drop Project não serve somente para a submissão de projetos, como também para fichas, trabalhos de casa e defesa do projeto semestral. Com todas estas aplicações é necessário perceber qual é a que a plataforma permite ter uma experiência mais positiva.

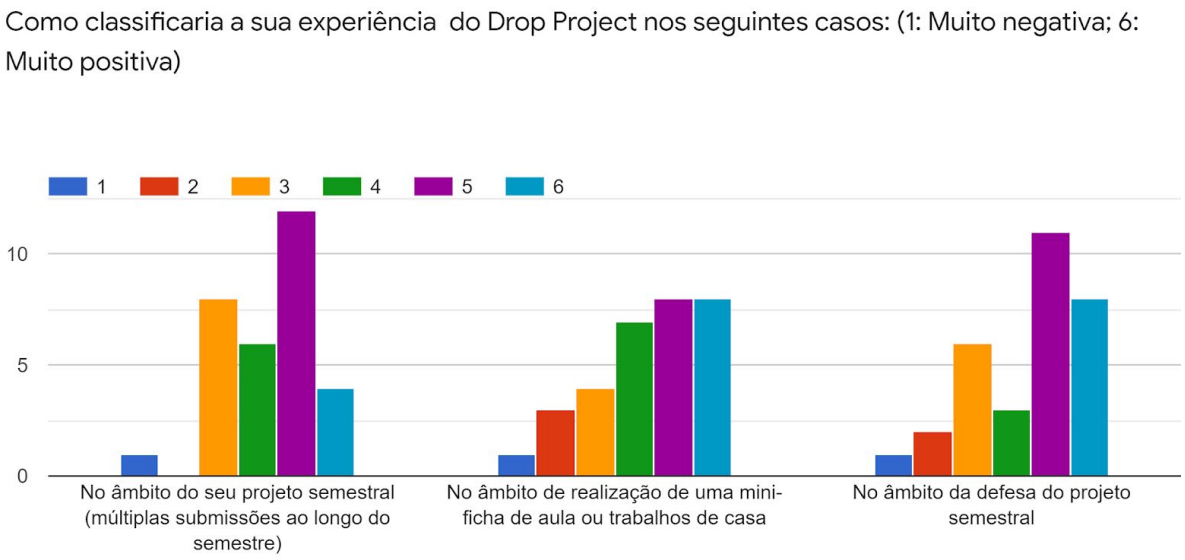


Figura 7. Satisfação dos alunos em usar o DP nas diversas aplicações

Com o gráfico da figura 7, é possível perceber que os os resultados quer da mini-ficha, quer da defesa, são um pouco melhores do que os resultados do projecto, mas a diferença não parece ser significativa.

O DP requer o cumprimento de alguns passos, estes passos causam ansiedade em alguns alunos, por isso foram realizadas as seguintes perguntas de modo a permitir perceber quais é que aumentam os níveis de ansiedade aos alunos:

- Criar o ficheiro AUTHORS.txt
- Criar o ZIP para o upload posterior na plataforma
- Obrigatoriedade de ter um package pré-definido
- Necessidade de cumprir com protocolos obrigatórios (classes obrigatórias, funções com tipo de retorno obrigatório)
- Resultado dos testes unitários

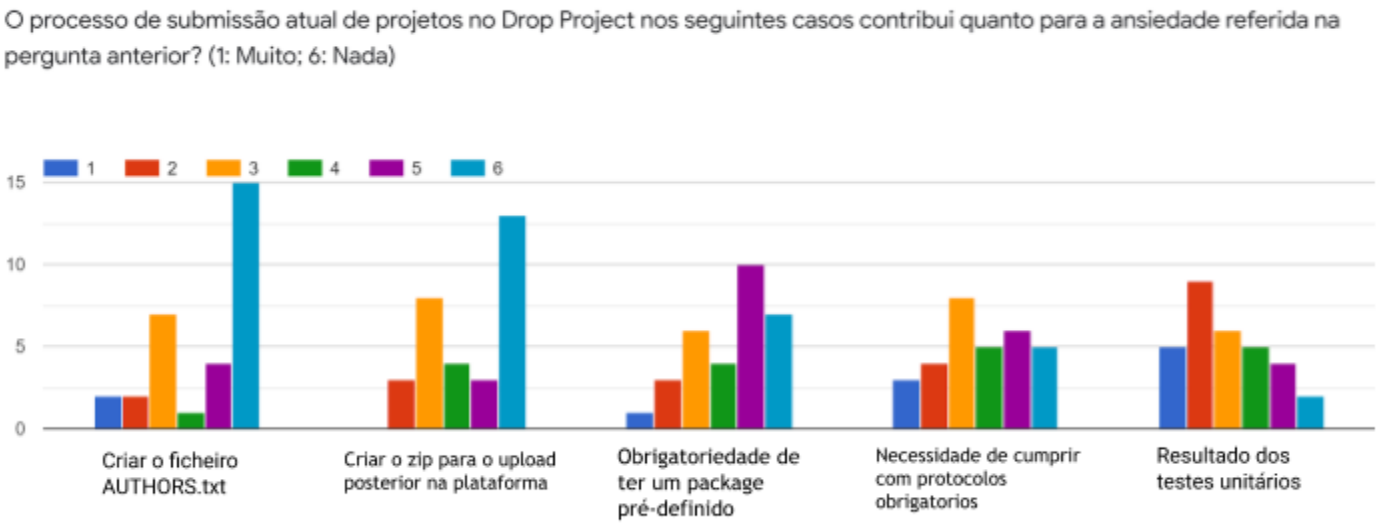


Figura 8. Ansiedade sentida em diversos casos

Após a análise das respostas, percebeu-se que um dos pontos que mais ansiedade trazem é no resultado dos testes unitários, e verificou-se também algum tipo de ansiedade na criação do ficheiro AUTHORS.txt e na criação do ficheiro ZIP.

Finalmente, foi feita uma pergunta de modo a perceber se a implementação de um plugin seria uma ideia que os alunos concordam com ou não, sendo que o plugin tem como objetivo facilitar o trabalho dos alunos e diminuir o tempo que levam a efetuar submissões. Com as respostas obtidas, a ideia da implementação e criação deste plugin foi reforçada pois mostra que os alunos estão abertos a um sistema de submissões mais simplificado.

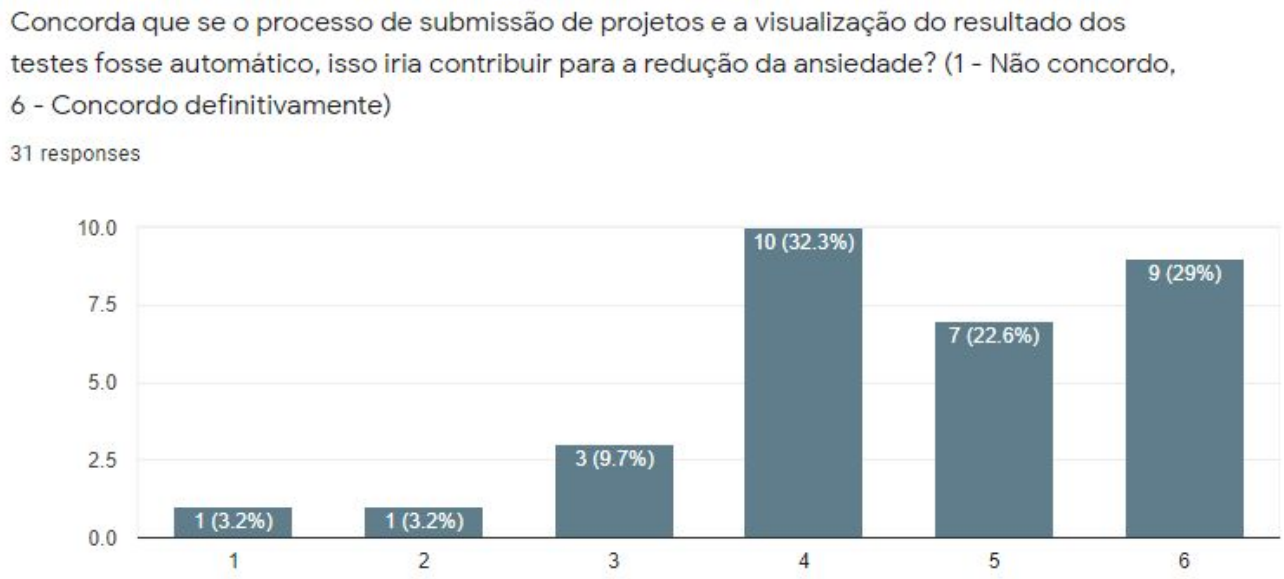


Figura 9. A ideia da implementação do plugin é suportada

3. Levantamento e análise dos requisitos

Antes de se iniciar o desenvolvimento do projeto foi feito um levantamento de requisitos funcionais e não-funcionais de modo a obter uma ideia geral dos objetivos gerais que o plugin deve cumprir.

Os requisitos funcionais são declarações de funções de como o sistema deve reagir a entradas específicas e como deve comportar em determinadas situações. É uma interação entre o sistema e o seu ambiente. [2]

Os requisitos não-funcionais podem ser designados como requisitos relacionados ao uso da aplicação em termos de desempenho, usabilidade, confiabilidade, segurança, disponibilidade, manutenção e tecnologias envolvidas. De um certo modo, descrevem restrições ao sistema que limitam às possibilidades de implementação. [3]

3.1 Requisitos Funcionais

Requisito Funcional 1 (RF1) - *Login* na plataforma Drop Project

- O plugin deve mostrar um diálogo de *login* (que estará acessível através de uma opção de "*Login*" na barra de ferramentas do IntellIJ), de modo a permitir o acesso na plataforma do Drop Project.

Requisito Funcional 2 (RF2) - Listagem de Submissões

- Após a realização do [RF1] o utilizador deve ter opção para escolher o *assignment* que pretende para submeter o projeto.
- Deve ser possível ter acesso ao enunciado do *assignment.*

Requisito Funcional 3 (RF3) - Submeter Projeto

- O plugin deve criar um ZIP com a estrutura necessária para submeter o projecto na plataforma[3].

[3] A estrutura consiste na pasta "*src*" e no ficheiro *"AUTHORS.txt"*

- Quando o utilizador escolher a opção de submeter, o plugin deve fazer fazer upload do ZIP criado previamente.

Requisito Funcional 4 (RF4) - Criação de um ficheiro AUTHORS.txt

- O plugin deve criar o ficheiro AUTHORS.txt usando as informações que o aluno forneceu (nome e número de aluno) quando realiza o *login* [RF1].
- (RF4.1) Deve ser validado o formato do número do aluno[4].
- O ficheiro deve conter os dados dos alunos no formato válido pela plataforma[5].

Requisito Funcional 5 (RF5) - Resultados dos testes

- O plugin deve ser capaz de mostrar os resultados sem ter de sair do IDE ou caso contrário deve fornecer aos alunos o link dos resultados para verem os resultados dos testes.

Requisito Funcional 6 (RF6) - Aviso de *login* efetuado

- O plugin deve ter um sistema que seja capaz de avisar ao utilizador que já tem o *login* efetuado sempre que o tentar efetuar novamente.

Requisito Funcional 7 (RF7) - *Login* Automático

- O plugin deve fazer *login* automático de modo a evitar que o utilizador o tenha de estar a fazer constantemente.

3.2 Requisitos Não Funcionais

➔ Requisito Não Funcional 1 (RNF1) - O plugin deve realizar a operação de login em modo assíncrono na abertura do IntelliJ de modo a evitar demora a carregar o projeto.

➔ Requisito Não Funcional 1 (RNF2) - O plugin deve estar disponível no *marketplace* do IntellIJ.

[4] Número de aluno deve conter 'a' e ter comprimento 9
[5] Formato deve ser (NUMERO_ALUNO;NOME_ALUNO)

4. Solução Desenvolvida

4.1 Enquadramento Técnico

O plugin, tal como tinha sido referido anteriormente, fará uso da linguagem de programação Kotlin e será usado o IDE IntelliJ para o seu desenvolvimento.

O IntelliJ é um IDE (Integrated Development Environment) escrito com recurso a linguagem Java e Kotlin usado para o desenvolvimento de aplicações software. É desenvolvido pela Jetbrains e a primeira versão foi lançada em janeiro de 2001 [4]. O IntelliJ usa técnicas que automatizam algumas tarefas (por exemplo a importação automática de dependências) e sugere ações que podem facilitar o trabalho do programador como por exemplo sugerir nomes de classes, métodos, *keywords*, campos. Com isso, oferece integração imediata com *frameworks* populares usadas pelos programadores Java. Essa integração nativa reduz drasticamente a existência de bugs e fornece grande sinergia entre as diferentes tecnologias usadas para construir qualquer projeto. Os produtos criados na plataforma IntelliJ são aplicações extensíveis, sendo a plataforma o responsável pela criação de componentes e pela injeção de dependências nas classes. Esta plataforma suporta totalmente plugins, e a JetBrains hospeda um repositório de plugins que pode ser usado para distribuir plugins que suportam um ou mais produtos. Também é possível hospedar repositórios pessoais e distribuir plugins separadamente. Muitas das funcionalidades existentes na plataforma IntelliJ são escritas como plugins que podem ser incluídos ou excluídos, dependendo das necessidades do produto final. Os plugins podem estender a plataforma de várias maneiras, desde a adição de um simples item no menu até permitir o suporte a uma linguagem completa, sistema de compilação e *debugger*, tipicamente caem numa das 4 seguintes categorias [5]:

- Suporte de um *Custom Language* - Este suporte fornece funcionalidade básica para trabalhar com uma linguagem de programação específica;
- Integração de *framework* - A integração consiste em recursos aprimorados de insight de código, típicos de uma determinada estrutura, bem como na opção de usar a

funcionalidade específica da estrutura diretamente do IDE. Às vezes, também inclui elementos de suporte ao idioma para uma sintaxe personalizada ou DSL;

- Integração de ferramentas - Torna possível manipular ferramentas e componentes de terceiros diretamente do IDE sem alternar contextos;
- Complementos do UI - Os plugins nesta categoria aplicam várias alterações à interface do utilizador padrão do IDE. Algumas componentes recentes são interativas e fornecem novas funcionalidades, enquanto que outros são limitados apenas a modificações visuais.

Kotlin é uma linguagem de programação multiplataforma que compila para a Java Virtual Machine (JVM), foi desenvolvida pela Jetbrains e a primeira versão oficial foi lançada em 2016 [6].

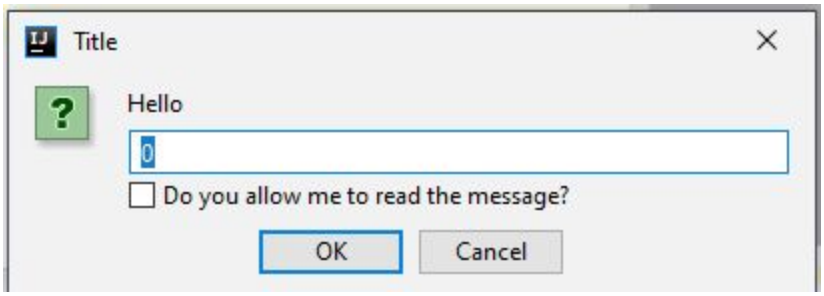


Figura 10. Diálogo de input

Com a ajuda de uma vasta biblioteca de frameworks da linguagem Java, podemos usar o Java Swing ou JavaFX para implementar diálogos, mostrar mensagens de erros, receber input do utilizador, etc. De modo a começar perceber melhor o funcionamento das bibliotecas e de como podemos obter os dados inseridos pelo utilizador foram criados diálogos que realizavam ações básicas, como por exemplo sucede na figura 10, o utilizador insere qualquer tipo de mensagem e simplesmente têm de escolher se permite que o plugin faça a leitura da mensagem escrita ou não, caso a permissão não tenha sido dada então é mostrada uma mensagem de erro ao utilizador para o alertar da situação. As funcionalidades não se limitam somente a diálogos e mensagens de erro, existe muitas outras funcionalidades da parte de código que podem ser implementadas pelo programador (isto já fora das bibliotecas Swing e JavaFX) como obter o *path* da pasta a ser trabalhada, mostrar notificações, pop-ups, leitura e escrita de ficheiros, entre outros.

4.2 Plugin Drop Project

Para poder ter acesso às funcionalidades descritas nos requisitos, foram usadas e criadas APIs no Drop Project [Anexo C]. Estas APIs tiveram de ser implementadas de modo a poder ter acesso a funcionalidades que originalmente não existiam.

Para poder aceder ao plugin, o utilizador terá, após a instalação do plugin, uma opção "Drop Project" no menu de ferramentas do IntelliJ, com diversas opções para selecionar como podemos observar na Figura 11.

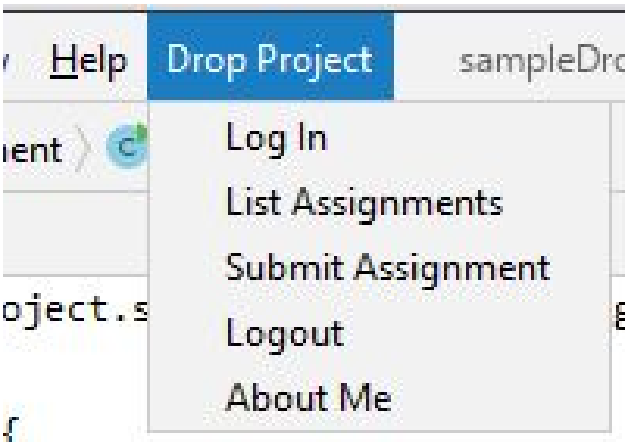
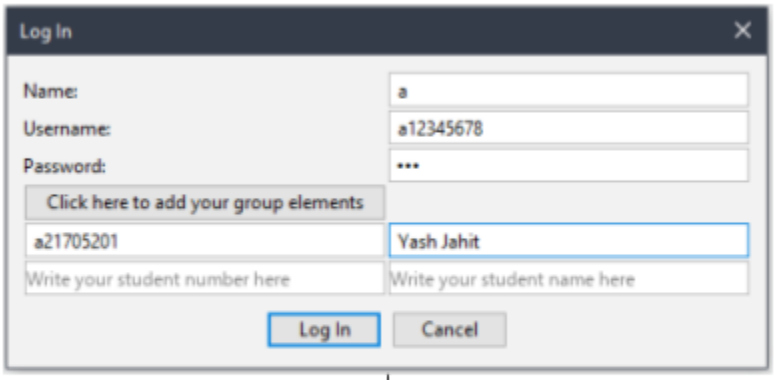


Figura 11. Opções do plugin

Segundo um dos requisitos funcionais especificados na secção 3.1 o plugin deve ter um diálogo de *login* que permita os utilizadores aceder à plataforma de Drop Project (RF1). Como podemos observar o fluxo do sistema de *login* na Figura 12, o diálogo de *login* (Figura 12.1) pede 3 informações principais (Nome, *username* e password do aluno), e ainda temos a opção de inserir o número e nome de mais dois colegas de grupo (estes dados serão adicionados no ficheiro AUTHORS.txt). Caso as credenciais do aluno forem válidas o plugin irá mostrar uma mensagem de sucesso (Figura 12.3) caso contrário será mostrado ao aluno o diálogo de credencial inválido (Figura 12.2).

Figura 12.1 Painel de login



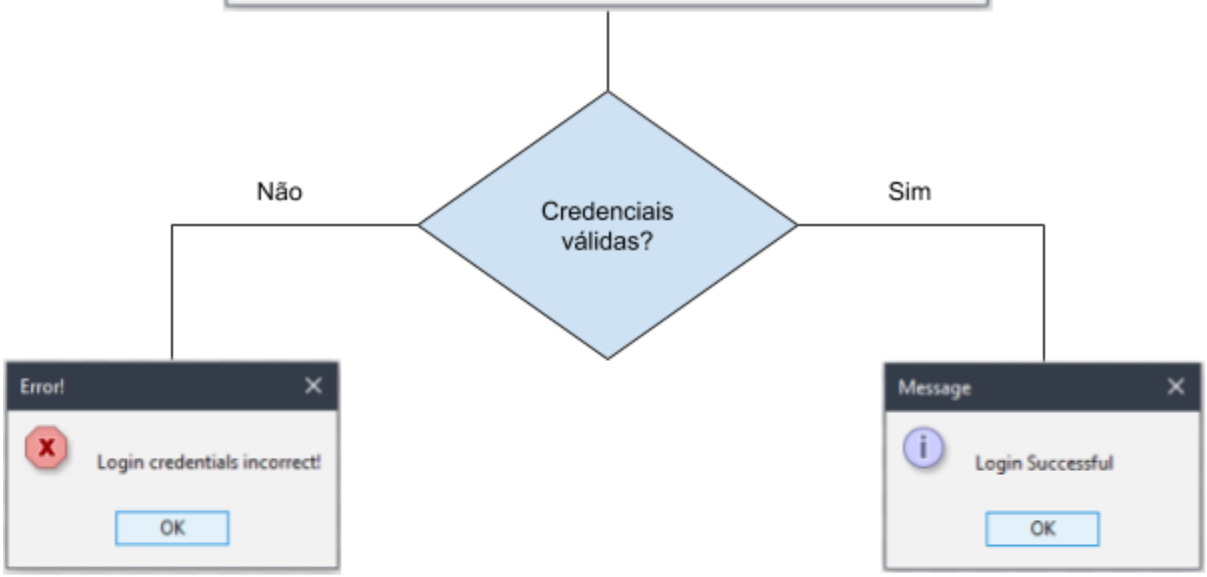


Figura 12.2 Credenciais inválidas

Figura 12.3 Login efetuado com sucesso

Figura 12. Fluxo do sistema de login

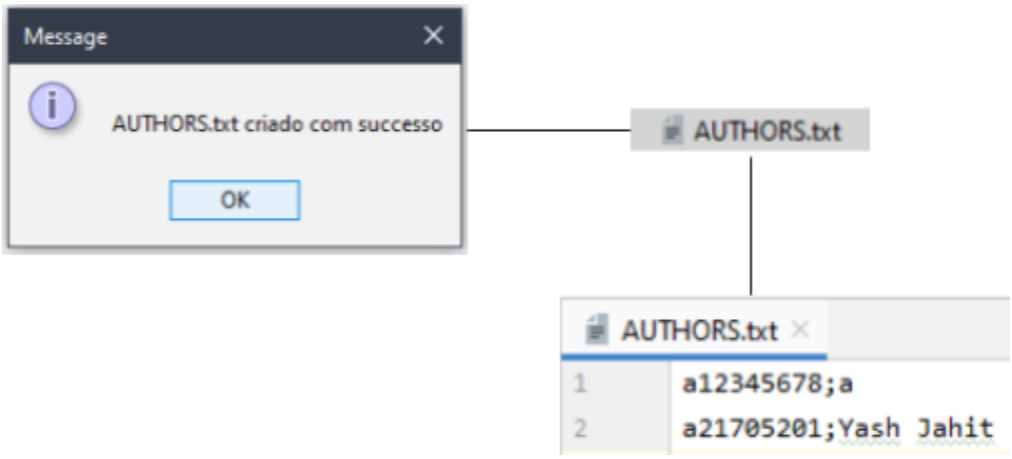


Figura 13. Criação do AUTHORS.txt

Após o aluno realizar o *login*, o plugin cria o ficheiro de AUTHORS.txt com os seus dados e dos restantes colegas do grupo caso se aplique (Figura 13) (RF4). O plugin também guarda as credenciais do aluno num ficheiro "*up.txt"*e aplica um simples algoritmo para cifrar a password do utilizador [Anexo A] de modo a que mesmo que alguém tenha acesso ao ficheiro com a password, terá de aplicar o algoritmo inverso [Anexo B] para obter a password original de volta. Pelo que se pode observar a password que está visível na Figura 14 é a password "cifrada" que foi inserida na Figura 12.3. Com o armazenamento de credenciais, o plugin consegue efetuar o *login* automático no *startup* do IDE dispensando deste modo que os alunos o tenham de estar sempre a fazer quando abrem o IDE.

Figura 14. Armazenamento da credencial

Este sistema não é totalmente seguro, pois permite a qualquer utilizador com conhecimento do algoritmo implementar o algoritmo inverso e obter a password do aluno. No entanto o ficheiro *up.txt,* não será submetido a plataforma de Drop Project, mantendo se localmente na máquina.

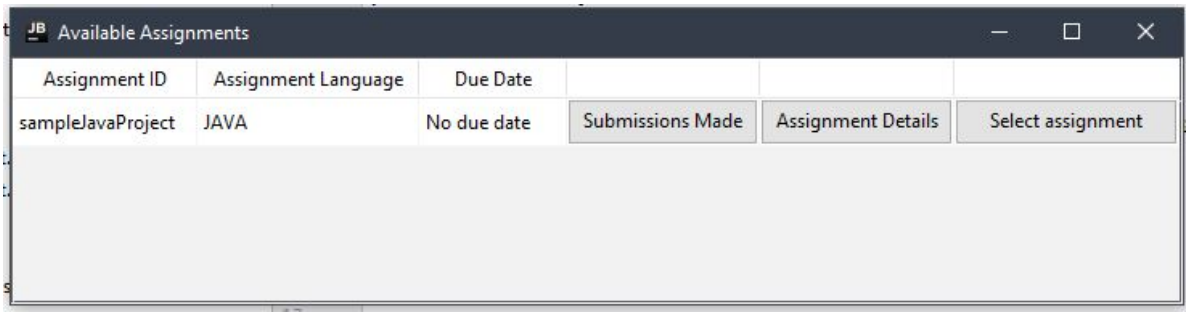


Figura 15. Diálogo de *assignment* disponíveis

De acordo com o RF2, o plugin tem a opção de visualizar todos os *assignments* que o aluno pode fazer, verificar a lista de submissões efetuadas para um determinado *assignment* e escolher o *assignment* para qual vai fazer a submissão, tudo através de um diálogo como se pode observar na Figura 15, este diálogo também consegue mostrar mais alguma informação como a linguagem de programação a ser usada no *assignment* e caso exista, a data limite de entrega.

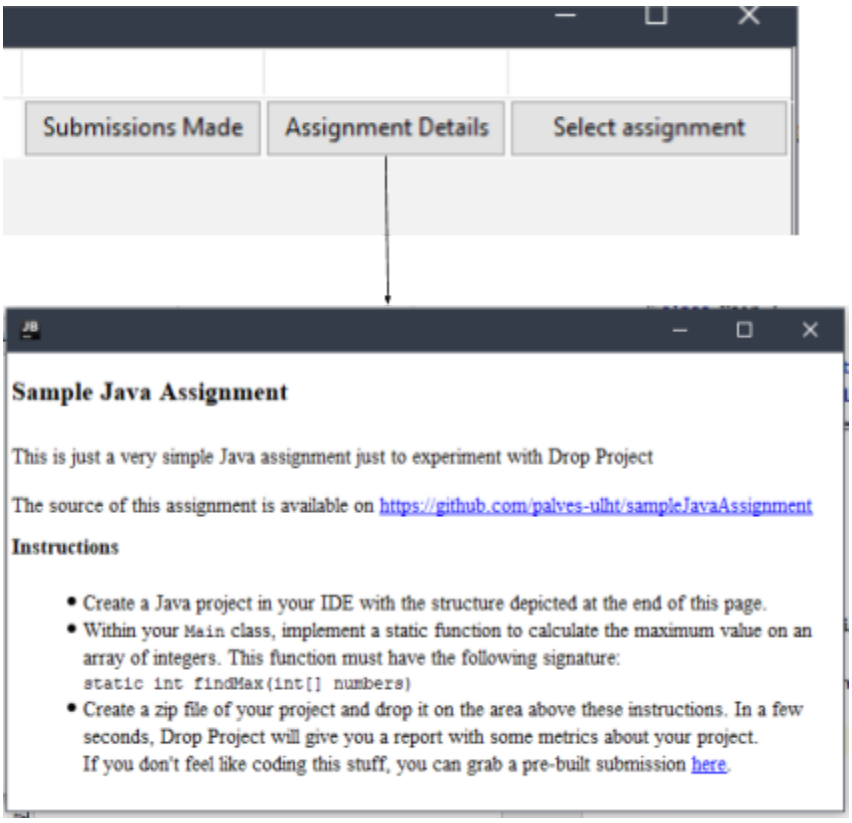


Figura 16. Enunciado de um assignment

O aluno têm a possibilidade de ver o enunciado de um *assignment* em qualquer momento desde que tenha o *login* efetuado como é possível ver na Figura 16.

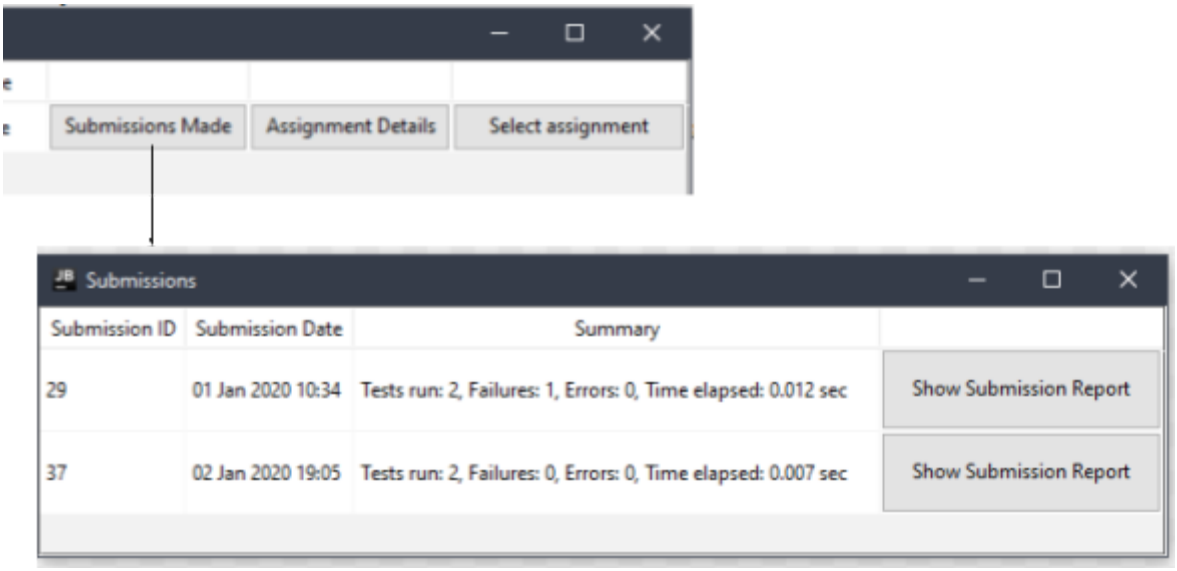


Figura 17. Diálogo de submissões efetuadas

No diálogo de submissões, o aluno consegue encontrar mais algumas informações em relação às submissões que efetuou, como a data de submissão. Consegue ver também um resumo do resultado dos testes que foram realizados pela plataforma do Drop Project, contendo o número de testes, quando falharam, quantos erros houve e o tempo total.

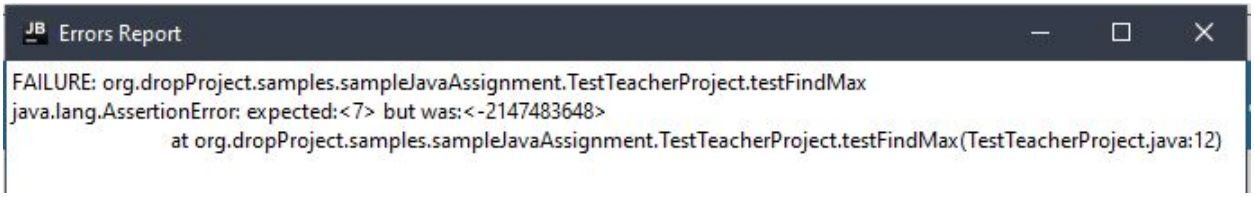


Figura 18. Relatório de erros

Também é possível ter acesso ao relatório devolvido pelo Drop Project no caso de encontrar erros ou falhas no projeto (Figura 18)

Figura 19. *Widget* no *status bar*

O plugin dispõe de um *widget* no canto inferior direito do *status bar* onde o aluno consegue identificar rapidamente o *assignment* que esteja selecionado no momento. Caso não tenha sido selecionado nenhum, é mostrada uma mensagem de acordo com o representado na Figura 19.

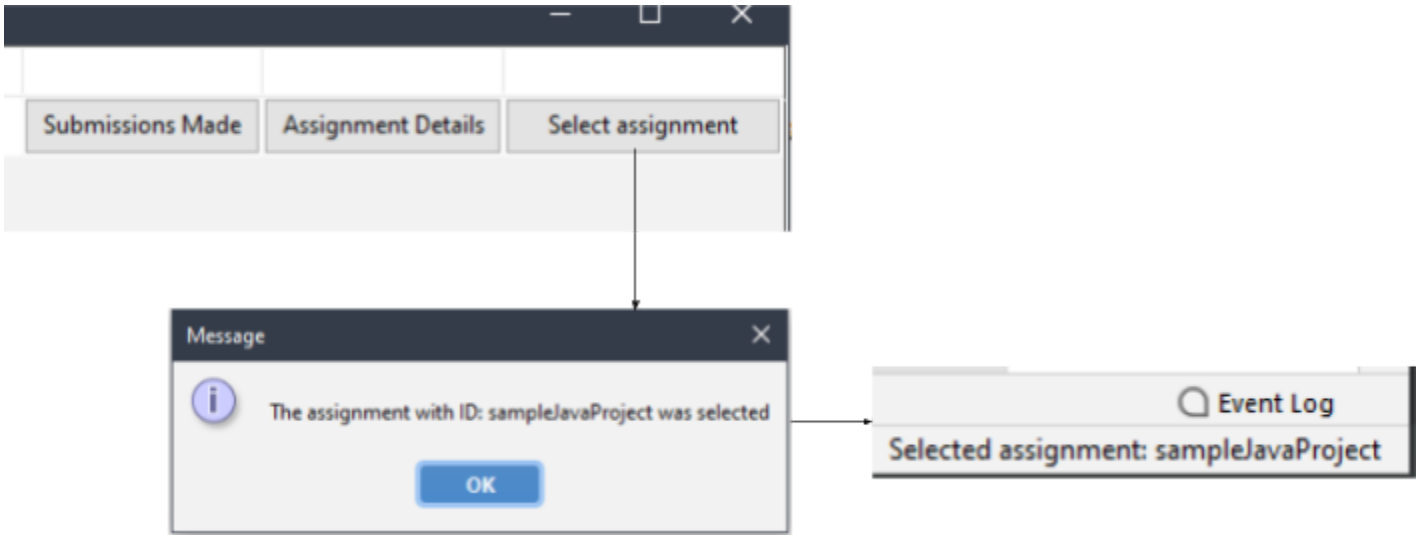


Figura 20. Atualização da informação no *widget*

No caso de o aluno selecionar um *assignment*, vai ser mostrada um diálogo a confirmar essa seleção e proceder-se-á à atualizar a informação no *status bar*.

4.3 Drop Project

O trabalho não se limitou a realizar somente no plugin. Devido a necessidade de obter informações como a lista de assignments ou o relatório dos testes, foi necessário implementar uma API no Drop Project que pudesse devolver essas informações em formato JSON[6], que é uma formato que permite a troca de dados simples e rápida entre sistemas.

Primeiramente foi necessário efetuar uma lista de API que iriam ser usadas pelo plugin [Anexo C]. O primeiro a ser usado foi o serviço de *login* [Anexo C.1] (já existente na plataforma), este serviço recebe as credenciais dos alunos em formato `base64`[7] num pedido POST, caso as credenciais não forem válidas é devolvida uma mensagem de erro, caso contrário o utilizador fica autenticado.

A próxima API desenvolvida obtém a lista dos *assignments* [Anexo C.2] do utilizador autenticado, esta lista é devolvida em formato JSON com informações como o ID do *assignment,* a linguagem a ser usada (Java ou Kotlin), a data limite para submissão e o enunciado em formato HTML.

De modo a permitir a submissão dos ficheiro ZIP [Anexo C.3], foi usada uma serviço já existente que recebem como parâmetros o ficheiro e o ID para a qual a submissão está a ser feita e caso este for aceite o Drop Project devolve o `submissionId`.

O plugin permite ao aluno visualizar todas as submissões que efetuou para um determinado *assignment* e para cada submissão, ver o respetivo relatório de erros (caso exista) [Anexo C.4-5].

Em todos os pedidos efetuados devem ser enviado às credenciais do aluno de modo a que seja possível saber se o determinado utilizador tem permissão para aceder às funcionalidades. Em cada pedido efetuado é possível ocorrer algum tipo de erro e por esta razão o servidor tem códigos de erro que possibilitam facilmente descobrir a causa do erro.

[6] https://pt.wikipedia.org/wiki/JSON
[7] https://pt.wikipedia.org/wiki/Base64

5. *Benchmarking*

No mercado existem algumas ferramentas semelhantes ao Drop Project, estas ferramentas também permitem a submissão de exercícios de modo a aprender e treinar linguagens de programação (chamados de *online judge*). No entanto, algumas das ferramentas só permitem submissões efetuadas pela *interface* web, não existindo nenhum plugin para IDEs ou um programa que facilite a submissão.

5.1 Codechef

Uma dessas ferramentas é a CodeChef [7] , uma iniciativa educacional sem fins lucrativos da Directi. É uma comunidade de programação global que promove aprendizagem e competição amigável entre diversos utilizadores. O objetivo da plataforma é ajudar os programadores a terem maior conhecimento na área de algoritmos, programação e permitir treinarem para concursos de programação.

A plataforma consiste em diferentes tipos de atividades que um utilizador pode aceder:

- Tem uma área de prática, com diversas dificuldades variando de principiante a experiente.
- Área com acesso a competições mensais;
- E finalmente uma área onde o utilizador pode aceder às redes sociais da página, aos fóruns para discussão e ao blog do CodeChef.

Ao contrário do Drop Project, que está focado na linguagem Java (e em Kotlin) o CodeChef permite a submissão de soluções em vários tipos de linguagens, nomeadamente Java, C, C++, Python, C# e entre outros. Outra funcionalidade que existe é a capacidade dos utilizadores escreverem código diretamente no browser em vez de escrever o código localmente na máquina, na figura seguinte podemos ver um simples programa escrito em Java que devolve a famosa frase “Hello World” muito usada quando se dá início a aprendizagem de uma nova linguagem.

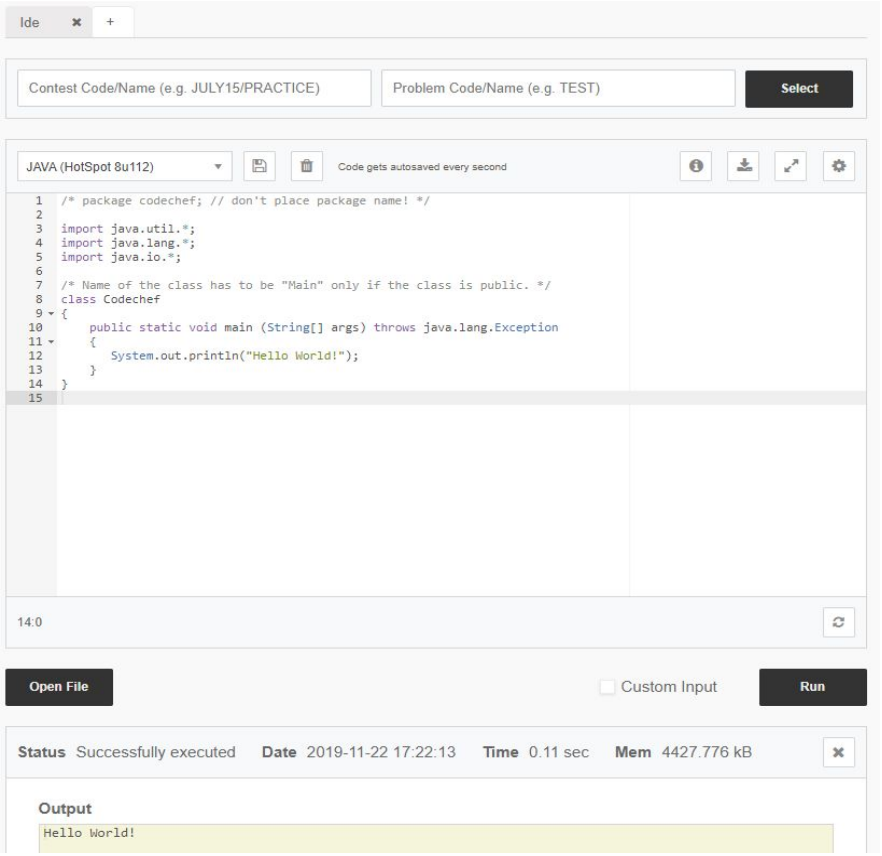


Figura 21. “Hello World” escrito no IDE do Codechef

O Codechef não dispõe de um plugin para o IntelliJ, por isso a única forma para submissão de ficheiro do código-fonte tem de ser feita upload manualmente.

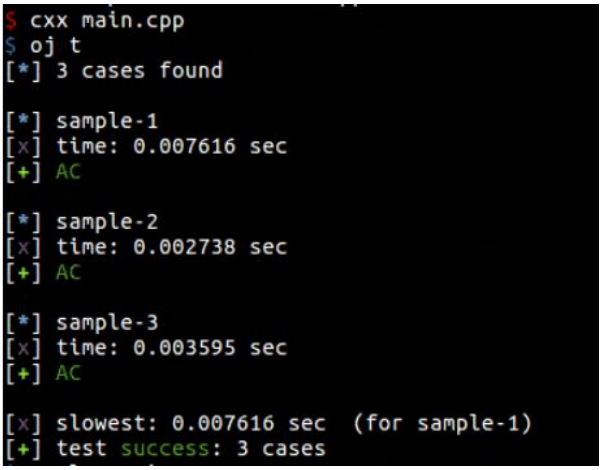


Figura 22. Exemplo de um teste

5.2 Online Judge Tools

Outro projeto semelhante, é o programa desenvolvido pelo Kimiyuki Onaka [8] que consiste numa ferramenta que permite a submissão de soluções a exercícios e/ou projetos não complexos em diversas plataformas semelhantes ao Drop Project. O programa funciona a partir da linha de comando e usa a linguagem Python para o correr.

Como foi referido, é possível através desta ferramenta ter acesso a diversas plataformas de "*online judge"* sem ter de estar a ir em cada lugar para fazer a submissão, download de casos simples ou até download de testes do sistema. O utilizador faz o download dos testes e depois faz a submissão do código através de um comando e a ferramenta vai mostrando os testes que falhou e os que passaram, juntamente com a informação do tempo de execução para cada teste realizado.

5.3 CourseMarker [9]

Uma ferramenta desenvolvida na Universidade de Nottingham. As principais vantagens são consideradas como sendo a escalabilidade, capacidade de manutenção e a segurança. As linguagens de programação suportadas para a classificação de trabalhos são Java e C++. A arquitetura da ferramenta contém 6 subsistemas:

- *Login* - Controla todo o processo de autenticação;
- Submissão - Recebe as diferentes submissões com precisão;
- Curso - Guarda a informação do processo;
- Classificação - Encarrega-se do processo de classificação e do armazenamento dos arquivos e da classificação obtida;
- Auditoria - Tem a responsabilidade de registar todas as ações realizadas no sistema;
- E um subsistema que controla a comunicação entre os outros.

Como métrica para estabelecer uma nota, considera tipografia (formatação, comentários, etc.), funcionalidade através de casos de teste, uso de estruturas de programação e verificação no

design e relações entre os objetos. Trabalha com tecnologias como Java RMI [10] (Remote Method Invocation) para comunicação entre os subsistemas, expressões regulares para verificar resultados e DATsys [11] para verificar o design dos objetos.

O sistema também contém funcionalidades extras como: feedback e as classificações podem ser customizadas, existe um sistema de detecção de plágio, o número de submissões e o uso de CPU pode ser alterado e finalmente, faz se o uso de um ambiente *sandbox* para detetar e evitar qualquer tipo de execução de código malicioso.

5.4 Marmoset

Foi desenvolvido na Universidade de Maryland com o principal objetivo de lidar com a submissão de projetos de programação de alunos, testes e revisão de código. Uma das grandes vantagens é fazer uma *snapshot* sobre o progresso do aluno, para que o desenvolvimento do aluno possa ser analisado em detalhe.

Originalmente às linguagens de programação suportadas eram Java, C, Ruby e Caml Objective. Segundo a página oficial[8], a ferramenta funciona para todas as linguagens de programação.

A arquitetura inclui: o J2EE (Java Enterprise Edition) webserver, uma base de dados em SQL e um ou mais servidores para fazer o *build* dos projetos. Estes últimos, são usados num ambiente seguro e isolado para prevenir a execução de código malicioso. A disposição dos servidores ajudam a providenciar escalabilidade e segurança. As métricas usadas para classificar uma submissão incluem análise dinâmica[9] e estática[10].

5.5 JavaBrat

Esta ferramenta foi desenvolvida no âmbito de uma tese de mestrado na universidade de San José State University [12]. Tem suporte para duas linguagens de programação, Java e Scala.

[8] https://marmoset.cs.umd.edu/
[9] Análise feita através de casos de teste
[10] É analisado o código fonte sem correr o programa

A ferramenta faz uso do Java para desenvolver o software de classificação e PHP para a construção do plugin para o Moodle. O design contém 3 módulos importantes:

- Um servidor da Moodle com o plugin;
- Um módulo que contém os classificadores dependendo da linguagem submetida e um repositório de problemas;
- E por último, o módulo é o JavaBrat que tem uma série de serviços que chamam os classificadores e os problemas

Apesar do plugin funcionar com o Moodle, está ferramenta também consegue funcionar a partir de uma interface web que foi desenvolvida como parte do projeto. Está interface foi desenvolvida com recurso ao JSF (Java Server Faces) 2.0 e os serviços foram implementados usando JAX-RS. Está ferramenta é semi-automática, logo não dispensa a revisão do *report* gerado pelo sistema para a existência de erros de coerência nos resultados.

5.6 Conclusão

As ferramentas acima mencionadas, acabam por ter semelhanças com o Drop Project na forma como funcionam. No entanto, a maior parte destas ferramentas funcionam em ambiente de browser, obrigando aos utilizadores a realizarem demasiados clicks para poderem fazer uma simples submissão. Uma desvantagem da maior parte das ferramentas é não terem plugins que integram com o *IDE*, como por exemplo: IntelliJ, NetBeans, Eclipse, entre outros. De modo a simplificar o processo de submissão e visualização dos resultados. Isto é um dos pontos que o presente trabalho de desenvolvimento do plugin visa resolver.

6. Método e Planeamento

O primeiro passo para a realização do projeto foi determinar o problema que este projeto visa resolver, seguidamente procedeu-se a verificar a sua viabilidade, isto foi feito com base na elaboração de um questionário e pedido aos alunos do curso de Licenciatura em Engenharia Informática, Informática de Gestão e Redes e Telecomunicações para que o respondessem. Após terem sido compilados os resultados, identificou-se que existe um grau de ansiedade alto nos alunos em alguns pontos chaves, já referidos no Capítulo 2.

Foi realizado um levantamento de requisitos funcionais e não funcionais e elaborado um plano de desenvolvimento do plugin, isto é, a escolha da linguagem a ser usada para o desenvolvimento, a plataforma para a qual o plugin será desenvolvido, entre outros. Tendo o plano feito, o projeto deu início, com o desenvolvimento de diálogos e opções básicas para que fosse possível ter um conhecimento maior sobre o uso da API oferecido pela IntelliJ. Tendo obtido novos conhecimentos, foi possível iniciar a implementação do login entre o plugin e plataforma do Drop Project em máquina local. Após terem sido corrigidos erros que foram surgindo, foi implementado mais funcionalidades como obter a lista de *assignments* do aluno, e permitir a submissão de projetos. Seguidamente efetuou-se a integração com IDE, permitindo que o plugin fosse instalado numa instância do IDE, dando uma experiência mais próxima do final. Entretanto devido a situações fora do controlo, não foi possível ligar o plugin com o servidor de autenticação da Lusófona, e como solução alternativa, fez-se o *deploy* da plataforma Drop Project (que estava a correr na máquina local) para o Heroku, está é uma plataforma como serviço (PaaS) que suporta várias linguagens de programação, que permitem aos programadores facilmente efetuar o *deploy* da sua aplicação sem ter de preocupar com a infraestrutura. Isto permitiu a realização de alguns testes de modo a obter um feedback, mesmo não sendo em grandes números.

Algumas das tarefas finais que tinham sido descritas no calendário, como a integração do plugin com a autenticação federada da Lusófona, não foi possível de se efetuar por razões

imprevisíveis. No entanto, soluções alternativas, mesmo que não fossem às esperadas inicialmente, permitiram que o projeto pudesse realizar algumas dessas tarefas finais.

Na seguinte figura, pode se observar o número de commits efetuados desde o início do desenvolvimento. O número mostra que houve uma maior frequência de commits na fase inicial do desenvolvimento, tendo está frequência diminuído pois com o desenvolvimento de funcionalidades já efetuadas, limitou-se a fazer pequenas correções de bugs.

Figura 23. Histórico de *commits*

7. Resultados

Um plano de teste é feito para colaborar com o desenvolvimento de software. É através deste plano que os componentes são verificados e validados para garantir o bom funcionamento do programa. De modo a poder testar o plugin, serão realizados teste de utilizador, com alunos de modo a poder testar o plugin num ambiente real onde será feito o seu uso futuro e encontrar eventuais falhas para poderem ser posteriormente resolvidas.

7.1 Teste de Utilizador

7.1.1. Ambiente de Teste

- Os testes deverão de ser realizados num computador com acesso a Internet e no IDE IntelliJ
- Os utilizador deve ter o *plugin* instalado manualmente ou ter feito a instalação pela *marketplace*.

7.1.2 Objetivos dos testes

Os testes a serem realizados devem cumprir os seguintes objetivos:

- O plugin deve efetuar o *login* dos alunos
- O plugin deve mostrar a lista de *assignment* disponíveis ao aluno
- O plugin deve permitir efetuar a submissão de projetos
- O plugin deve possibilitar a visualização do enunciado e outros detalhes do *assignment*
- O plugin deverá efetuar *login* automático após o primeiro *login*

7.1.3 Testes a serem realizados

Identificação do Teste	Passos a serem realizados	Resultado esperado
Teste 1 - *Login*	Efetuar o *login* no plugin	O plugin deverá mostrar um diálogo de sucesso/erro quando é efetuado o *login*
Teste 2 - Listar *assignments*	A partir do menu do plugin selecionar a opção para listar os *assignments*	Deverá ser mostrado um diálogo com os assignments disponíveis ao aluno, juntamente com informação sobre a data limite e a linguagem de programação a ser usada no *assignment*
Teste 3 - Ver enunciado do *assignment*	Selecionar a opção de detalhe no diálogo de *assignments*	O plugin deverá mostrar em formato HTML o enunciado do assignment, e caso exista links no texto, deverá ser possível fazer click
Teste 4 - Verificar submissões efetuadas	Selecionar a opção de submissões efetuadas	Deverá ser mostrado um novo diálogo com a lista de submissões efetuadas pelo aluno até a data.
Teste 5 - Verificar relatório da submissão	Selecionar a opção de mostrar o relatório no diálogo de submissões efetuadas	Caso não tenha havido nenhum erro, deverá ser mostrado um diálogo a avisar ao aluno. Em caso contrário deverá ser mostrado o relatório de erros

Teste 6 - Selecionar o *assignment*	Selecionar o *assignment*	Deve ser mostrado uma mensagem a avisar que o *assignment* foi selecionado e o *status bar* do IntelliJ deve refletir essa escolha
Teste 7 - Submeter o projeto	A partir do menu do plugin deve ser submetido o projeto	Em caso de erro, o aluno deve ser avisado sobre o erro.
Teste 8 - *Log out*	Efetuar o *log out*	Deve ser mostrado uma mensagem de aviso após ter sido efetuado o *log out* com sucesso

Tabela 1. Plano de testes

7.2 Resultados, *outputs* e *outcomes*

O projeto teve como um dos pontos importantes o levantamento de requisitos. Estes requisitos permitiram perceber quais as funcionalidades que eram necessários priorizar e os que não. O plugin atingiu todos os objetivos que foram traçados.

O login no sistema (RF1) permite a autenticação do plugin com a plataforma do Drop Project, usando para este caso a API de login, já existente na plataforma [Anexo C.1]. Após a sua autenticação, o aluno tem acesso a lista de submissões (RF2) mediante uma opção no menu e consegue ter acesso ao enunciado de qualquer *assignment* que lhe tenha sido atribuído, não só, mas o aluno consegue também visualizar os diferentes resultados dos testes (RF5) realizados a cada submissão que tenha sido efetuada (RF3). Outros requisitos como a criação do ficheiro AUTHORS.txt (RF4), o diálogo que mostra se o login foi sucedido ou não (RF6) e o *login* é efetuado automaticamente (RF7), foram implementados com sucesso.

8. Conclusão e trabalhos futuros

O plugin desenvolvido durante o ano letivo, conseguiu atingir parcialmente os seus objetivos, sendo que não foi possível verificar se de facto, o plugin permitiu uma diminuição nos níveis de ansiedade que foram notados nos inquéritos realizados aos alunos. Devido a situações fora do controlo, não foi possível efetuar a ligação do plugin com o servidor da Lusófona e deste modo aplicar o plugin num exercício prático de modo a poder obter testes numa situação de avaliação real. Apesar destes percalços, às funcionalidades inicialmente planeadas foram desenvolvidas com sucesso, alguns mostrando terem um grau de dificuldade superior, como representar o nome do assignment atualmente escolhido no status bar. Outras mais simples como a criação do ficheiro ZIP, tendo esta sido feita com o uso de uma *3rd party library*.

Este projeto permitiu aumentar os conhecimentos em Kotlin e o seu uso em pedidos aos *webservices,* a criação de diálogos usando *libraries* de Java e a integração de tudo isto num plugin para o IntelliJ.

O projeto tem funcionalidades que podem ser melhoradas, nomeadamente na parte onde o plugin guarda às credenciais do aluno, o relatório de erros poderia oferecer opção para ir a linha onde o erro se encontra. Funcionalidades novas como o *cool-off* do *assignment*, iriam permitir aos alunos saberem se podem submeter o projeto ou se devem de esperar um determinado tempo para voltarem a efetuar às submissões que não foi um requisito inicialmente definido.

Bibliografia

[1] «IntelliJ Platform SDK», *JetBrains IntelliJ Platform SDK*. http://www.jetbrains.org/intellij/sdk/docs//welcome.html (acedido Nov. 13, 2019).

[2] G. Municipal, «Engenharia de Software». http://www.governancamunicipal.sp.gov.br/conteudo/arquivos/Analise%20de%20requisitos.pdf (acedido Jan. 25, 2020).

[3] C. dos P. da Wikimedia, «Requisito não funcional – Wikipédia, a enciclopédia livre», *Fundação Wikimedia, Inc.*, Out. 06, 2008. https://pt.wikipedia.org/wiki/Requisito_n%C3%A3o_funcional (acedido Jan. 25, 2020).

[4] Contributors to Wikimedia projects, «IntelliJ IDEA - Wikipedia», *Wikimedia Foundation, Inc.*, Set. 07, 2004. https://en.wikipedia.org/wiki/IntelliJ_IDEA (acedido Nov. 13, 2019).

[5] «Main Types of Plugins», *JetBrains IntelliJ Platform SDK*. http://www.jetbrains.org/intellij/sdk/docs//basics/types_of_plugins.html (acedido Nov. 21, 2019).

[6] «FAQ - Kotlin Programming Language», *Kotlin*. https://kotlinlang.org/docs/reference/faq.html (acedido Nov. 13, 2019).

[7] «CodeChef | Programming Competition,Programming Contest,Online Computer Programming». https://www.codechef.com/ (acedido Nov. 22, 2019).

[8] «Kimiyuki Onaka». https://kimiyuki.net/ (acedido Nov. 22, 2019).

[9] «CourseMarker - automatedprogrammingtool». https://sites.google.com/site/automatedprogrammingtool/rivew-of-similar-products/coursemarker (acedido Jun. 24, 2020).

[10] «Trail: RMI (The Java™ Tutorials)». https://docs.oracle.com/javase/tutorial/rmi/ (acedido Jun. 24, 2020).

[11] C. Higgins, P. Symeonidis, e A. Tsintsifas, «Diagram-based CBA using DATsys and CourseMaster», *International Conference on Computers in Education, 2002. Proceedings.* doi: 10.1109/cie.2002.1185893.

[12] A. Patil, «Automatic Grading of Programming Assignments», Master of Science (MS), San José State University, 2010.

Anexos

Anexo A - Algoritmo de Cifragem

```
private fun encrypt(textToEncrypt: ByteArray): ByteArray {
   val enc = ByteArray(textToEncrypt.size)

   for (i in textToEncrypt.indices) {
       enc[i] = (if ((i % 2 == 0)) textToEncrypt[i] + 1 else textToEncrypt[i] -
1).toByte()
   }

   return enc
}
```

Anexo B - Algoritmo de “desencriptação”

```
fun decrypt(textToDecrypt: ByteArray): ByteArray {
   val enc = ByteArray(textToDecrypt.size)

   for (i in textToDecrypt.indices) {
       enc[i] = (if ((i % 2 == 0)) textToDecrypt[i] - 1 else textToDecrypt[i] +
1).toByte()
   }

   return enc
}
```

Anexo C - Documentação da API

Código de erros da API

Sempre que um pedido API falha, o servidor devolve um código de erro e uma descrição em formato JSON do erro.

Erros Comuns:

Código	Texto	Informação
`400`	`Bad Request`	O pedido era inválido ou não pode ser atendido de outra forma. Uma mensagem de erro explicará melhor às razões.
`401`	`Unauthorized`	Credenciais de autenticação em falta ou incorrectas. Este erro pode ser devolvido em outras circunstâncias indefinidas.
`403`	`Forbidden`	O pedido é compreendido, mas foi recusado ou o acesso não é permitido.
`500`	`Internal Server Error`	Erro ao processar o pedido. Em alguns casos é devolvido uma mensagem da causa do erro.

1. *Login*

De modo a ter acesso às funcionalidades do *plugin* deve ser realizado o *login* na plataforma. O *header* de autorização deverá ser enviado em todos os futuros pedidos.

Exemplo do pedido:

```
curl --location --request POST 'http://localhost:8080/login' \
--header 'Authorization: Basic <base64>' \
```

Parâmetros do pedido HTTP POST:

Tipo	Objeto	Descrição
`HEAD`	`<base64>`	**[Obrigatório]** Deve ser enviado às credenciais codificadas em `base64`

Resposta:

Estado	Resposta
`200`	Página principal em HTML do utilizador
`401`	`{ "timestamp": "2020-04-17T14:09:58.114+0000", "status": 401, "error": "Unauthorized", "message": "Unauthorized", "path": "/login" }`

2. **Listar *Assignments***

Para conseguir obter a lista de *assignments* que estão atribuídas ao utilizador.

Exemplo do pedido:

```
curl --location --request GET
'http://localhost:8080/api/v1/assignmentList' \
--header 'Authorization: Basic <base64>' \
```

Parâmetros do pedido HTTP GET:

Tipo	Objeto	Descrição
`HEAD`	`<base64>`	**[Obrigatório]** Deve ser enviado às credenciais codificadas em `base64`

Resposta:

Estado	Resposta

`200`	Devolve uma lista contendo todos os *assignments* disponíveis ao utilizador. O parâmetro `date` devolve `null` quando o *assignment* não tem data limite de submissão. O parâmetro `html` devolve uma *String* em formato HTML
	
	Exemplo de resposta:
	
	`[`
	`  {`
	`  "id": "sampleJavaProject",`
	`  "language": "JAVA",`
	`  "date": "null",`
	`  "html": "<assignment_details>",`
	`  }`
	`]`

`401`	`{`
	`  "timestamp": "2020-04-17T11:58:02.541+0000",`
	`  "status": 401,`
	`  "error": "Unauthorized",`
	`  "message": "Unauthorized",`
	`  "path": "/api/v1/assignmentList"`
	`}`

3. **Submissão do ficheiro ZIP**

   Para o utilizador conseguir submeter o projeto em formato ZIP.

   Exemplo do pedido:

```
curl --location --request POST 'http://localhost:8080/upload/' \
--header 'Authorization: Basic <base64>' \
--form 'file=@path_to_ZIP' \
--form 'assignmentId=<assignmentId>'
```

Parâmetros do pedido HTTP POST:

Tipo	Objeto	Descrição
`HEAD`	`<base64>`	**[Obrigatório]** Deve ser enviado às credenciais codificadas em `base64`
`POST`	`file`	**[Obrigatório]** É necessário fornecer o path do ficheiro ZIP para que seja feita o upload a plataforma de Drop Project
`POST`	`<assignmentId>`	**[Obrigatório]** O assignment ID deve ser fornecido de modo a que a plataforma consiga perceber para qual assignment é que se está a fazer o upload

Resposta:

Estado	Resposta

`200`	Devolve o ID da submissão para o utilizador poder obter o relatório da submissão
	
	`{`
	`"submissionId": "31"`
	`}`

401	{
	"timestamp": "2020-04-17T11:58:02.541+0000",
	"status": 401,
	"error": "Unauthorized",
	"message": "Unauthorized",
	"path": "/upload"
	}

500	O ficheiro de *AUTHORS.txt* não contém o número de aluno que está a fazer a submissão {"error": "O utilizador que está a submeter tem que ser um dos elementos do grupo."}
500	Não foi fornecido o ID do assignment como um dos parâmetros do POST {"error": "java.lang.IllegalArgumentException: assignmentId is null"}

500	Submissão sem ter fornecido nenhum parâmetro no POST
	
	{
	"timestamp": "2020-04-17T12:07:30.983+0000",
	"status": 500,
	"error": "Internal Server Error",
	"message": "Current request is not a multipart request",
	"path": "/upload/"
	}

500	O ficheiro ZIP está vazio {"error": "Falha a gravar ficheiro => Failed to unZIP projeto.ZIP"}

400	O parâmetro "file" não foi fornecido
	
	{
	"timestamp": "2020-04-17T15:42:13.262+0000",
	"status": 400,
	"error": "Bad Request",
	"message": "Required request part 'file' is not present",
	"path": "/upload/"
	}

4. **Visualizar *Build Report***

   Abre o relatório da submissão no browser usando o ID recebido no pedido anterior.

Exemplo do pedido:

```
curl --location --request GET
'http://localhost:8080/buildReport/<submissionId>' \
--header 'Authorization: Basic <base64>' \
```

Parâmetros do pedido HTTP GET:

Tipo	Objeto	Descrição
HEAD	<base64>	**[Obrigatório]** Deve ser enviado às credenciais codificadas em base64
URL_PARAM	<submissionId>	**[Obrigatório]** É necessário especificar o ID da submissão que pretende obter o relatório.

Resposta:

Estado	Resposta
200	Devolve em HTML a página do relatório da submissão

401	{
	"timestamp": "2020-04-17T11:58:02.541+0000",
	"status": 401,
	"error": "Unauthorized",
	"message": "Unauthorized",
	"path": "/upload"
	}

403	Sem permissão para ver o relatório da submissão não associada ao utilizador `Access denied: a21705201 is not allowed to view this report`

5. **Listar submissões**

   Devolve a lista de submissões efetuadas pelo aluno, num determinado *assignment.*

Exemplo do pedido:

```
curl --location --request GET
'http://localhost:8080/api/v1/submissionsList/<assignmentId>' \
--header 'Authorization: Basic <base64>'
```

Parâmetros de pedido HTTP GET:

Tipo	Objeto	Descrição
HEAD	<base64>	**[Obrigatório]** Deve ser enviado às credenciais codificadas em base64
URL_PARAM	<assignmentId>	**[Obrigatório]** É necessário especificar o ID do assignment que pretende, para obter a lista de submissões efetuadas.

Resposta:

Estado	Resposta
200	Devolve uma lista com todas as submissões efetuadas pelo aluno para um determinado *assignment* e o respetivo resultado dos testes realizados. Caso a submissão tenha passado todos os testes, o `report` é devolvido como null


```
[
    {
        "submissionId": 29,
        "submissionDate": "01 Jan 2020 10:34",
        "report": "FAILURE:
org.dropProject.samples.sampleJavaAssignment.TestT
eacherProject.testFindMax\njava.lang.AssertionErro
r: expected:<7> but was:<-2147483648>\n\tat
org.dropProject.samples.sampleJavaAssignment.TestT
eacherProject.testFindMax(TestTeacherProject.java:
12)\n\n",
        "summary": "Tests run: 2, Failures: 1,
Errors: 0, Time elapsed: 0.012 sec",
        "assignmentId": "sampleJavaProject"
    },
    {
        "submissionId": 37,
        "submissionDate": "02 Jan 2020 19:05",
        "report": null,
        "summary": "Tests run: 2, Failures: 0,
Errors: 0, Time elapsed: 0.007 sec",
        "assignmentId": "sampleJavaProject"
    }
]
```

401	{
	"timestamp": "2020-04-25T16:30:27.934+0000",
	"status": 401,
	"error": "Unauthorized",
	"message": "Unauthorized",
	"path": "/api/v1/submissionsList/sampleJavaProject"
	}