NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTAMENTO DE INFORMÁTICA

BERNARDO DA PALMA A. SOARES DE ALBERGARIA

Licenciatura em Engenharia Informática

AMBIENTE DE DESENVOLVIMENTO PARA CRIAÇÃO DE ACTIVIDADES DE PROGRAMAÇÃO

MESTRADO EM ENGENHARIA INFORMÁTICA

Universidade NOVA de Lisboa
Março, 2022

NOVA SCHOOL OF SCIENCE & TECHNOLOGY

DEPARTAMENTO
DE INFORMÁTICA

AMBIENTE DE DESENVOLVIMENTO PARA CRIAÇÃO DE ACTIVIDADES DE PROGRAMAÇÃO

BERNARDO DA PALMA A. SOARES DE ALBERGARIA

Licenciatura em Engenharia Informática

Orientadora: Fernanda Barbosa
Professor Auxiliar, NOVA University Lisbon

Coorientadora: Carmen Morgado
Professor Auxiliar, NOVA University Lisbon

MESTRADO EM ENGENHARIA INFORMÁTICA

Universidade NOVA de Lisboa
Março, 2022

Ambiente de desenvolvimento para criação de actividades de programação

Agradecimentos

Gostaria de deixar o meu agradecimento em relação às orientadoras da minha dissertação, Prof. Fernanda Barbosa e Prof. Carmen Morgado, pela motivação e apoio que me deram durante este último período de universidade.

Deixo também uma saudação especial ao corpo docente do Departamento de Informática da Universidade NOVA de Lisboa por todo o conhecimento que me foi passado ao longo dos anos.

"You cannot teach a man anything; you can only help him discover it in himself." (Galileo)

Resumo

Nas últimas décadas, testemunhámos um enorme desenvolvimento da tecnologia. Este desenvolvimento alterou significativamente a forma como lidamos com as nossas tarefas diárias, sejam estas relativas a ocupação profissional ou lazer. A crescente utilização de dispositivos electrónicos de computação nas nossas tarefas diárias é acompanhada com um interesse, cada vez maior, em programação. Um dos impactos deste interesse em programação, na área do ensino, é o aparecimento de metodologias como o pensamento computacional, que visa aplicar a metodologia de solução de problemas de programação noutros problemas mais genéricos ou noutras áreas fora do contexto das ciências da computação.

Nos últimos anos, foram desenvolvidas plataformas para ajudar os utilizadores a cultivar o pensamento computacional e técnicas de programação, algumas delas utilizando programação por blocos. Programação por blocos tornou-se uma forma para introduzir pensamento computacional e técnicas de programação, apresentando ambientes apelativos e intuitivos para resolver problemas. Uma desvantagem presente nas plataformas mais utilizadas é não permitirem a criação de novas actividades de programação, isto é, exercicios de programação.

Neste documento apresenta-se uma aplicação web *Bricks* que sirva de ferramenta de apoio a aulas e que permita que sejam criadas actividades de programação por blocos de uma forma simples. A possibilidade de criação de actividades de programação beneficia professores e alunos, permitindo aos primeiros criar conteúdos que encaixem exactamente com os seus programas escolares e, em simultâneo, fornecendo aos segundos um ambiente apelativo onde podem realizar as actividades de programação. Estas actividades de programação têm como finalidade estimular o pensamento computacional e introduzir conceitos básicos de programação aos utilizadores.

Palavras-chave: Ambientes de Aprendizagem, Programação por Blocos, Actividades de Programação, Pensamento Computacional.

Abstract

In the past few decades, we have witnessed a tremendous progress in the evolution of technology. This evolution changed how we perform a great number of tasks on our daily basis, both for work and leisure. Electronic devices have become increassingly more present in our daily tasks which has, over the last decades, also increased the interest in programming. This rise in intrest in programming impacted in certain areas of expertise such as education, where certain methodologies like computiational thinking, which applies the programming methodology for problem solving to either more generic problems or even problems beyond computational science's scope, have come into existence.

In the last few years, many platforms have been created to help users develop computational thinking and programming skills, using block programming. Block programming has become a standard for introducing these skills, presenting attractive and intuitive settings to solve exercises. A major disadvantage presented by the most commonly used platforms is the lack of support for instructors to create their own programming activities or exercises.

In this dissertation, we propose a platform to allow users to create programming activities without much prior expertise. The creation of custom programming activities benefits both teachers and students alike, enabling teachers to create content to fit their exact planned program and presenting learners with more custom study materials and providing students with an appealing and intuitive environment where they can preform programming activities. These activities aim to stimulate computational thinking while introducing basic programming knowledge to users.

Keywords: Learning Environments, Block Programming, Programming Activities, Computational Thinking.

Índice

Índice de Figuras

Índice de Tabelas

1

Introdução

Nestas últimas décadas, temos acompanhado um desenvolvimento significativo da tecnologia. Estes desenvolvimentos trouxeram-nos inúmeras aplicações nas mais diversas áreas como, por exemplo, no ensino. Para acompanhar estes desenvolvimentos da tecnologia na área do ensino, é necessário criar estratégias para usufruir destas novas tecnologias, com vista a adaptar os métodos de ensino, como por exemplo a utilização de plataformas digitais como ferramenta para apoio da componente prática no ensino. Estas novas estratégias, tendencialmente, focam-se em métodos mais práticos, para complementar as aulas tradicionais, em que um professor apresenta o material de estudo presente em livros através de slides. O objectivo destas novas estratégias não é substituir a presença do agente educador, mas tornar o processo de aprendizagem mais eficiente e cativante para os alunos.

1.1 Motivação e Enquadramento

Nos últimos 30 anos, os avanços tecnológicos tornaram computadores e dispositivos móveis acessíveis à população. A banalização da tecnologia tornou comum a utilização de dispositivos electrónicos tanto no meio profissional como no meio pessoal. Como tal, hoje em dia, é comum a maioria das pessoas terem acesso a computadores e internet, permitindo assim formas mais apelativas de apresentar matérias, procurando despertar maior interesse nos alunos. Também devido à banalização da utilização de dispositivos electrónicos, tem surgido um interesse crescente em áreas de programação, com objectivo profissional de desenvolvimento de aplicações e funcionalidades para tais dispositivos.

Estes avanços tecnológicos fizeram também acentuar a importância de fomentar o pensamento computacional como instrumento na educação. Pensamento computacional é definido como o processo de pensamento necessário para compreender e resolver um problema e expressar as soluções de forma compreensível para humanos e máquinas [33]. Na publicação de *Jeanette Wing* [32] são descritos paralelismos entre formas de resolver problemas do dia-a-dia e métodos usados na resolução de problemas computacionais, assim como o impacto que a área da informática tem sobre outras áreas cientificas. Segundo

Jeanette Wing, estes paralelismos servem como argumento para a importância de promover o pensamento computacional nos estabelecimentos de ensino, de forma a ajudar as novas gerações a lidar melhor com os problemas que vão surgindo.

Existe uma ligação profunda entre pensamento computacional e a área de programação. De forma a introduzir o pensamento computacional e competências de programação por diferentes áreas e para diferentes faixas etárias, nomeadamente as mais jovens, têm-se vindo a utilizar um tipo de programação visual, a programação por blocos [30]. Programação visual consiste em gerar programas através da manipulação de elementos gráficos [12]. Estes elementos gráficos, no caso de programação por blocos, são blocos que representam as instruções. Este tipo de programação permite ao utilizador focar-se na essência dos problemas de programação sem ser necessário dominar o léxico de uma linguagem de programação ou haver preocupação com erros comuns de sintaxe. Programação por blocos procura então oferecer uma forma de aprendizagem mais apelativa que programação textual através de um conjunto de características que tornam o desenvolvimento de programas mais intuitivo. Algumas destas características são: (1) adaptação do léxico que cada bloco exibe, para que seja adequado ao utilizador alvo da actividade, por exemplo, um bloco de ciclo pode ser referido por 'repetir x vezes' ou '*loop* x', e (2) construir programas arrastando blocos e ligando-os entre si, em vez de escrever instruções. Estas características são exemplos de como ambientes de programação por blocos podem ser adaptados de acordo com a faixa étaria ou experiência de utilização do utilizador [30].

No âmbito de plataformas que utilizem programação por blocos, surgiram algumas iniciativas como o *Code.org*[1], o *Scratch*[2] e o *Snap!*[3] e, como exemplo mais recente, o *RoBlocks* [9]. Estas iniciativas têm actividades que promovem o ensino de programação e estimulam o pensamento computacional. O *Code.org* surge em 2013 e oferece aos seus utilizadores, diversas actividades de programação, tendo cada uma destas um objectivo bem definido. O *Code.org* está desenhado de forma a que professores possam utilizar a plataforma como ferramenta de apoio para aulas, permitindo a um professor criar turmas, atribuir cursos e acompanhar o progresso dos seus alunos na plataforma. O *Scratch*, lançado em 2007, e o *Snap!*, lançado em 2011, proporcionam aos seus utilizadores um ambiente para desenvolverem as suas actividades de programação, onde o objectivo é definido pelos próprios utilizadores. Estas duas plataformas estão desenhadas de forma a apelar à criatividade dos utilizadores, enquanto aprendem conceitos matemáticos e de programação [22]. Nas plataformas *Scratch* e *Snap!*, um utilizador pode executar, ver e comentar qualquer actividade, criar e publicar actividades. O *RoBlocks* [9], criado em 2016, é uma aplicação desenhada para ensinar conceitos de programação e algoritmos, utilizando *Mobile Robotics* como contexto. *Mobile Robotics* é um ramo do conhecimento que se foca no estudo associado com a programação de robôs móveis com objectivo de realizar tarefas, podendo executar movimentos em duas ou três dimensões [16]. Apesar

[1] *Learn today, build a brighter tomorrow.* | *Code.org*, https://code.org/
[2] *Scratch - Imagine, Program, Share*, https://scratch.mit.edu/
[3] *Snap! Build Your Own Blocks*, https://snap.berkeley.edu/

do *RoBlocks* utilizar programação por blocos, o contexto da aplicação foca-se muito em robótica.

No contexto do ensino de programação, o *feedback* acerca da avaliação da solução apresentada para resolver um dado problema é uma característica importante. A plataforma de concursos de programação *Mooshak* [4] e o *plug-in* do *Moodle*, *CodeRunner* [5], são exemplos evidentes da incorporação de avaliação de soluções de exercícios de programação no contexto do ensino de programação. O *Mooshak* é um sistema que foi desenhado para avaliar concursos de programação, avaliando código submetido de acordo com um *input/output* esperado e um conjunto de restrições como, por exemplo, tempo limite de execução. Um aspecto desfavorável deste sistema é a interface gráfica, mais focada nas características funcionais, não contribuindo favoravelmente para uma experiência de utilização apelativa. O *CodeRunner* é um exemplo de um *plug-in* que permite executar código submetido por alunos num questionário *moodle*, sendo o código avaliado de acordo com um conjunto de *input/output* previamente fornecido. Apesar do suporte multi-linguagem, nem o *Mooshak*, nem o *CodeRunner*, suportam linguagens de programação visual.

Contudo, não existe uma forma simples e acessível, para um professor, de criar actividades de programação por blocos através destas plataformas. Uma actividade de programação por blocos, no contexto da dissertação, é um problema que deve ser resolvido através dum programa, utilizando de programação por blocos. Do ponto de vista de um professor, cuja unidade curricular assente em programação, seria interessante poder usar uma aplicação que lhe permitisse criar actividades para os seus alunos exactamente alinhado com o seu plano curricular e poder acompanhar o progresso dos seus alunos. Para os alunos, é apelativo poder usufruir de uma plataforma em que possam realizar e consultar actividades de programação de uma forma simples e apelativa.

1.2 Objectivos

O objectivo desta dissertação consiste em criar uma aplicação web que permita, a um utilizador do tipo professor, criar actividades de programação por blocos de uma forma simples. Estas actividades poderão ser executadas por utilizadores do tipo aluno, ou seja, editar programas para resolver os problemas enunciados nas actividades de programação. Uma actividade de programação neste sistema é um problema que terá que ser resolvido através dum programa, usando a programação por blocos.

A aplicação web será dividida em duas componentes principais, a primeira componente servirá para criar actividades de programação e a segunda servirá para editar as soluções ou programas associadas às actividades de programação.

O utilizador do tipo professor terá ao seu dispor um ambiente onde pode:

- Criar actividades de programação por blocos.

[4] *Mooshak: a Web-based multi-site programming contest system*, https://doi.org/10.1002/spe.522
[5] *CodeRunner question type - MoodleDocs*, https://docs.moodle.org/39/en/CodeRunner_question_type

- Disponibilizar actividades de programação às turmas (grupos de alunos) que pretender.
- Consultar *feedback* sobre a realização das actividades de programação que atribuiu às suas turmas (grupos de alunos).

Um utilizador do tipo aluno terá acesso a um ambiente onde pode editar programas que sejam a resolução de exercícios disponibilizados. Para além disso, é possível, para os programas do aluno, executar e validar por um conjunto de testes associado a cada actividade, previamente criados pelo professor. Exercícios de programação já concluídos por um aluno poderão ser sempre consultados como material de estudo.

1.3 Estrutura do documento

Este documento está organizado em seis capítulos. Neste primeiro capítulo, é introduzido o tema da dissertação, juntamente com a motivação e os objectivos.

No capítulo 2, são apresentados alguns estudos, ambientes e bibliotecas relacionadas com a aplicação proposta, seguida de uma análise comparativa dos ambientes estudados.

No capítulo 3, é apresentada a aplicação web implementada, salientando as funcionalidades da aplicação e o modelo de dados.

No capítulo 4, é apresentado o desenvolvimento e implementação da aplicação, salientando a arquitectura da aplicação, a integração da biblioteca de programação por blocos na aplicação e a estrutura das interfaces da aplicação.

No capítulo 5, são apresentados os testes de usabilidade realizados sobre a aplicação e os seus respectivos resultados.

Finalmente, no capítulo 6, são apresentadas as conclusões a extrair sobre a dissertação, juntamente com alguma reflexão sobre alterações e funcionalidades a adicionar sobre a aplicação.

2

Trabalho relacionado

Neste capítulo, é apresentado um conjunto de tecnologias relevantes para a concepção da aplicação que foi desenvolvida nesta dissertação. Numa primeira instância, é definido o conceito de programação por blocos e actividade de programação. De seguida, são analisados alguns ambientes de programação usados no ensino, de forma a identificar características vantajosas que sejam integráveis na nossa aplicação. Posteriormente, são apresentadas bibliotecas para implementar ambientes de programação por blocos. Para finalizar, são expostas algumas *frameworks* para desenvolvimento de aplicações.

2.1 Programação por blocos

A programação visual, à semelhança da programação textual, tem como objectivo a criação de programas. Na programação visual, os programas são criados a partir da manipulação de elementos gráficos e não através de expressões textuais, como em programação textual [12]. Entre os vários tipos de programação visual, programação por blocos tem características adequadas para ser uma aplicação de programação visual na área do ensino. Características como *drag and drop* de blocos de instruções e *outputs* gráficos, como animações que acompanham a execução de blocos, contribuem para oferecer um ambiente interactivo e intuitivo de forma a cativar os utilizadores, principalmente os mais jovens.

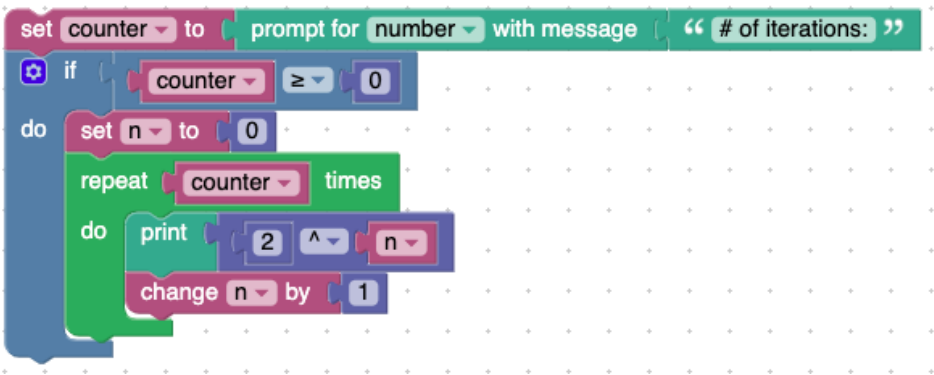
O elemento principal de uma linguagem de programação por blocos é o bloco, ilustrado na Figura 2.1. O bloco consiste numa representação visual de uma instrução simples ou composta. Um programa é gerado através da sequenciação de um conjunto de blocos, exemplificado na Figura 2.2 com um pequeno programa que pede ao utilizador um número e mostra na consola as potências de 2 entre 0 e o valor dado.

A representação visual do bloco está desenhada para ser o mais intuitiva possível, apresentando três características principais: forma, cor e etiqueta. A forma do bloco é um indicador de como cada bloco pode ser encaixado noutros blocos. Como pode ser observado na Figura 2.1, o bloco 'repetir x vezes' tem duas posições para encaixar blocos, onde a primeira, situada no topo, delimita o número de vezes que o ciclo é executado,

portanto só poderão ser encaixados blocos que resultem numa expressão numérica inteira, e a segunda, na parte inferior, encaixa a sequência de blocos que o ciclo irá repetir. À semelhança do bloco 'repetir x vezes', o bloco de condição 'se', apresentado à direita na Figura 2.1, tem também duas posições para encaixar blocos, onde a primeira posição, no topo, apenas permite blocos que representam uma expressão booleana e a segunda posição encaixa a sequência de blocos que será executada dependendo da avaliação da expressão booleana. A Figura 2.2 apresenta um exemplo de utilização dos blocos presentes na Figura 2.1 para gerar um programa. A cor de cada bloco especifica o tipo de instrução que o bloco representa, por exemplo, todos os blocos relacionados com variáveis do programa serão representados pela mesma cor, ilustrado a cor-de-rosa na Figura 2.2. Finalmente, a etiqueta de cada bloco descreve a instrução representada pelo bloco, por exemplo, um bloco de ciclo pode ser representado com a etiqueta 'repetir x vezes'. As etiquetas dos blocos suportam também desde caractéres *unicode* a imagens, de forma a tornar a compreensão de funcionalidade de um bloco mais simples e intuitiva.

Figura 2.1: Exemplos de blocos

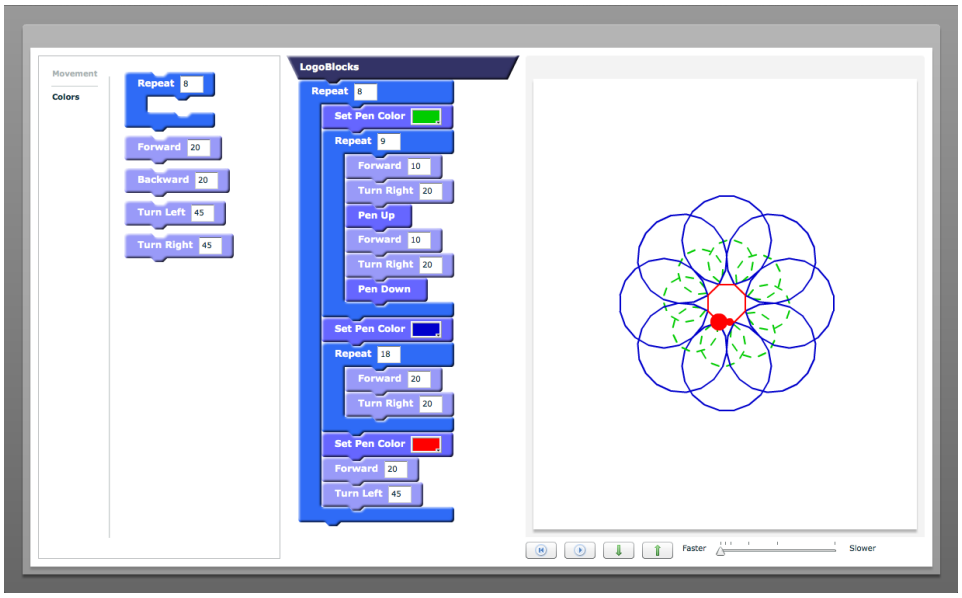
Figura 2.2: Exemplo de um programa utilizando programação por blocos



Ao longo do tempo foram surgindo diversas linguagens de programação, criadas com diferentes objectivos, incluindo linguagens dedicadas ao ensino de programação. O *LogoBlocks* [4], consiste numa implementação das instruções da linguagem *Logo* em elementos gráficos, tirando partido das vantagens de programação por blocos. A Figura 2.3 exibe o interface de um ambiente que permite desenvolver programas em *LogoBlocks*. Nesta

figura podemos observar: a *toolbox* com os blocos disponíveis, à esquerda, a sequência de blocos que definem o programa, no centro da imagem, e na parte direita o *output* gráfico resultante da execução do programa.

Figura 2.3: Exemplo de uma implementação de *LogoBlocks* (*Flashblocks*) [10]



Desde o aparecimento do *LogoBlocks*, desenhado para tirar partido das vantagens de programação por blocos no contexto do ensino de programação, surgiram algumas outras linguagens de programação como o *Scratch* e o *Snap!*. Com base nestas liguagens e, consequentemente, no *Google Blockly*, têm surgido plataformas como, por exemplo, o *Scratch* e o *Snap!*. Estas plataformas, desde a sua criação, têm sido actualizadas com funcionalidades e conteúdo de forma a proporcionar a melhor experiência de aprendizagem.

No contexto desta dissertação, é definido actividade de programação como um exercício cuja solução é um programa. Uma actividade de programação por blocos é uma actividade cuja solução é um programa em linguagem de programação por blocos.

2.2 Ambientes de programação dedicados ao ensino

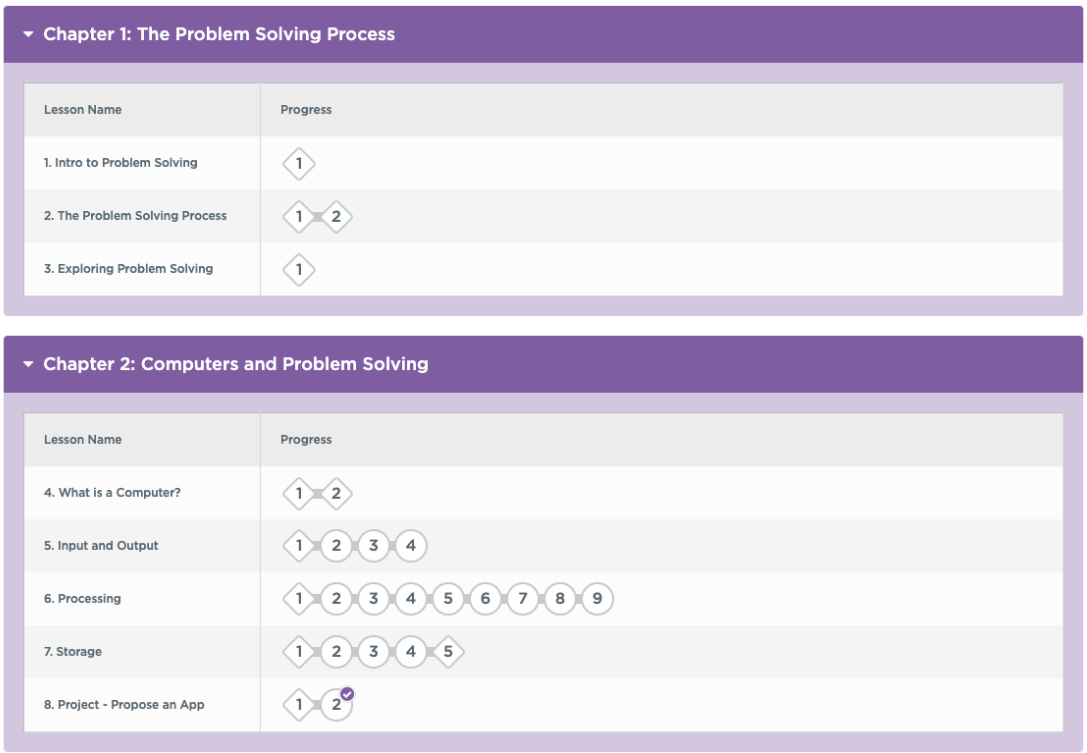
Como exemplo de ambientes de programação dedicados ao ensino enquadram-se plataformas que têm como objectivo incentivar os utilizadores a aprender a programar, promover o pensamento computacional ou avaliar soluções de problemas. Estas plataformas podem ser desde ferramentas de apoio a aulas, que beneficiam professores e alunos, a ambientes que dão uso da criatividade para encorajar os alunos a aprender de uma forma independente. No seguimento desta secção, serão apresentadas plataformas e ferramentas relevantes na área do ensino de forma a expor características que sejam interessantes no contexto do protótipo desta dissertação.

2.2.1 Code.org

O *Code.org* é uma organização sem fins lucrativos dedicada a promover a ciência da computação nas escolas. Um dos objectivos desta organização é tornar ciências da computação e programação uma parte do plano curricular das escolas, juntamente com disciplinas como matemática, fisica e biologia [26]. Para este fim, criaram uma plataforma onde disponibilizam material desde cursos completos e estruturados, com objectivo de serem lecionados, a pequenos conjuntos de actividades para os alunos realizarem através de iniciativa própria.

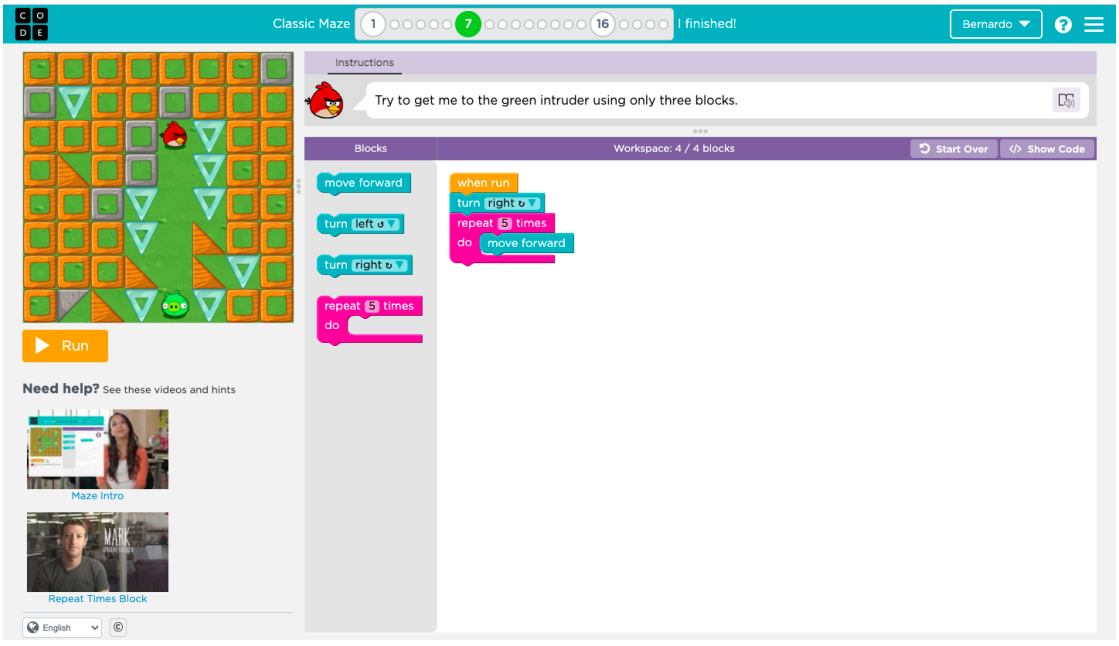
A plataforma, no contexto dos cursos completos, permite utilizadores de dois tipos: alunos e professores. Estes cursos, do ponto de vista do professor, consistem em matéria pronta a lecionar, apoio detalhado de como organizar e expor cada aula e exercicios para testar os alunos. A plataforma está desenhada para um professor poder criar turmas de alunos, atribuir aulas a turmas e aceder ao progresso dos seus alunos. Para um utilizador aluno, a plataforma consiste numa ferramenta apelativa para aprender. Os cursos consistem não só em actividades de programação por blocos, mas também em apresentações de conceitos de informática e exercicios de escrita e/ou escolha múltipla. Na Figura 2.4 é apresentado parte de uma unidade de um curso, organizada por capítulos que contêm lições, de forma a mostrar actividades presentes numa lição. Cada uma destas actividades ou lições pode ser composta por uma tarefa única ou uma sequência de tarefas.

Figura 2.4: Exemplo de actividades em capítulos num curso no *Code.org*



A plataforma também oferece, fora do contexto dos cursos, pequenas aulas com duração estimada para uma hora designadas *Hour of Code*. Estas actividades do *Hour of Code* consistem numa pequena explicação em formato video ou textual seguido de um conjunto de tarefas de programação. A Figura 2.5 apresenta uma actividade em progresso do *Hour of Code*. No *layout* conseguimos observar duas áreas principais: à esquerda temos a área de execução e à direita a área de trabalho. A área de execução permite ao utilizador testar o seu trabalho, refletindo as alterações na área de trabalho. A área de trabalho consiste num espaço onde o utilizador pode arrastar blocos de forma a compor funcionalidades do programa, juntamente com uma zona que contém os blocos permitidos para a actividade. Na Figura 2.5 está também presente a *Navbar* da plataforma que, para além de permitir sair da actividade para navegar na plataforma, permite também navegar sobre as tarefas da própria actividade através do elemento central presente na *Navbar*. Algumas das actividades na plataforma utilizam programação por blocos apesar de não estar limitado a programação por blocos. Na plataforma, a criação de actividades de programação está limitada à entidade gestora da plataforma.

Figura 2.5: Exemplo de uma actividade de programação por blocos do *Hour of Code* no *Code.org*

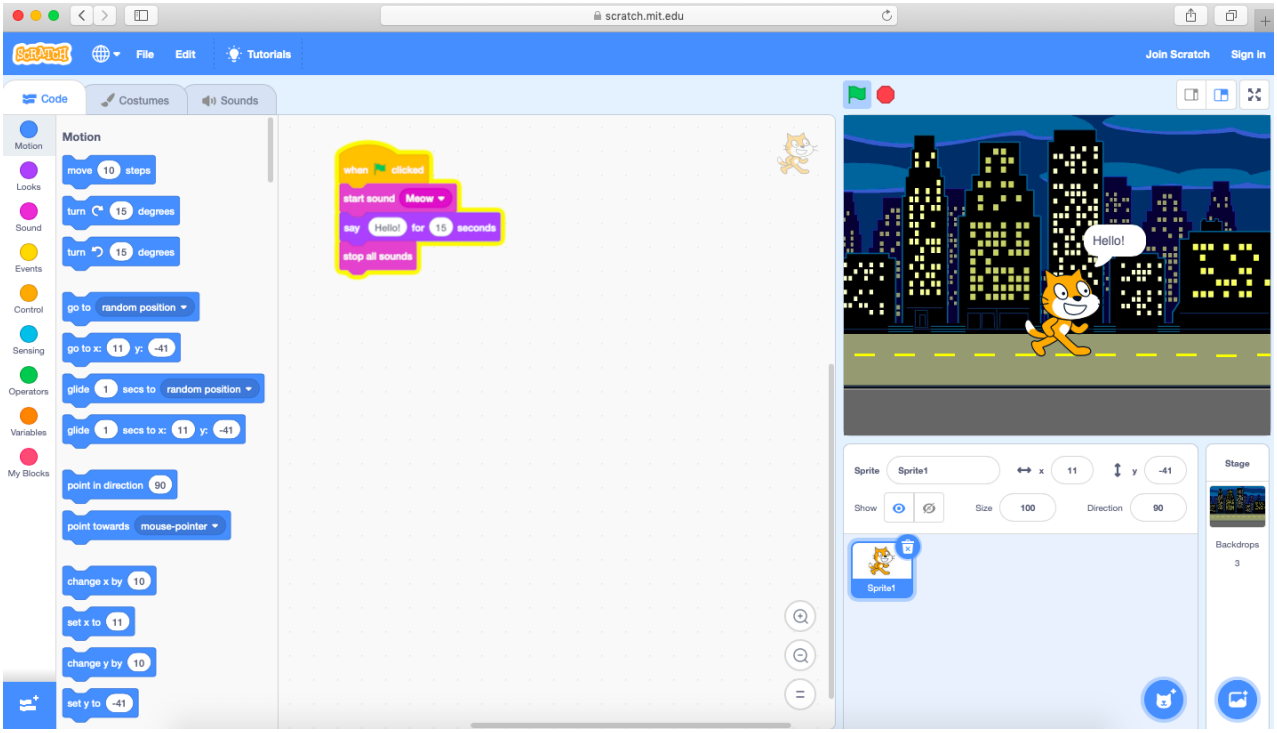


2.2.2 Scratch

O *Scratch* [23] é uma plataforma de actividades de programação por blocos desenvolvida pelo MIT. As actividades de programação, no contexto desta plataforma, consistem em criar projectos através de animações de elementos visuais usando blocos de instruções. Estes projectos podem ser desde histórias a jogos e, quando o utilizador termina o seu,

pode publicá-lo, ficando acessível a todos os utilizadores. Um projecto publicado pode ser executado, comentado e inspecionado por qualquer utilizador da plataforma. Apesar do formato destes projectos indicar uma utilização da plataforma mais independente, a plataforma fornece um guia para educadores que desejem utilizar em contexto de aula. A criação de actividades de programação na plataforma existe sob um formato menos restrito, onde o objectivo de cada actividade é ditado pelo utilizador ou, no contexto de aula, pelo professor, sem intervenção da plataforma.

Figura 2.6: Exemplo de uma actividade de programação por blocos no *Scratch*

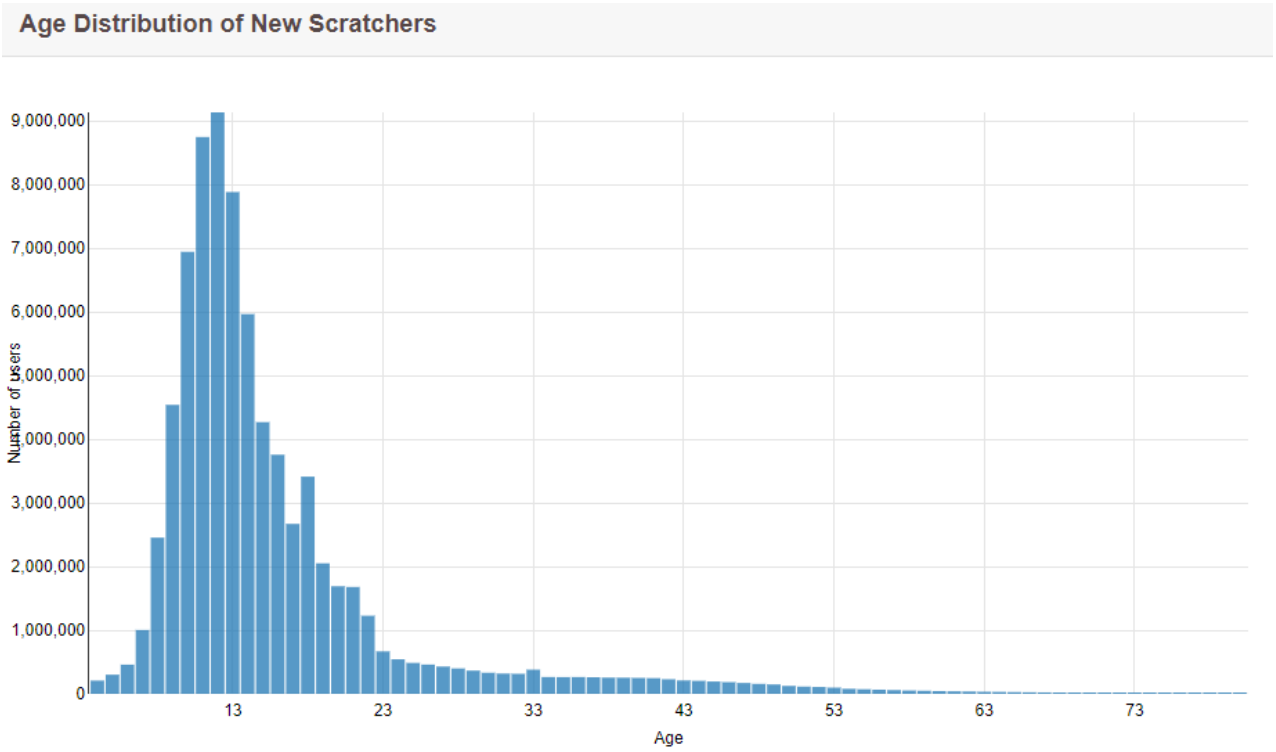


Na Figura 2.6 é apresentado o interface de uma actividade de programação no *Scratch*. No interface está presente, à esquerda, uma *toolbox* com os blocos, organizados por tipo de bloco, para desenvolver o programa. Na *toolbox* é possível criar blocos, apesar da funcionalidade ser algo limitada, sendo possível apenas utilizar valores para serem encaixados no bloco criado. Acima da *toolbox* encontram-se *tabs* que permitem aceder a controlos de som e trajes para os *sprites*. Os *sprites* consistem em elementos gráficos que podem ser controlados de forma independente, exemplificados na Figura 2.6 pela personagem que aparece no lado direito da Figura. A área central é onde se desenvolve o programa, arrastando os blocos e encadeando-os para obter as funcionalidades desejadas. Na direita, na parte superior, encontra-se a área de execução, onde se pode testar o programa, e uma área de controlo dos *sprites* e fundos, na parte inferior. O exemplo apresentado na figura representa uma simples animação de um *sprite* a mostrar uma mensagem, usando o bloco 'say x for y seconds' que faz aparecer uma bolha de mensagem com o conteúdo de 'x'

durante 'y' segundos. No exemplo são também utilizados dois blocos do tipo 'som', o primeiro que activa um som antes da mensagem e o segundo que desactiva o som.

Determinadas características da plataforma como o esquema de cores e o léxico usado são indicadores do tipo de utilizadores para o qual a plataforma foi criada. O *Scratch* está desenhado para utilizadores alvo entre os 8 e os 16 anos [22]. Apesar de se verificar uma maioria dos utilizadores nesta faixa étaria, a plataforma tem utilizadores de todas as idades [31], como se pode verificar no gráfico apresentado na Figura 2.7.

Figura 2.7: Distribuição de idades na plataforma *Scratch*



2.2.3 Snap!

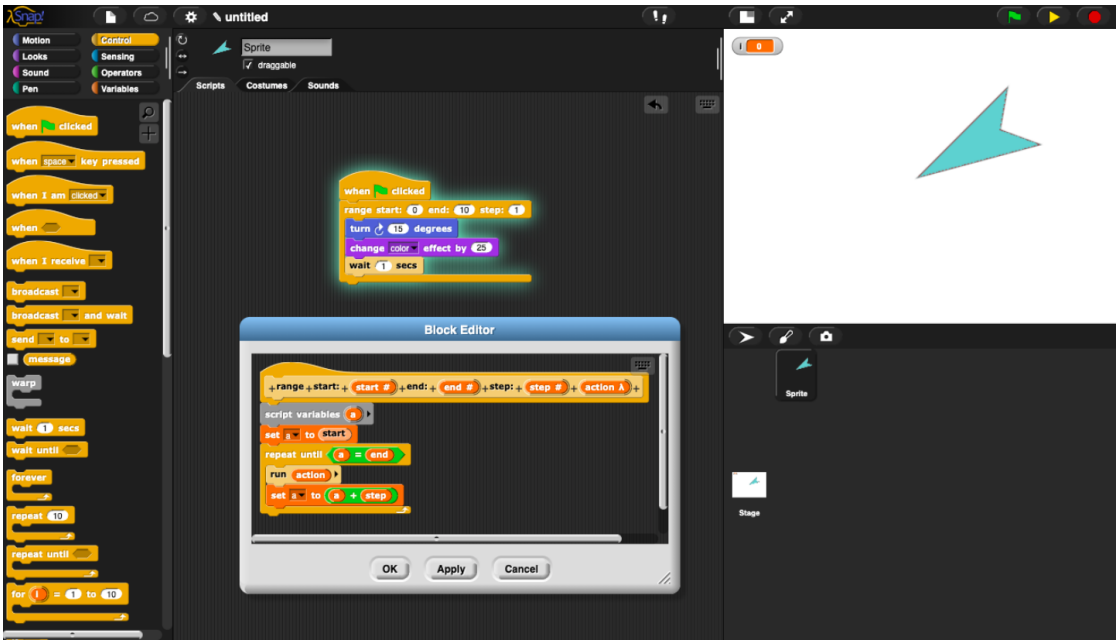
O *Snap!* [2] é uma plataforma de actividades de programação por blocos desenvolvida pela UC Berkley. É uma plataforma que surgiu inicialmente como uma reimplementação do *Scratch* que permite criar os próprios blocos. Esta funcionalidade foi acrescentada no *Scratch* posteriormente. A criação de blocos no *Snap!* continua a ser uma funcionalidade mais completa, permitindo criar qualquer tipo de blocos, em oposição ao *Scratch*. O criador de blocos na plataforma permite que, num bloco criado que tenha encaixes, seja possivel encaixar qualquer tipo de bloco. Esta característica, juntamente com um conjunto de blocos base mais elaborados, é especialmente vantajosa pois permite a implementação de estruturas de dados, como árvores e dicionários, e estruturas de controlo, como um ciclo *for*, ilustrado na Figura 2.8. Estas características foram implementadas com

objectivo de criar uma plataforma para introduzir programação a alunos do secundário e universidade. O objectivo das actividades de programação é feita fora do contexto da plataforma, isto é, uma actividade de programação está concluida quando o utilizador assim o entender.

Figura 2.8: Exemplo de criação de um bloco *for* no *Snap!*

A interface do *Snap!*, ilustrado na Figura 2.9, em comparação ao *Scratch*, é menos chamativo, utilizando um esquema mais sóbrio de cores, apesar do *layout* ser semelhante. Na figura é possível observar semelhanças no *layout* entre o *Scratch*. Elementos como a *toolbox*, área de trabalho e área de execução têm aparência e posições muito semelhantes. Na figura é exibido também a janela de criação de blocos, na parte inferior central, onde se observa a definição do bloco utilizado como exemplo na área de trabalho.

Figura 2.9: Exemplo de uma actividade de programação por blocos no *Snap!*



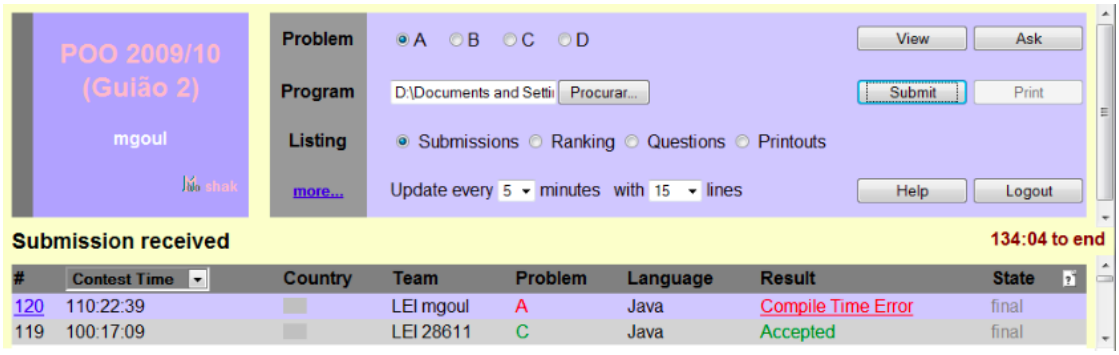
2.2.4 Mooshak

O *Mooshak* é um sistema desenhado para servir de ferramenta a concursos de programação [13]. É um sistema que tem por objectivo avaliar soluções submetidas de acordo

com *inputs/outputs* fornecidos préviamente, aquando a criação do problema. No contexto desta plataforma, pode-se considerar cada problema uma actividade de programação. A plataforma, sendo criada no contexto de concursos de programação, tem um foco sobre *ranks*, tempos de submissão, submissões aceites e número de submissões realizadas. A utilização deste sistema num contexto de ensino ajuda os professores na correcção de trabalhos e alunos na submissão de trabalhos, de forma a receberem *feedback* em relação às soluções submetidas.

Na Figura 2.10 é ilustrado a interface do sistema na perspectiva de participante do concurso. Na parte superior da figura observa-se dados relativos ao concurso como o nome do concurso, à esquerda, e um painel à direita onde se trata de submissões, onde se seleciona o problema e se pode ver os enunciados dos problemas do concurso. A parte inferior da figura consiste numa listagem de submissões realizadas e o *feedback* relacionado com a avaliação da solução, onde se pode observar que uma das submissões foi aceite (o programa foi executado dentro das características programadas e a solução passou nos testes efectuados) e a outra acusou *Compile time error* (o programa teve um erro durante a fase de compilação). O interface do sistema, ilustrado na Figura, é focado mais em apresentar toda a informação pelo que, em comparação com outros ambientes, não é muito apelativo.

Figura 2.10: Exemplo de um concurso de programação no *Mooshak* [15]



A criação de concursos de programação é um processo simples e rápido. No sistema basta criar uma nova pasta dentro da pasta de concursos, que terá todos os dados relativos ao concurso que queremos criar [20]. Ao selecionar a pasta, podem ser definidos os seguintes atributos: Descrição do concurso, quem orgraniza o concurso, e-mail do organizador, dia e hora de inicio e final do concurso. A criação de problemas dentro de um concurso é um processo mais complexo que o processo da criação de um concurso. Existem duas formas de criar um problema [21]. A primeira consiste na importação de um ficheiro **.tar* com a definição do problema que pode ser obtido através da funcionalidade de exportação implementada no sistema. A segunda forma consiste na definição manual do problema, que implica adicionar uma pasta com o nome do problema, definir os atributos do problema (formulário ilustrado na Figura 2.11) e o conjunto de testes

para o problema. Para adicionar um conjunto de testes, seleciona-se a pasta de testes que se encontra dentro da pasta do problema e cria-se uma pasta nova para cada teste do problema.

Figura 2.11: Exemplo dos atributos de um problema de um concurso no *Mooshak* [19]

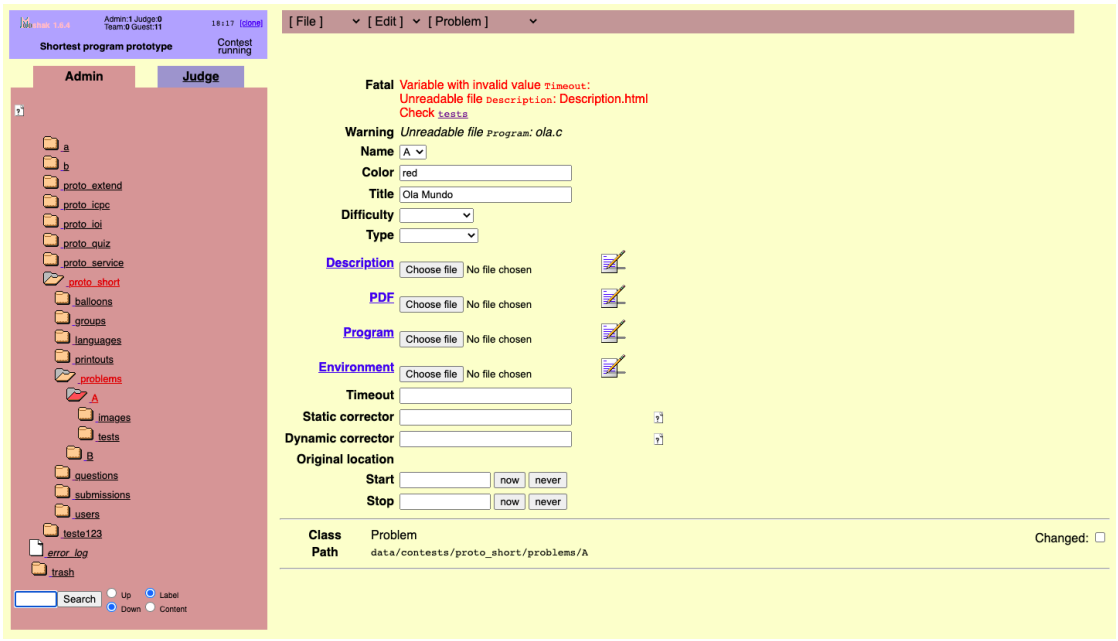
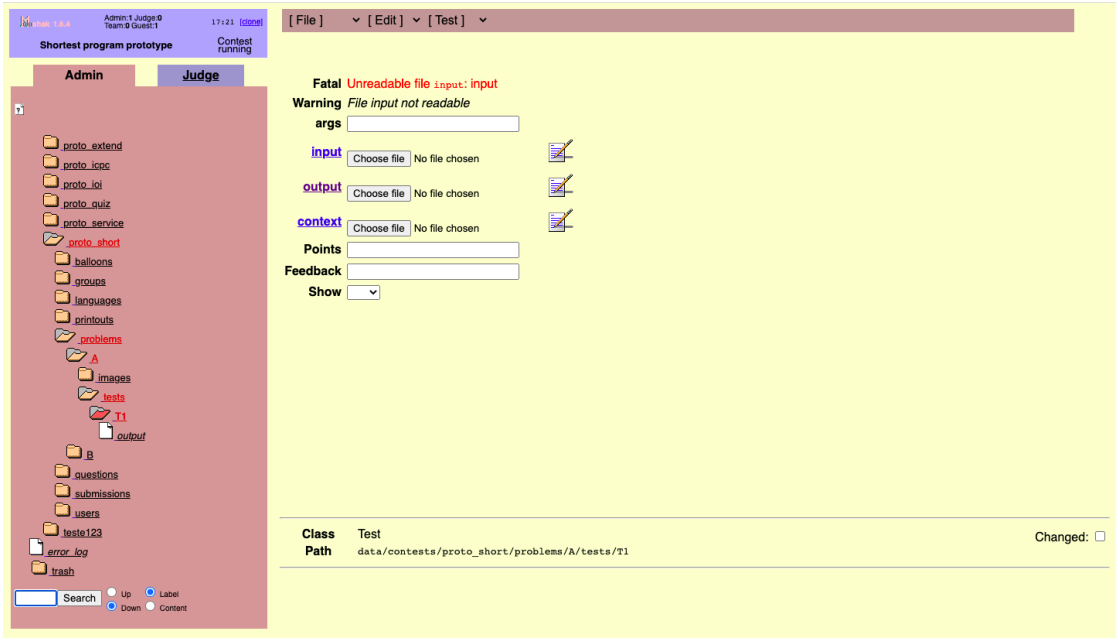


Figura 2.12: Exemplo de um teste de um problema de um concurso no *Mooshak* [19]



Na Figura 2.12 é ilustrado o interface de definição de um teste onde se pode acrescentar, por exemplo, os argumentos de execução de um programa, o ficheiro de *input* e o ficheiro de *output* do teste.

2.2.5 CodeRunner

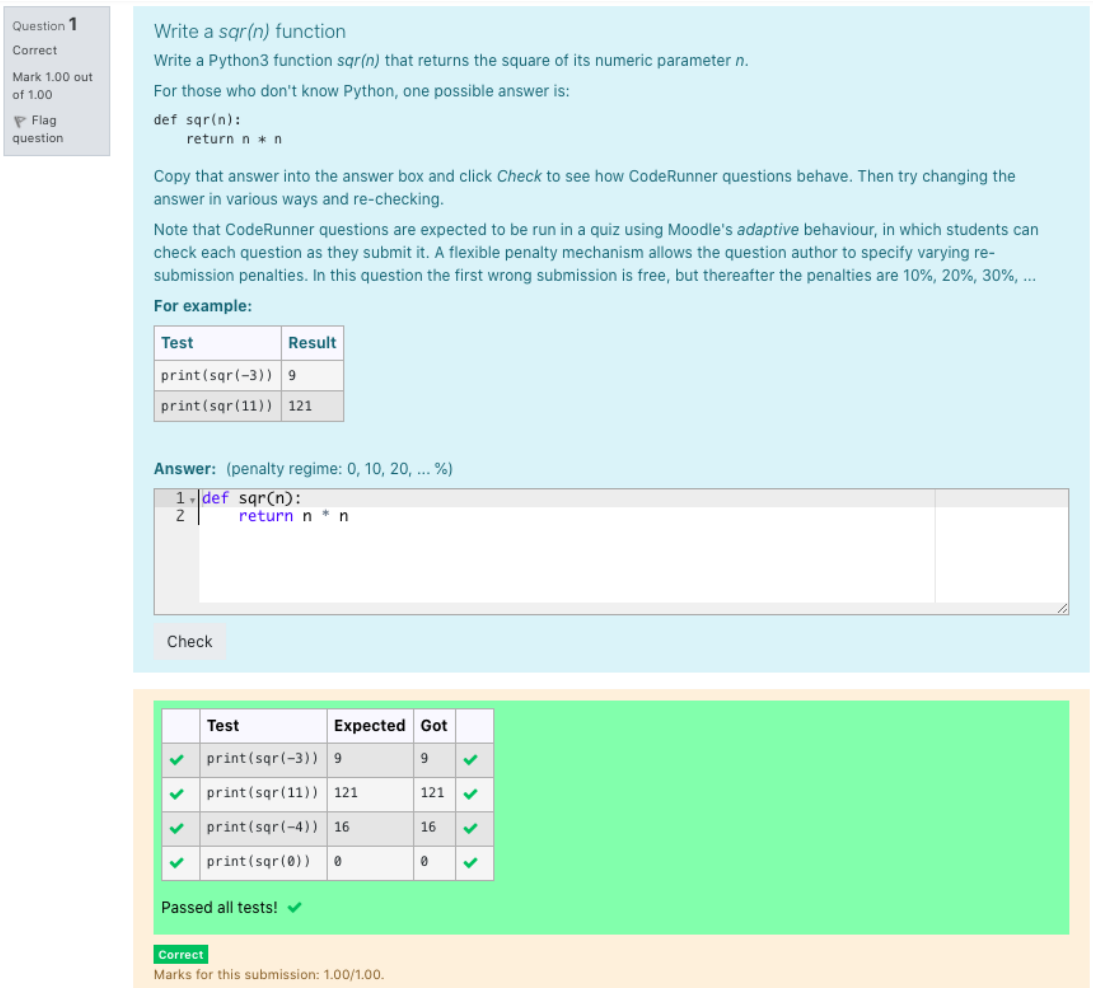
O *Moodle* [1], "*Modular Object-Oriented Dynamic Learning Environment*", é uma plataforma de ensino desenhada para proporcionar a professores e alunos um sistema robusto, seguro e integrado para criar ambientes de aprendizagem personalizáveis. Para permitir a criação destes ambientes personalizáveis, a plataforma permite a integração de extensões. Esta plataforma pode ser usada como ferramenta de apoio a disciplinas ou cadeiras pois permite não só colocar ficheiros de texto, imagem ou video com material de estudo, mas também permite a criação de quizzes. No contexto da plataforma, um quiz é um conjunto de questões, em que cada questão tem um enunciado, uma resposta esperada e uma pontuação associada por ser completada.

O *CodeRunner* [8] é uma extensão para o *Moodle* que permite ter perguntas de programação, num quiz, de forma a avaliar soluções de execução de código. O *CodeRunner* tem suporte para diferentes linguagens de programação como Python2, Python3, C, C++, Java, PHP, JavaScript e Octave. A extensão pode ser instalada em qualquer sistema *Moodle* (versão 2.9 ou superior) que opere em *Linux*, *Windows* ou *MacOS* e, as execuções de código são corridas numa máquina virtual (*Jobe server*) por questões de segurança.

Na Figura 2.13 é apresentada uma questão exemplo, de um quiz, que utiliza a extensão. O enunciado da questão é o primeiro elemento apresentado, sendo permitido adicionar texto, imagens, video, tabelas, entre outros elementos. O utilizador escreve a solução à questão, na linguagem de programação indicada no enunciado, utilizando o editor de código que se encontra no final do enunciado, juntamente com um botão para enviar a resposta. Ao enviar a resposta, o código é compilado, executado numa máquina virtual e comparado com um conjunto de *input/output* esperado. A parte inferior da imagem contém o *feedback* da avaliação da solução, que depende dos testes inseridos durante a criação da pergunta. Este *feedback* apresenta os testes realizados, o *output* esperado, o *output* executado pelo código submetido e se a comparação de *outputs* é igual. A extensão permite ainda que a pontuação de um aluno possa ser configurada para requerer passar a todos os testes com uma pequena penalização de tentativas, ou ser pontuado por cada teste passado.

A criação das actividades de programação utilizando esta extensão [18] envolve a criação de questões do tipo *CodeRunner question*, acrescentado durante a instalação do *CodeRunner* no *Moodle*. Este tipo de questões tem, para além das opções comuns a outras questões do *Moodle* como titulo da pergunta e enunciado como é possível observar na Figura 2.14, opções específicas para actividades de programação. Nestas opções, ilustradas na Figura 2.15, é onde podemos configurar, por exemplo, o tipo de linguagem de programação esperada na resposta, o número máximo de linhas de código permitidas

Figura 2.13: Exemplo de uma pergunta de quiz utilizando o *CodeRunner* [7]



e penalizações de tentativas incorrectas. Num último separador encontramos as configurações para os conjuntos de teste, apresentado na Figura 2.16, onde se compõem os testes com o *input/output* esperado. No final da Figura 2.16 observam-se ainda algumas opções como, por exemplo, utilizar o teste como exemplo no enunciado e atribuição de pontuação individual do teste.

2.2.6 Virtual programming lab

O *virtual programming lab* [17] é também uma extensão para o *Moodle* desenhado para gerir trabalhos de programação no contexto do ensino e permitir criar actividades de programação. Entre as características principais desta extensão destacam-se [29] disponibilizar um ambiente que permite aos alunos editar o seu código fonte no *browser*; permitir a colocação de restrições à edição; permitir impedir *copy-paste* de fontes externas; permitir aos alunos executar os seus programas no *browser*; permitir aos alunos e professores executar testes para avaliar as soluções do código; permitir procurar semelhanças entre

Figura 2.14: Separador *General* da criação de perguntas com o *CodeRunner*

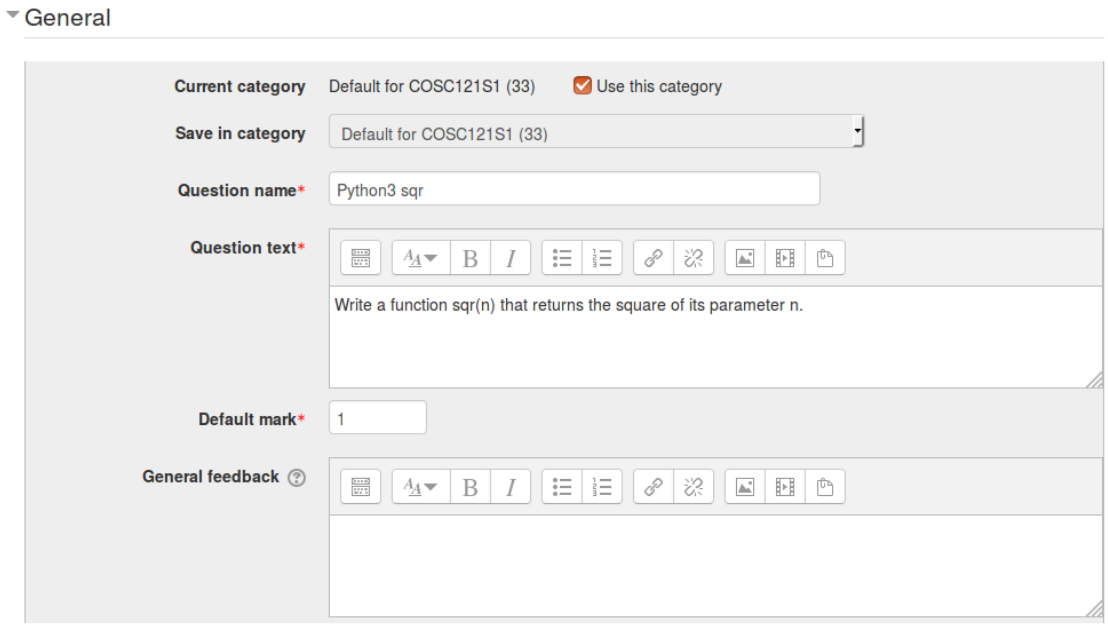
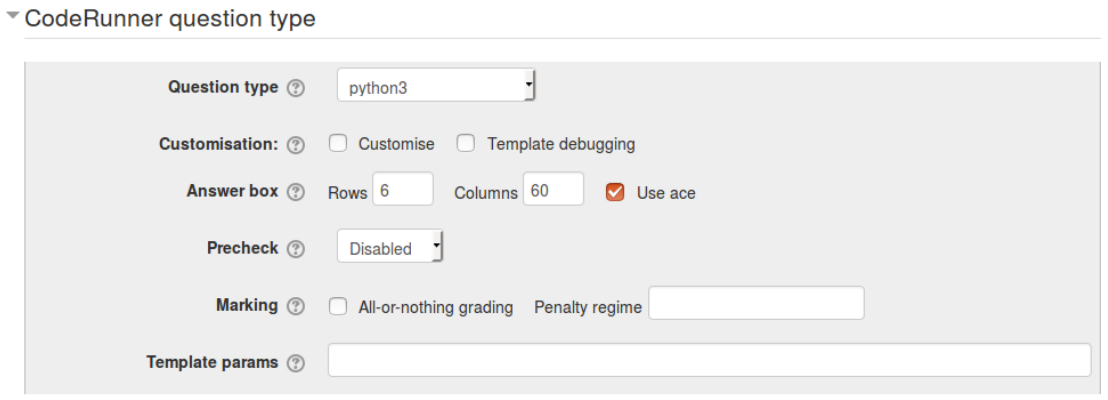


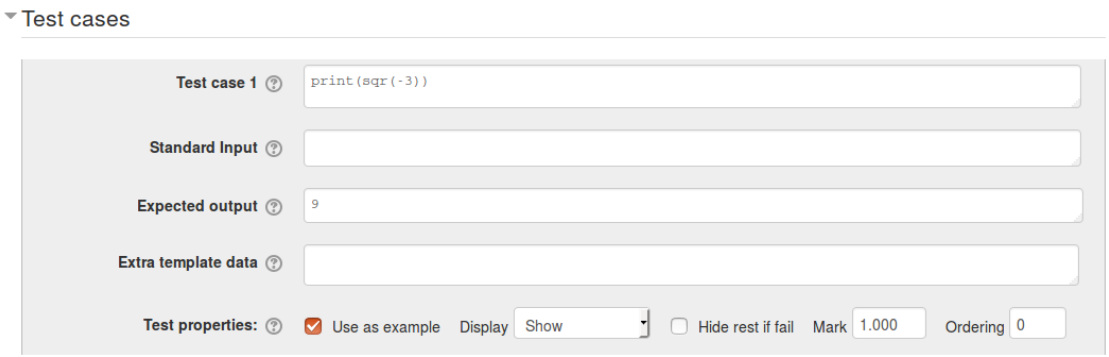
Figura 2.15: Separador *CodeRunner question type* da criação de perguntas com o *CodeRunner*



ficheiros; ter suporte para linguagens de programação como: Ada, C, C++, C#, Fortran, Haskell, Java, Matlab/Octave, Pascal, Perl, PHP, Prolog, Python, Ruby, Scheme, SQL e VHDL.

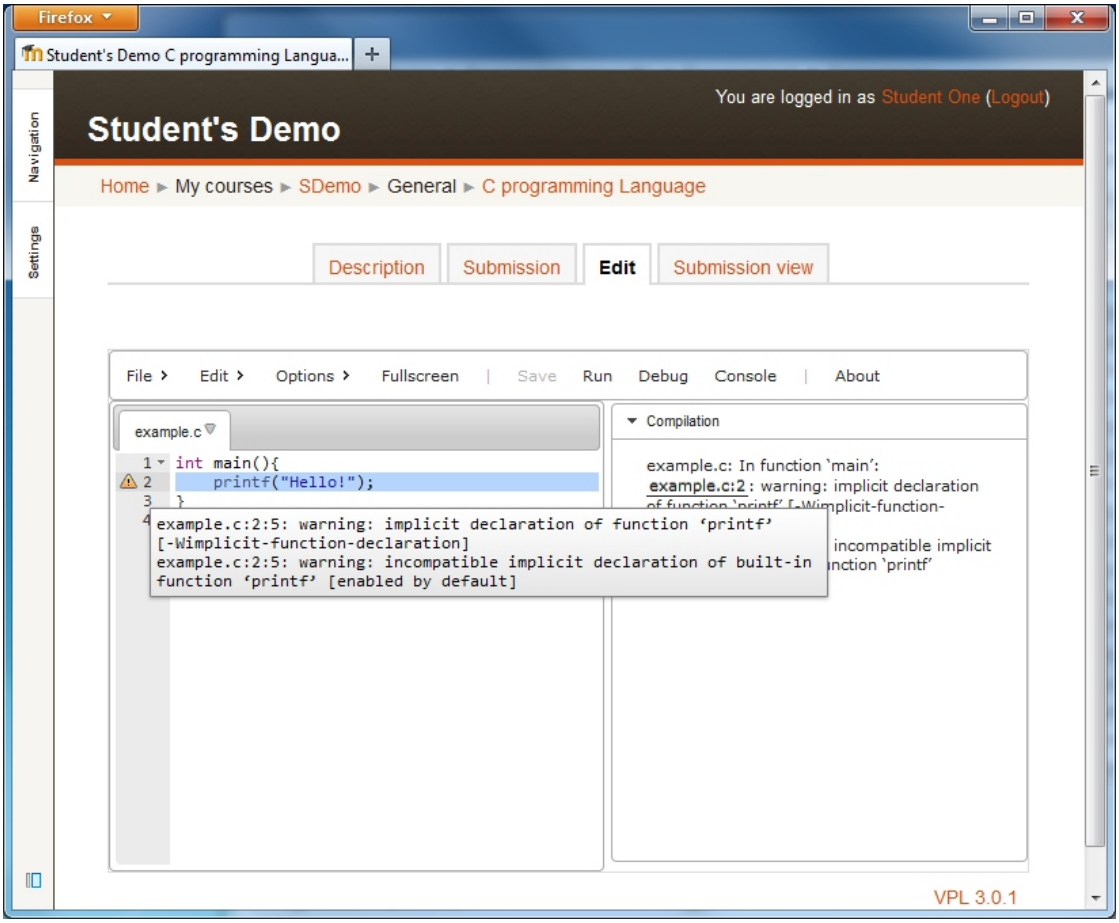
Na Figura 2.17 é apresentado um exemplo do interface disponibilizado pela extensão para realização de exercícios de programação. O separador activo mostra o interface de edição dos ficheiros submetidos, que permite modificar e executar o programa submetido. O ambiente de edição tem um módulo que ajuda a detectar erros de sintaxe [3], e uma área que mostra o resultado do compilador. Nos outros separadores é possível consultar a descrição da actividade, submissão de ficheiros e consulta de informação sobre a

Figura 2.16: Separador *Test cases* da criação de perguntas com o *CodeRunner*



submissão.

Figura 2.17: Exemplo do ambiente *Virtual Programming lab*



Quando esta extensão é instalada, é adicionado ao *Moodle* uma nova opção de criação de uma actividade VPL [28], desta forma permitindo a criação de actividades de programação. A criação de uma actividade é realizada através de um formulário onde existem

campos, por exemplo, para nome, descrição e prazo para a actividade, assim como outras opções relativas à pontuação e ficheiros de submissão. A avaliação das soluções propostas pode ser composta através de dois métodos, o primeiro é um conjunto de *input/output* esperado, que é colocado num ficheiro que a extensão utiliza para comparação, e o segundo método é utilizar *scripts* para avaliar as submissões. Este segundo método é mais complexo mas permite uma maior variedade de testes.

2.2.7 Análise comparativa

Tendo em consideração o nosso objetivo (plataforma de criação de actividades de programação por blocos e ambiente de execução destas actividades que seja apelativo e promova a aprendizagem) existem algumas características importantes que a nossa plataforma deve ter: criação de actividades de programação, execução dessas actividades utilizando programação por blocos e ambos os ambientes devem ser fáceis de usar e apelativos.

De forma a analisar algumas das características dos ambientes de ensino estudados, foi feita uma tabela comparativa com algumas características importantes, ilustradas na Tabela 2.1. Em seguimento do estudo e análise das características dos vários ambientes dedicados ao ensino, foram seleccionadas algumas características importantes a incorporar na aplicação como a possibilidade de criação de actividades, a utilização de linguagem por blocos, a possibilidade de edição de código na plataforma e apresentação de *feedback* das soluções apresentadas. As funcionalidades do *Code.org* relativas aos utilizadores, como criar turmas e atribuir actividades, enquadram-se perfeitamente no contexto da aplicação da dissertação, podendo ser considerado como uma boa base para a aplicação. Em relação ao interface de execução de actividades e interface de criação de actividades, no caso dos ambientes que permitem criação de actividades, cada uma dos ambientes com excepção do *Mooshak*, tem alguma consideração sobre a simplicidade, intuitividade e apelo. Quanto aos interfaces, o *Mooshak* tem um maior foco sobre as funcionalidades necessárias a apresentar. A criação de actividades é uma característica importante, porém, os ambientes analisados que permitem a criação de actividades de programação não têm suporte para linguagens de programação por blocos. Os ambientes com suporte para actividades de programação por blocos analisados não permitem a criação de actividades.

2.3 Bibliotecas de programação por blocos

Uma biblioteca de programação consiste numa colecção de recursos utilizados por programas. Estes recursos podem ser: dados de configuração, documentação, sub-rotinas, classes, valores, tipos, entre outros. Nesta secção, serão apresentadas bibliotecas de programação por blocos com objectivo de servir de base de implementação na aplicação da dissertação.

Tabela 2.1: Análise comparativa de ambientes dedicados ao ensino

Características	Code.org	Scratch	Snap!	Mooshak	CodeRunner	VPL
Criação de actividades	Não	Não	Não	Sim	Sim	Sim
Actividades de programação por blocos	Sim	Sim	Sim	Não	Não	Não
Ambiente para edição de código	Sim	Sim	Sim	Não	Sim	Sim
Feedback da solução apresentada	Sim	Não	Não	Sim	Sim	Sim

2.3.1 Google Blockly

O *Google Blockly* [5] é uma biblioteca *open source*, desenvolvida pela *Google*, que fornece um ambiente para criar e executar programas utilizando blocos. Este ambiente permite também criar e editar blocos, assim como converter o código associado a um conjunto de blocos em linguagens de programação como *Javascript*, *Python*, *Lua* ou *PHP*. O *Google Blockly* não é utilizado como aplicação final, mas sim como suporte ao desenvolvimento de aplicações que tencionem utilizar programação por blocos. Esta biblioteca é utilizada em algumas aplicações como *Blockly Games*, *Code.org*, *App Inventor*, entre outras.

A utilização desta biblioteca para construção de aplicações pode ser abordada de duas formas distintas. A primeira forma consiste construir, de raiz, os elementos da aplicação, utilizando a API da biblioteca. Esta API oferece ao programador os meios para implementar na sua aplicação áreas de trabalho, onde os utilizadores poderão construir os seus programas utilizando os blocos disponibilizados, e executar as instruções das sequências de blocos presentes nestas áreas de trabalho. A biblioteca permite criar blocos personalizados, porém, dispõe ainda de alguns blocos base. A segunda forma de utilização da biblioteca consiste em adaptar os exemplos da aplicação, ilustrados na Figura 2.18, de forma a ajustar-se à aplicação a desenvolver. A Figura 2.18 apresenta exemplos de aplicações, completamente funcionais, desenvolvidos pela equipa do *Google Blockly*, que podem servir como template de forma a incluir as funcionalides da biblioteca na aplicação. Em suma, os pontos fortes do *Blockly* consistem nos seguintes aspectos[11]:

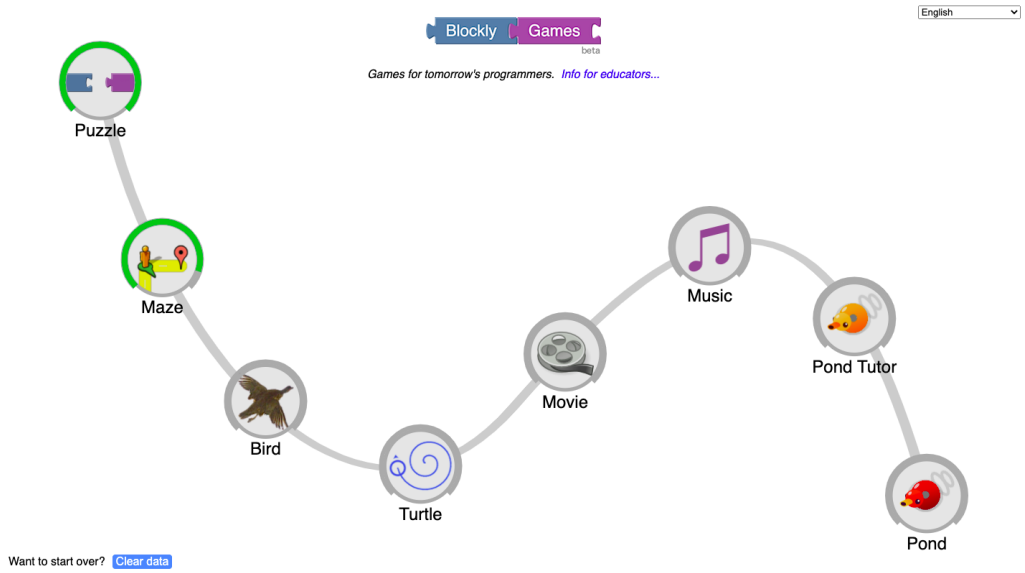
- Open source: A biblioteca é actualizada frequentemente com contribuições relevantes e pode ser clonada e alterada localmente, se necessário.

- Exportável: Os blocos podem ser convertidos noutras linguagens de programação, permitindo aos utilizadores reutilizarem algoritmos que tenham sido criados através de blocos.

- Sem barreira de linguagem: Tem suporte para mais de 40 línguas.

- Extensível: Blocos personalizados podem ser criados de acordo com as necessidades do programa.
- Alta Capacidade: Pode ser utilizado para criar programas complexos como, por exemplo, calcular o desvio padrão de um conjunto de dados.

Figura 2.18: Exemplo do templates para actividades do blockly

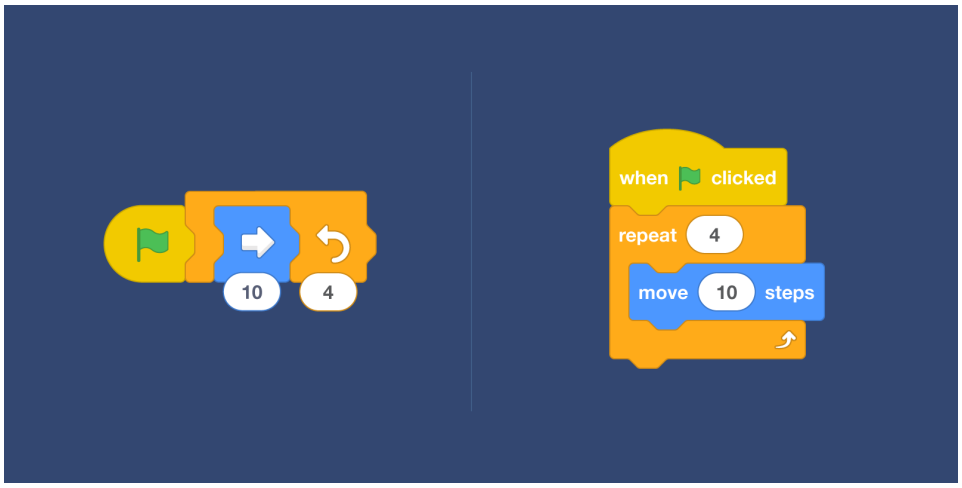


2.3.2 Scratch blocks

Scratch blocks é uma biblioteca, em desenvolvimento, para construção de interfaces de programação por blocos [25]. Esta biblioteca é um clone do *Google Blockly* com algumas alterações, pelo que herda as funcionalidades do *Blockly*. A alteração que se destaca na apresentação desta biblioteca consiste na adição de um tipo de apresentação dos blocos, ilustrado no lado esquerdo da Figura 2.19. Esta apresentação alternativa de blocos tem uma aparência mais compacta, usa ícones nas etiquetas em vez de etiquetas com texto e os blocos encaixam horizontalmente. Estas características deste tipo diferente de blocos têm como objectivo ser mais apelativas e intuitivas para utilizadores alvo mais jovens que os utilizadores frequentes do *Scratch*. Esta biblioteca pode ainda tirar partido do *Scratch-VM* [27], que é uma biblioteca de suporte para representar, executar e manter o estado de programas que utilizem o *Scratch Blocks*, para proporcionar um ambiente estável e consistente para ser utilizado em aplicações que tencionem utilizar programação por blocos. Este projecto é uma colaboração entre a *Google* e a equipa do *Scratch* do MIT [24] e tem como objectivo substituir o *Google Blockly* como padrão para aplicações de programação por blocos. A situação do *Scratch blocks* estar em estado de desenvolvimento

é um argumento desfavorável para a utilização desta biblioteca, pelo que pode não ser tão estável como o *Google Blockly*.

Figura 2.19: Exemplo dos tipos de apresentação de blocos do *Scratch blocks*



2.4 Síntese

Face ao estudo apresentado neste capítulo, podemos constatar que os métodos e ferramentas para o ensino têm sido actualizadas de forma a adoptar os avanços na tecnologia.

Através da análise comparativa dos ambientes estudados, concluiu-se que será importante implementar uma aplicação que adopte uma estrutura que permita criar actividades de programação em blocos, criar conjuntos de alunos, agrupar actividades de programação, atribuir conjuntos de actividades a turmas e realizar actividades de programação na própria aplicação (edição da solução mais execução), para além de oferecer *feedback* da resolução da actividade. Para tal, foi necessário estudar algumas bibliotecas de programação por blocos e escolheu-se o *Google Blockly* por ser uma biblioteca bastante utilizada noutros ambientes. Em relação ao *feedback* da resolução de actividades, o ambiente *CodeRunner* será tomado como um exemplo do tipo e disposição da informação que será relevante para os alunos que executam actividades.

Com este estudo, consideramos que é possível desenvolver uma aplicação que integre ambientes de criação, edição e execução de actividades de programação e ofereça *feedback* relevante sobre as execuções, de forma a oferecer aos utilizadores uma ferramenta apelativa e fácil de usar.

3

Bricks - Um sistema de actividades de programação

Neste capítulo será apresentada a aplicação, juntamente com as suas funcionalidades e interfaces, e modelo de dados do sistema implementado.

3.1 O sistema *Bricks*

A aplicação *Bricks* é um sistema que permite a criação e execução de actividades de programação por blocos e permite a gestão de actividades, turmas de alunos e resoluções de actividades. O objectivo principal da aplicação é ser uma ferramenta de apoio à aprendizagem de programação.

Para garantir o objectivo da aplicação, a natureza de um serviço *web* torna-se uma escolha apropriada permitindo acessos remotos e concorrentes, por parte dos utilizadores, às funcionalidades da aplicação.

De seguida serão caracterizadas as actividades de programação, a linguagem de programação usada na resolução das actividades, e também o editor de resolução de actividades. Finalmente nesta secção são apresentados os diferentes tipos de *feedback* de execuções de actividades que a aplicação fornece.

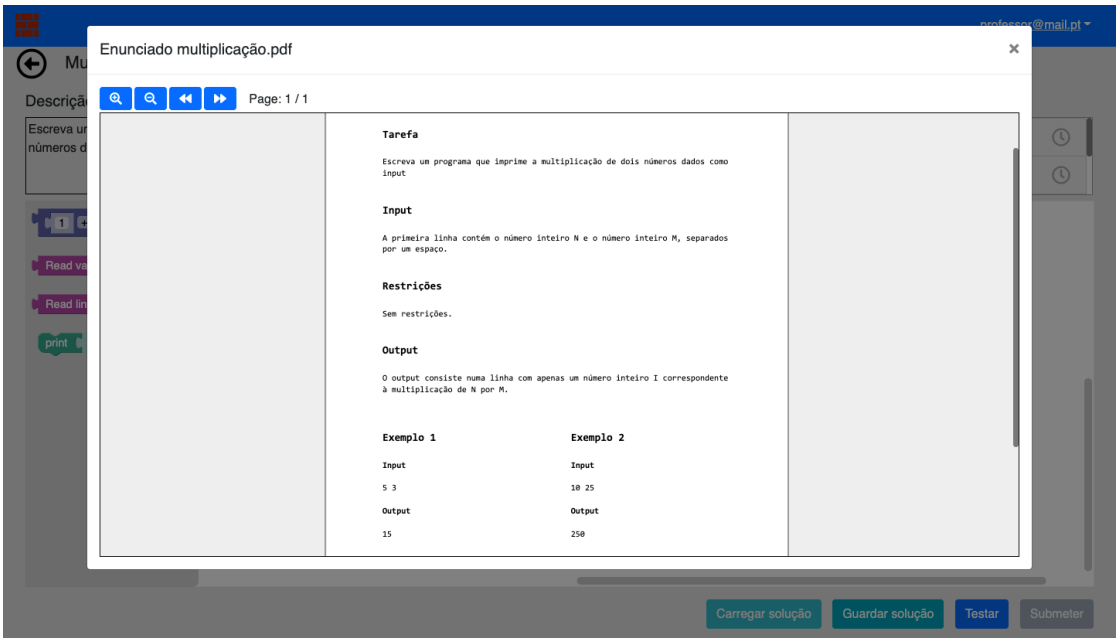
3.1.1 Actividades de programação

Uma actividade de programação, no contexto da aplicação, é um problema que deve ser resolvido através dum programa, utilizando de programação por blocos. Um simples exemplo de uma actividade, na aplicação, seria um problema cuja solução seria um programa que realizasse a multiplicação de dois números. A estrutura de uma actividade de programação pode ser dividida nas seguintes componentes: enunciado, documentação, testes e linguagem.

O enunciado de uma actividade de programação descreve o problema a resolver, especificando claramente o objectivo do programa a implementar e a interacção deste com o utilizador. Este enunciado, na aplicação, é um ficheiro em formato PDF (*Portable*

Document Format). Na Figura 3.1 é exibida a visualização do documento de enunciado da actividade proposta como exemplo (Multiplicação de dois números), onde é possível observar que o conteúdo do ficheiro exprime o objectivo da actividade, juntamente com exemplos de entrada e saída de testes, de forma a guiar a construção do programa de solução.

Figura 3.1: Visualização do enunciado da actividade exemplo (Multiplicação de dois números)



A documentação de uma actividade de programação é constituída por documentos relevantes que sirvam de apoio teórico à execução da actividade. Esta documentação, na aplicação, é composta por um conjunto de ficheiros em formato PDF (*Portable Document Format*) e a inclusão deste(s) ficheiros é opcional. A visualização da documentação, durante a realização de uma actividade, é semelhante à visualização do enunciado da actividade. No seguimento da actividade exemplo (Multiplicação de dois números), um exemplo de documentação poderia ser um documento cujo conteúdo fosse uma explicação do funcionamento dos blocos permitidos pela actividade.

Os testes de uma actividade de programação são a forma de validação da actividade. Cada teste é constituído por um nome, uma entrada e uma saída. A entrada é um ficheiro que contém o que o programa deverá ler de forma a transformar no conteúdo presente no ficheiro de saída. Na aplicação, os ficheiros de entrada e saída têm o formato TXT (*Text file*). O nome do teste é apenas uma *String* que identifica cada par de ficheiros de entrada e saída. Na actividade de exemplo (Multiplicação de dois números), os ficheiros de entrada teriam dois números separados por um espaço e os respectivos ficheiros de saída teriam um número, que seria o resultado da multiplicação dos números presentes no ficheiro de

entrada.

Estas actividades de programação utilizam uma linguagem de programação por blocos. Cada actividade *Bricks* tem associado um conjunto de blocos, com os respectivos limites de utilização, que define a linguagem de programação a utilizar na sua resolução. Durante a realização de uma actividade, apenas os blocos escolhidos na criação são apresentados no editor de programas e, ao passar o cursor sobre cada bloco da *toolbox*, é apresentada uma mensagem indicando o respectivo limite de utilização.

Cada actividade está associada a um ou mais conjuntos de alunos (turmas). A resolução de uma actividade (programa) pode ser guardada, validada de acordo com os testes da actividade e submetida. Uma submissão de uma resolução de uma actividade consiste em guardar a solução e o resultado da validação da solução (saída do programa e comparação com a saída esperada). Após uma submissão de uma resolução de uma actividade, os dados da submissão estão acessíveis ao professor que está encarregue da turma ao qual o aluno que realizou a submissão pertence. No entanto, cada actividade tem associada uma data de início e uma data de fim que determina o período em que é possível submeter um programa nessa actividade. Estas datas determinam o estado que uma actividade pode ter. Logo, uma actividade, num dado momento, pode estar em um dos seguintes estados: "aberta", "encerrada" e "agendada". Uma actividade encontra-se "aberta" enquanto estivermos entre a data de início e a data de fim. Uma actividade está "encerrada" quando já tiver passado a data de fim. Uma actividade fica "agendada" se ainda não tiver chegado a data de início.

3.1.2 Linguagem de programação

Para resolver as actividades, são realizados programas escritos em linguagem por blocos. Os blocos usados são disponibilizados pelo *Google Blockly*.

Cada bloco representa uma instrução ou um conjunto de instruções e tem características visuais como a forma, a cor e a etiqueta. Estas características visuais ajudam o utilizador a compreender como é que os blocos podem ser ligados entre eles e que tipo de valor representa. Para dois blocos poderem ser encaixados, é necessário:

- Formas compatíveis: os blocos têm que ter encaixes que permitam a ligação, como juntar duas peças de um puzzle.
- Tipos compatíveis: algumas ligações necessitam que os blocos sejam de um determinado tipo para a ligação ser permitida. Por exemplo, um bloco para realizar uma soma irá obrigar a que os blocos que irão ser encaixados sejam do tipo "número".

A linguagem de programação por blocos da aplicação é composta por blocos de operações aritméticas, de operações lógicas, de operações de texto, de operações de iterações/ciclos, de operações de manipulação de listas e de operações de variáveis. Em acréscimo, foi necessário implementar uma nova categoria de blocos com o objectivo de ler os valores de entrada de uma actividade.

Os blocos de operações aritméticas são uma categoria que contém todos os blocos relacionados com valores constantes, expressões aritméticas, entre outros. Como exemplos relevantes deste tipo de blocos temos o bloco de valor constante, onde o utilizador pode escrever um número e o bloco representa esse valor constante, o bloco de operação matemática, que consiste num bloco que se pode ser encaixado dois blocos e tem um *dropdown* onde se escolhe o tipo de operação seja soma, multiplicação, entre outros, ou o bloco de operação de agregação, que pode ser encaixado um bloco que contém uma lista com valores e extrai somas, médias, entre outras operações de agregação. Na Figura 3.2 são apresentados os blocos dados como exemplo, onde podemos distinguir o bloco de valor constante, á esquerda, o bloco de operação matemática, ao centro, e o bloco de operação de agregação, à direita.

Figura 3.2: Exemplo de blocos de operações aritméticas.

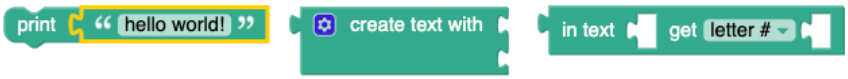
Os blocos de operações lógicas são uma categoria de blocos que contém todos os blocos relacionados às operações com valores booleanos. Nesta categoria podemos encontrar o bloco de valor constante, que pode tomar o valor "true" ou "false", um bloco de operação de condição, que representa uma instrução condicional, e outros blocos de operações associados aos operadores lógicos, como "and", "or" ou "not". A Figura 3.3 apresenta, da esquerda para a direita, o bloco de valor constante, o bloco de operação de condição, o bloco de operador lógico e o bloco de negação lógica.

Figura 3.3: Exemplo de blocos de operações lógicas.

Os blocos de operações de texto são uma categoria de blocos que contém todos os blocos relacionados com operações sobre *Strings*. Nesta categoria enquadram-se blocos como o bloco de valor constante *String*, onde o utilizador pode escrever uma *String* e o bloco representa esse valor constante, o bloco de escrita, que imprime o valor do bloco que estiver ligado ao bloco de escrita, entre outras operações sobre *Strings* como concatenações ou obtenção de um *char* de um determinado índice. Na Figura 3.4 são ilustrados os blocos dados como exemplo, sendo estes, da esquerda para a direita, o bloco de escrita ligado ao

bloco de valor constante com o texto "Hello world!", o bloco de concatenação de texto e o bloco de obtenção de um *char* de um determinado índice.

Figura 3.4: Exemplo de blocos de operações de texto.



Os blocos de operações iterativas/ciclos são uma categoria de blocos que contém todos os blocos relacionados operações de ciclos. Na Figura 3.5 são apresentados exemplos de blocos desta categoria. Da esquerda para a direita, é apresentado o bloco de repetição que repete o conjunto de blocos encaixados no seu interior mediante um bloco de valor constante do tipo "número" que pode ser encaixado no topo do bloco de ciclo, de seguida é apresentado o bloco de repetição que repete o conjunto de blocos encaixados no seu interior dependendo de um valor booleano e, finalmente, o bloco de controlo do ciclo, que pode ser usado para terminar a execução do ciclo ou saltar iterações.

Figura 3.5: Exemplo de blocos de operações de ciclos.

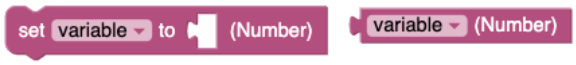
Os blocos de operações de manipulação de listas são uma categoria de blocos que contém todos os blocos relacionados com listas. Esta categoria contém, entre outros, blocos para criar listas e blocos para obter, atribuir ou remover itens das listas. Na Figura 3.6, da esquerda para a direita, são exibidos o bloco de criação de uma lista vazia, o bloco de criação de uma lista com os elementos encaixados na sua sequência, o bloco de obtenção de um item de determinado índice da lista e o bloco de atribuição de um elemento a determinado índice da lista.

Figura 3.6: Exemplo de blocos de operações de listas.

Os blocos de operações de variáveis são uma categoria de blocos que permite manipular e utilizar as variáveis. Os blocos pertencentes a esta categoria são o bloco de atribuição

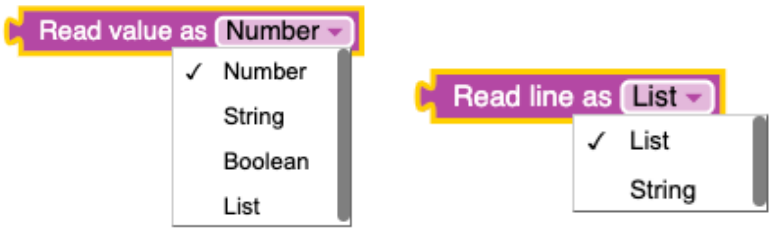
de um valor à variável e o bloco de obtenção do valor da variável. Quando esta categoria é utilizada, o utilizador terá acesso a um botão para criar novas variáveis. A criação de uma nova variável requer ao utilizador um nome e a escolha do tipo que a variável representará. Na Figura 3.7 são apresentados o bloco de atribuição de um valor a uma variável, à esquerda, e o bloco de obtenção do valor presente na variável, à direita.

Figura 3.7: Exemplo de blocos de operações de variáveis.



Para o funcionamento das actividades de programação na aplicação, é necessário que os programas construidos na aplicação suportem operações de leitura. Para tal, foi criada uma nova categoria de blocos, os blocos de operações de leitura, que contém dois blocos para leitura de valores. Estes dois novos blocos são o bloco de leitura de valor e o bloco de leitura de linha, exibidos na Figura 3.8. O bloco de leitura de valor lê individualmente valores de um ficheiro por cada utilização do bloco, sequencialmente, até ao final do ficheiro. O bloco de leitura de linha lê, sequecialmente, as linhas do ficheiro, por cada utilização do bloco, e converte-a no formato escolhido pelo *dropdown*, podendo escolher entre uma lista com os elementos presentes na linha ou uma *String* que contém os elementos da linha separados por um espaço. No contexto da aplicação, os ficheiros que serão lidos por estes blocos serão os ficheiros de entrada dos testes da actividade. A implementação destes blocos será detalhada no capítulo seguinte.

Figura 3.8: Bloco de leitura de próximo valor de entrada (a), à esquerda, e Bloco de leitura de próxima linha de entrada (b), à direita



Finalmente, não sendo inserido em nenhuma categoria, foi criado o bloco "start" com o objectivo de indicar o início do programa.

3.1.3 Editor de programas

Para poder resolver actividades de programação, a aplicação integra o editor de programas do *Google Blockly*. Este editor é composto por duas partes: a *toolbox* e a área de trabalho.

Figura 3.9: Interface do editor de programas

Na Figura 3.9 é apresentado a interface do editor de programas onde se pode observar a *toolbox* à direita, que contém o conjunto de blocos disponíveis para resolver a actividade, e a restante área constitui a área de trabalho.

A construção de programas é realizada arrastando os blocos presentes na *toolbox* para a área de trabalho e encaixando-os. O utilizador, ao passar o ponteiro sobre os blocos na *toolbox*, tem acesso aos limites de utilização do bloco durante a realização da actividade e, quando atingir esse limite, o bloco torna-se inactivo como é possível observar na Figura 3.9 para o caso do bloco "Repeat". A área de trabalho, à medida que o utilizador compõe o programa, aumenta a área disponível para acomodar os blocos. Quando o programa do utilizador ultrapassa a área visível da área de trabalho, o utilizador pode usar as barras horizontais e verticais ou arrastar a área de trabalho para visualizar o programa. Neste editor é possível guardar a solução, testar as soluções de acordo com os testes da actividade e submeter a solução actual.

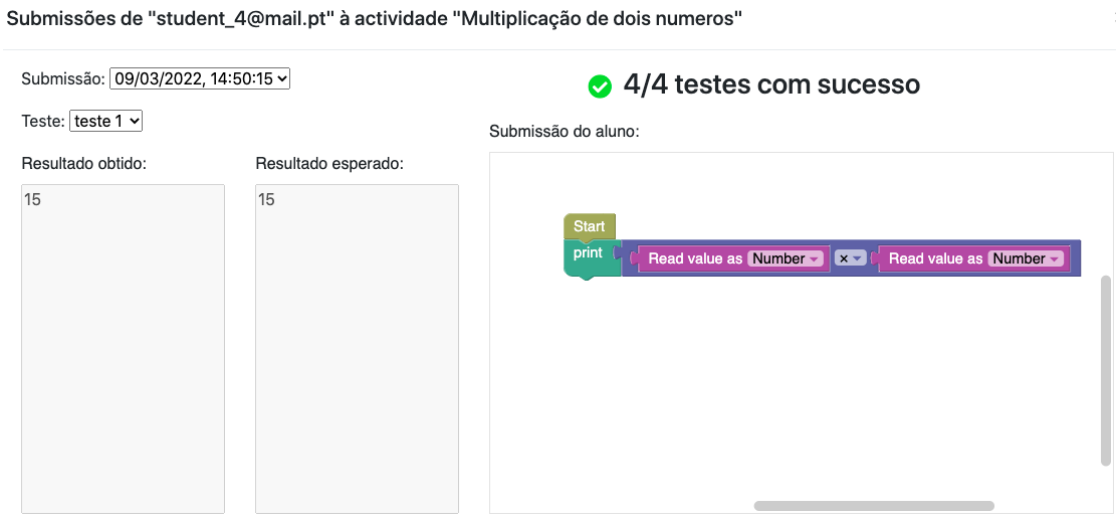
3.1.4 *Feedback* das submissões de actividades

Em acréscimo à criação e execução de actividades de programação, a aplicação permite a visualização de resultados relativos à execução dos programas que resolvem as actividades. O *feedback* das submissões dos alunos pode ser visualizado pelo professor num contexto individual, num contexto colectivo e na visão global das submissões de todas as actividades de uma dada turma.

O *feedback* individual do aluno é dado por cada submissão de um programa a uma actividade. O aluno, ao realizar uma submissão, terá também um *feedback* indicando se a submissão foi aceite ou se ocorreu algum problema. Quando é realizada uma actividade e esta resolução é submetida, existe um *feedback* acerca do número de testes que a solução dada passou (ver Figura 3.10, no topo à direita). Na Figura 3.10 pode ser observado a interface que mostra o *feedback* de uma submissão, exibindo o programa submetido, à

direita, e o resultado dos testes da actividade, à esquerda.

Figura 3.10: Exemplo de resultados de uma submissão de actividade por aluno



O *feedback* colectivo consiste em apresentar uma estatística por cada actividade acerca do estado das últimas submissões dos alunos da turma, relativos a uma actividade, ou seja, a quantidade de submissões completas, parciais e ausência de submissões dos alunos da turma para uma dada actividade. Na Figura 3.11 é apresentado o exemplo do *feedback* colectivo de uma actividade.

Figura 3.11: Exemplo de resultados gerais de uma actividade

Para cada actividade atribuída à turma é possível o professor inspeccionar detalhes de

todas as submissões realizadas pelo aluno. Estes detalhes incluem o programa submetido pelo aluno, o número de testes bem sucedidos e o resultado dos testes da actividade.

A visão global das submissões realizadas é apresentada por turma (ver Figura 3.12). Na turma apresenta-se para cada actividade o resultado da última submissão individual de cada aluno em todas as actividades associadas à turma. Esta visão global da turma é apresentada sob o formato de tabela contendo, para cada coluna excepto a primeira, a referencia à actividade e cada linha contém o identificador do aluno seguido do resultado da última submissão realizada para cada actividade correspondente.

Figura 3.12: Exemplo da visão geral de uma turma

3.2 Funcionalidades da aplicação

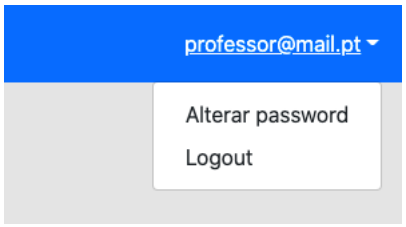
Nesta aplicação existem dois tipos de utilizadores: professor e aluno. Ambos podem registar-se independentemente. No entanto no caso do aluno este poderá ser criado pelo professor. Ambos os utilizadores nas suas interfaces principais tem uma barra superior, onde podem alterar a sua palavra-chave ou terminar sessão, como ilustrado na Figura 3.13. De seguida serão apresentadas as funcionalidades de cada um dos utilizadores.

3.2.1 Funcionalidades do utilizador professor

O utilizador professor tem 2 grandes áreas de interesse, relacionadas respectivamente com actividades e turmas de alunos. As funcionalidades relacionadas com criação de actividades são:

- Criar actividades de programação por blocos: de forma a permitir aos professores transformarem os seus exercícios de programação em actividades da aplicação.

Figura 3.13: Barra superior da aplicação, focado no canto superior direito na zona de operações de conta.



- Visualizar e testar actividades de programação: permitindo ao utilizador verificar se os conteúdos das actividades são apresentados correctamente e se os testes fazem as devidas validações.

- Fazer *download* de actividades de programação: com o objectivo de permitir ao utilizador guardar localmente os ficheiros das actividades e partilhá-los com outros utilizadores professor.

As funções relacionadas com gestão de actividades e turmas são:

- Fazer *upload* de actividades de programação: de forma a permitir utilizar ficheiros de actividade criados por outros utilizadores.

- Criar turmas: permitindo ao utilizador organizar conjuntos de alunos, facilitando a atribuição de actividades.

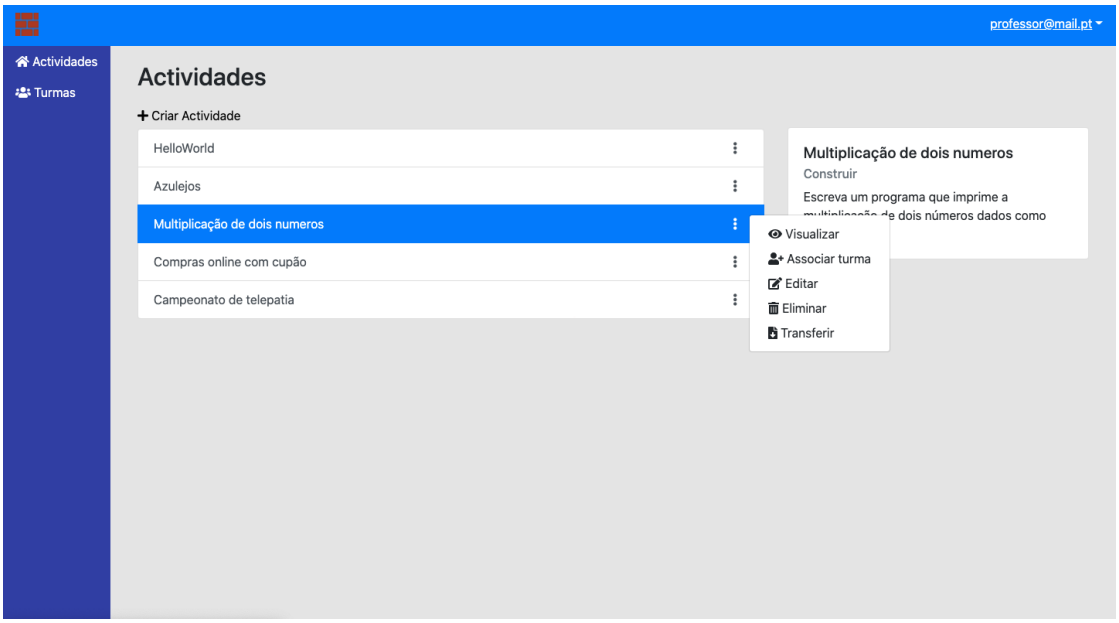
- Atribuir actividades de programação às turmas criadas: desta forma permitindo a possibilidade dos alunos realizarem actividades.

- Obter *feedback* relativo às execuções das actividades nas turmas: fornecendo informação sobre as submissões dos alunos às actividades das turmas do utilizador.

Na Figura 3.14 é apresentado a interface da página principal do professor, onde é possível observar as suas 2 áreas principais: actividades e turmas (na barra de navegação lateral na Figura 3.14). Na área central da página do professor, o utilizador tem uma lista com as actividades criadas por ele. Para visualizar o objectivo da actividade basta seleccionar a actividade e será mostrado do lado direito uma pequena área à direita onde são apresentados alguns detalhes da actividade seleccionada, como o titulo e a descrição da actividade.

Cada actividade nesta lista tem um botão à direita que oferece um conjunto de opções como visualizar a actividade, atribuir actividade a uma turma e fazer *download* da actividade.

Figura 3.14: Interface da página principal do professor



***Download* da actividade**

Na aplicação é possível fazer um *download* da actividade utilizando a opção "transferir"do menu contextual. Nesta funcionalidade é criado um ficheiro de extensão ZIP (arquivo) com a informação/dados da actividade. Este ficheiro tem a seguinte composição:

- Um ficheiro *manifest* que contém a organização do ficheiro de extensão ZIP (arquivo) e os dados gerais sobre a actividade (titulo, descrição, ficheiro do enunciado, ficheiros de documentação, nome e ficheiros de teste).
- Uma pasta ”Resources” com os ficheiros PDF (*Portable Document Format*) de enunciado e documentação.
- Uma pasta ”Tests” com os testes cujo conteúdo tem, por teste, uma pasta com o nome do teste, no seu interior tem duas pastas com o nome *input* e *output* que contêm, respectivamente, os ficheiros de texto para entrada e saída de cada teste.
- Um ficheiro de extensão XML (Extensible Markup Language) com os blocos permitidos para a execução da actividade e os seus respectivos limites. Na Figura 3.15 é apresentado um exemplo destes ficheiros aplicado à actividade ”Multiplicação de dois números”, onde é possível observar que a actividade permite uma utilização do bloco de operação aritmética, limitado a um uso, e do bloco de escrita, sem limite de utilização.

Figura 3.15: Exemplo do ficheiro que contém os blocos permitidos e os seus respectivos limites

```
<xml>

  <category name="Matemática" colour="%{BKY_MATH_HUE}">
    <block type="math_arithmetic" limit="1">
      <value name="A">
        <shadow type="math_number">
          <field name="NUM">1</field>
        </shadow>
      </value>
      <value name="B">
        <shadow type="math_number">
          <field name="NUM">1</field>
        </shadow>
      </value>
    </block>
  </category>

  <category name="Texto" colour="%{BKY_TEXTS_HUE}">
    <block type="text_print" limit="0">
      <value name="TEXT">
        <shadow type="text">
          <field name="TEXT">abc</field>
        </shadow>
      </value>
    </block>
  </category>

</xml>
```

Criação de actividades

Para criar uma nova actividade deve seleccionar no topo da lista de actividades a opção "+Criar actividade".

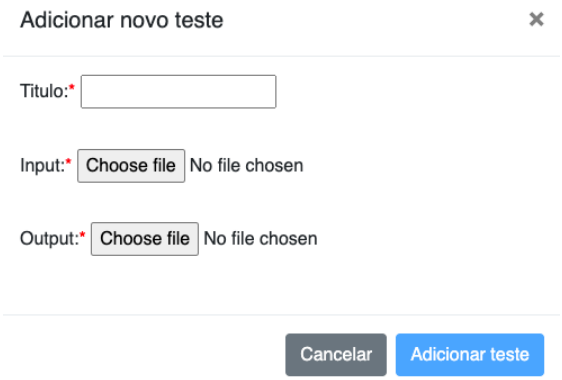
A criação de actividade pode ser realizada de duas formas diferentes: por selecção ou arraste de ficheiro de actividade ou por preenchimento do formulário de criação de actividade (ver Figuras 3.16 e 3.18). Na criação por ficheiro basta arrastar ou seleccionar um arquivo, que contenha uma actividade, previamente criado na aplicação (*download* de actividade).

A criação da atividade é realizada em duas interfaces da aplicação. Na Figura 3.16, o utilizador poderá colocar o título, um enunciado em formato de ficheiro com extensão PDF (*Portable Document Format*), documentação que achar relevante para a actividade em formato PDF (*Portable Document Format*), escolher o tipo de actividade, uma breve descrição e o conjunto de testes para validar a actividade, que são compostos por um titulo, um ficheiro de entrada e um ficheiro de saída, ambos de extensão TXT (*Plain Text*). O processo de introdução de um novo teste pode ser observado na Figura 3.17. A

Figura 3.16: Interface de criação de actividades

Figura 3.18, apresenta a primeira interface para a criação da atividade. Nesta interface é apresentada uma *toolbox* onde o utilizador pode escolher as categorias dos blocos que deseja permitir na execução da actividade e, ao seleccionar a categoria, os blocos dessa categoria serão adicionados à lista à direita, onde o professor pode seleccionar e limitar o número de blocos que poderão ser utilizados das categorias seleccionadas.

Figura 3.17: Interface de criação de um novo teste para a actividade



Criação de turmas

A criação de turma consiste em agrupar conjuntos de alunos. O conceito de turma torna-se importante por uma questão de organização, facilitando um professor que queira atribuir actividades ou consultar resultados.

Na Figura 3.19 é apresentado a interface onde o professor pode criar e consultar as

Figura 3.18: Interface de criação de actividades

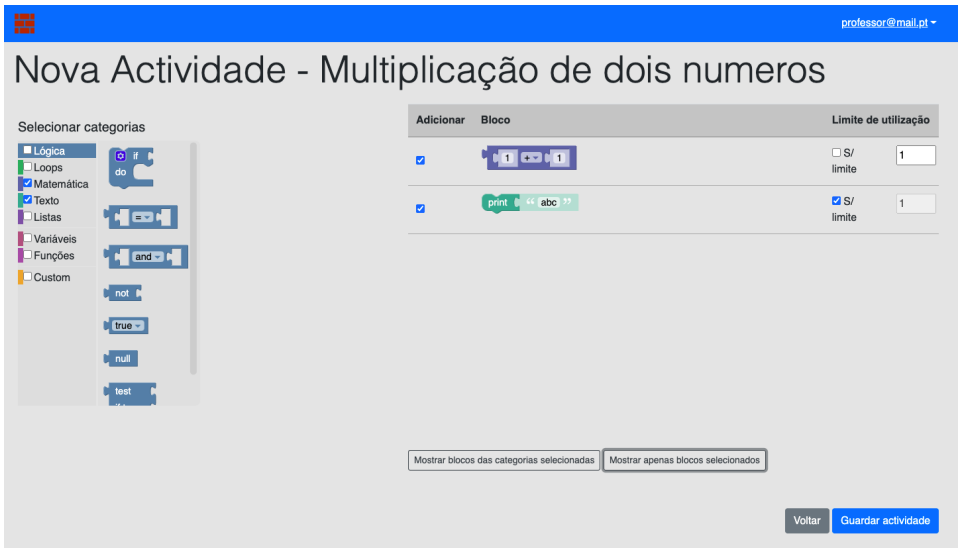
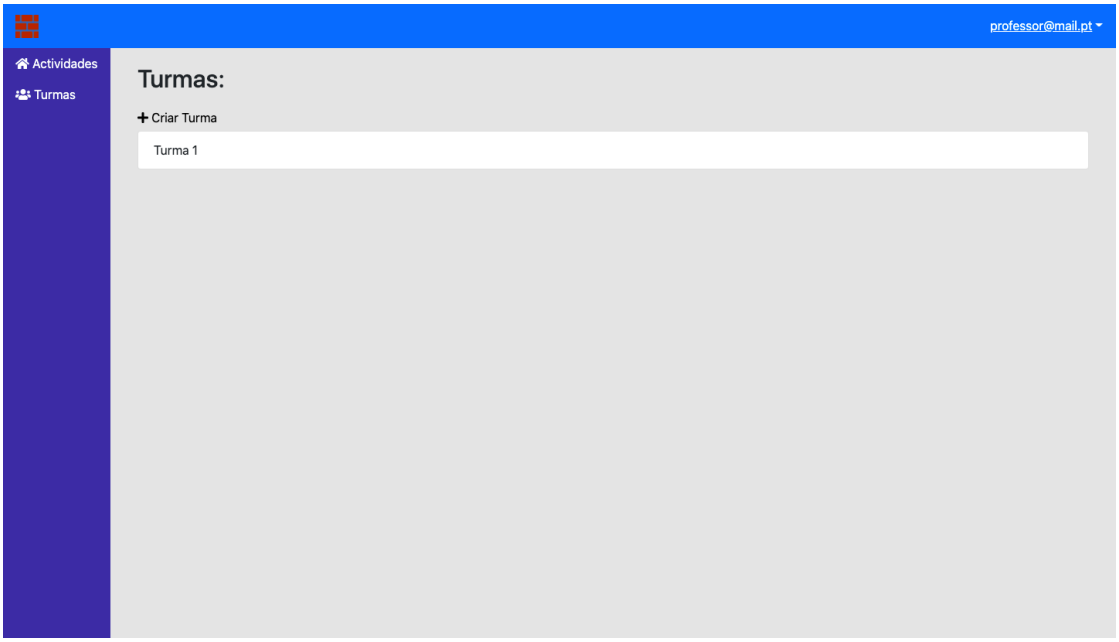
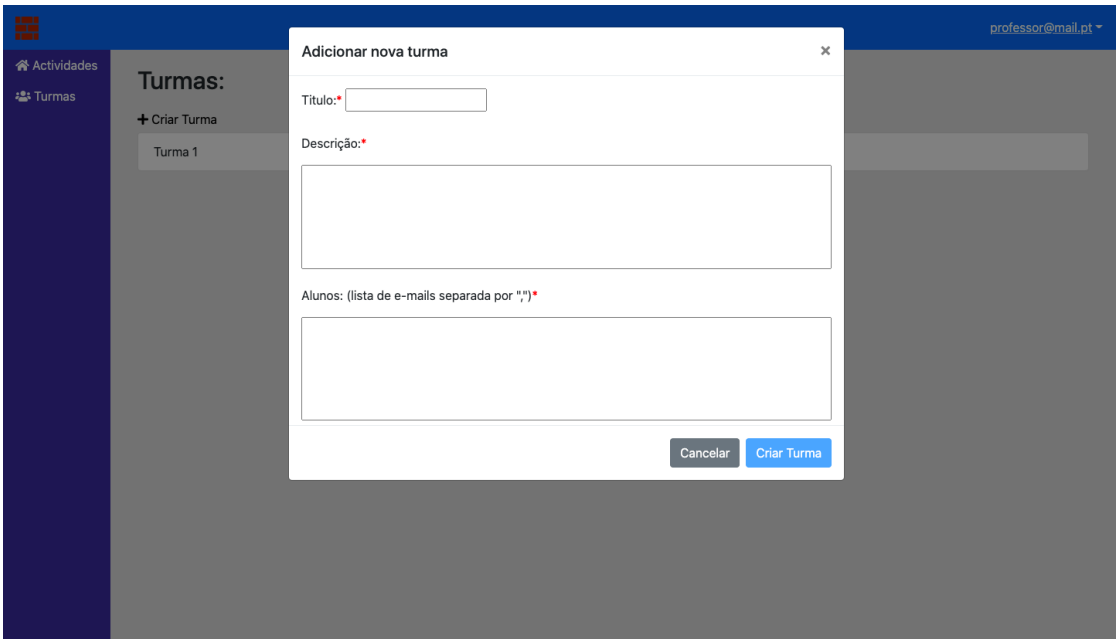


Figura 3.19: Interface da página de turmas



suas turmas. No ecrã central é exposta uma lista com as turmas criadas pelo utilizador e, na parte superior da lista, tem opção de criar uma nova turma. A criação de turma é um processo simples pedindo apenas ao utilizador um nome para a turma, uma pequena descrição e uma lista de *e-mails* de alunos para acrescentar à turma. Este processo é realizado através do preenchimento do formulário apresentado na Figura 3.20.

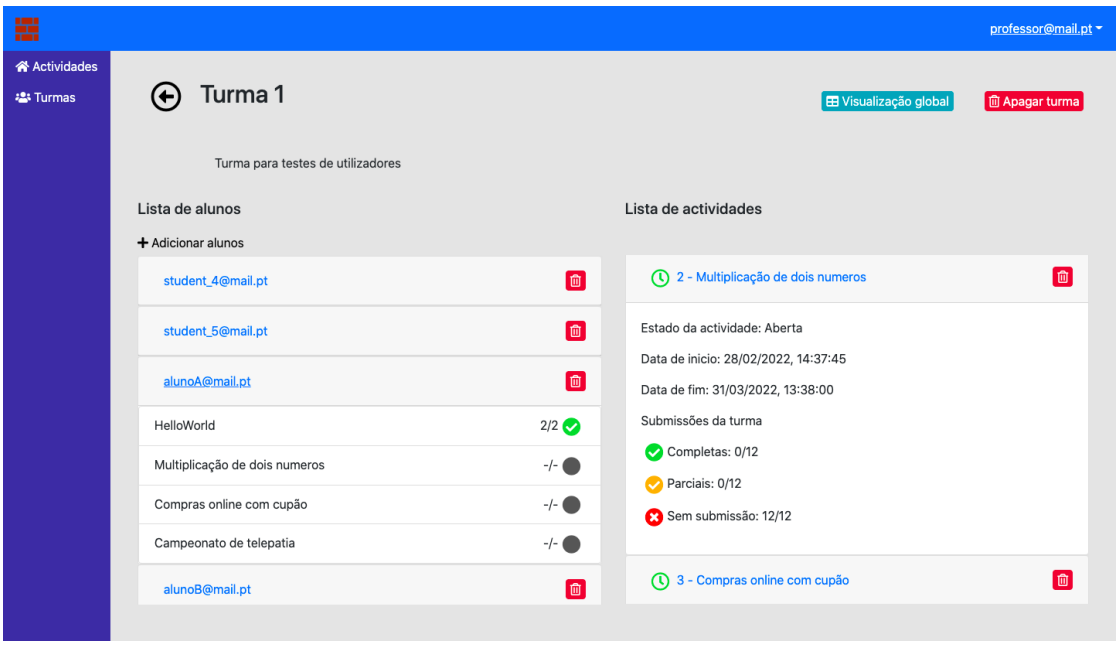
Figura 3.20: Interface de criação de turmas



Feedback de submissões

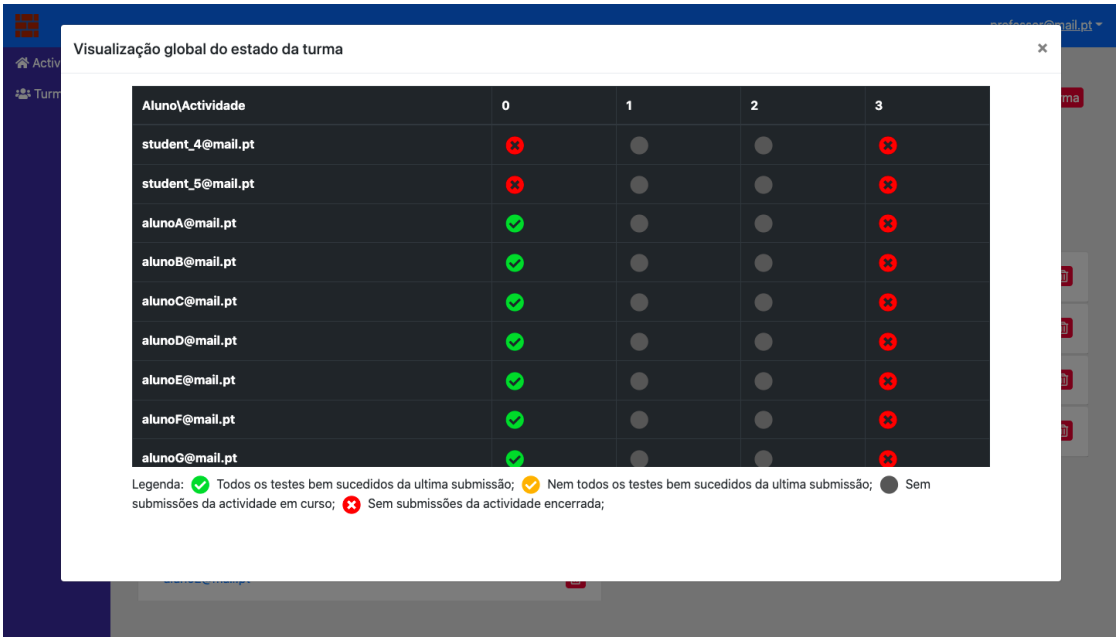
Para consultar o *feedback* das submissões realizadas, o utilizador deve pressionar num elemento específico da lista da Figura 3.19, e será encaminhado para uma interface onde tem acesso a detalhes sobre a turma seleccionada, como exemplificado na Figura 3.21.

Figura 3.21: Interface da página de consulta de uma turma específica



Esta interface para detalhes específicos de uma turma, ilustrado na Figura 3.21, apresenta o nome da turma, uma pequena descrição da turma, o acesso à visualização global da turma, uma lista de alunos que pertençam à turma com a opção de adicionar alunos novos e uma lista de actividades associadas à turma. Ao seleccionar um dos elementos da lista de actividades, o utilizador tem acesso ao estado da actividade, aos dados gerais da actividade em relação aos alunos da turma e às datas de inicio e fim da actividade. No canto superior direito da interface, o utilizador pode aceder à visualização global da turma. A visualização global da turma consiste numa tabela de estados de actividade de cada aluno para cada actividade. Na Figura 3.22 é apresentado a interface da visualização global da turma, onde o utilizador tem acesso, em simultâneo, ao estado de todos os alunos da turma relativo às actividades atribuídas.

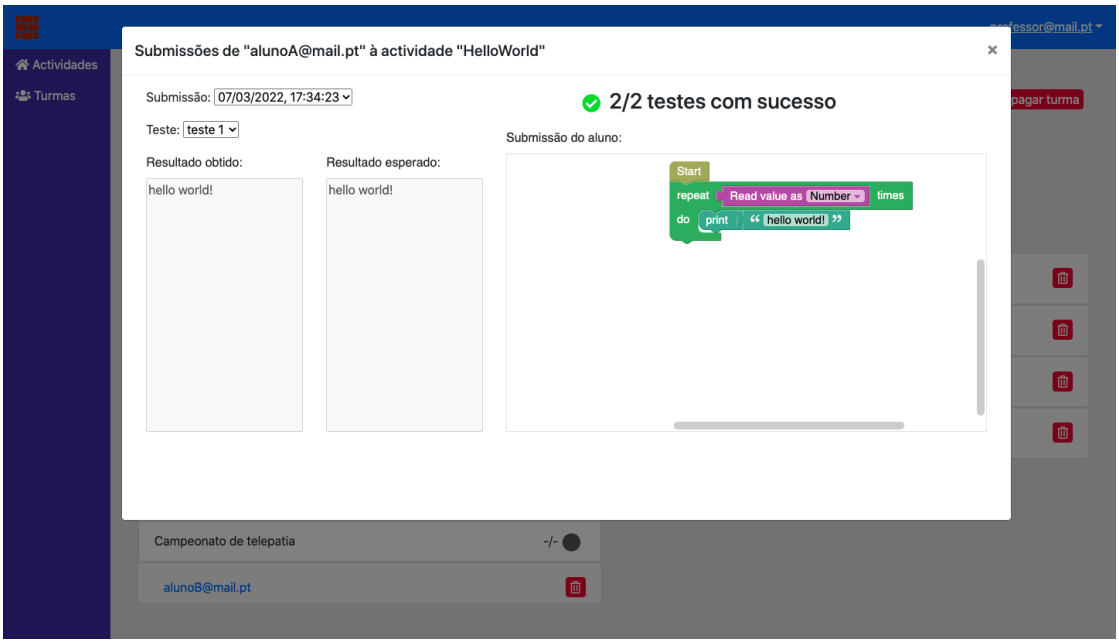
Figura 3.22: Interface da visualização global de uma turma específica



Ao seleccionar um dos elementos da lista de alunos, o utilizador tem acesso a dados do aluno seleccionado em relação ás actividades atribuídas como, caso exista, o estado da última submissão à actividade. O utilizador pode ainda seleccionar uma destas actividades para ver as submissões do aluno.

A Figura 3.23 mostra a interface onde o professor pode visualizar as submissões de um aluno relativo a uma actividade específica. Nesta interface o professor pode visualizar todas as submissões efectuadas pelo aluno através do *dropdown* no canto superior esquerdo. Em cada submissão, o professor pode visualizar os resultados do aluno para cada teste da actividade e a solução submetida pelo aluno.

Figura 3.23: Interface da visualização de uma submissão de um aluno pertencente a uma turma específica



Atribuição de actividades a turma

Após criação de uma turma, o utilizador pode então associar actividades à sua turma. Como descrito anteriormente, esta associação tem uma data de inicio e uma data de fim que determinam o estado que a actividade pode tomar. Os estados que uma actividade pode tomar após ser associada a uma turma são ”aberta”, ”encerrada” e ”agendada”. Estes estados determinam também a visibilidade das actividades aos alunos, nas turmas.

Visualização e execução actividades de programação

Um utilizador do tipo professor pode também executar as suas actividades de forma a poder testar antes de as atribuir aos seus alunos. A interface de execução de uma actividade por um utilizador do tipo professor é semelhante à interface para execução de actividades do utilizador do tipo aluno ilustrado na Figura 3.25. A única diferença é ter o botão de submeter desactivado pois não é permitida submissão da solução à actividade. Nesta interface temos o titulo da actividade, o enunciado, a documentação e os testes, no topo. A lista de testes, ao se seleccionar um dos testes abre uma janela com o resultado da execução do teste individual (ou ”por realizar”, caso o teste ainda não tenha sido realizado), o *output* recebido pela execução do programa do aluno e o *output* esperado pelo teste. Um teste é bem sucedido se o *output* obtido pela execução do programa for igual ao *output* contido no ficheiro de saída do teste. Na área central existe uma área de trabalho e uma *toolbox*, onde o utilizador pode implementar o seu programa arrastando os blocos existentes na *toolbox* para a área de trabalho e encaixando-os. Na parte inferior da

interface existem 4 botões onde o utilizador pode guardar a sua solução actual, carregar a última solução guardada, testar a execução da sua solução com os testes da actividade e submeter a sua solução. A submissão de soluções apenas é permitida a utilizadores do tipo aluno.

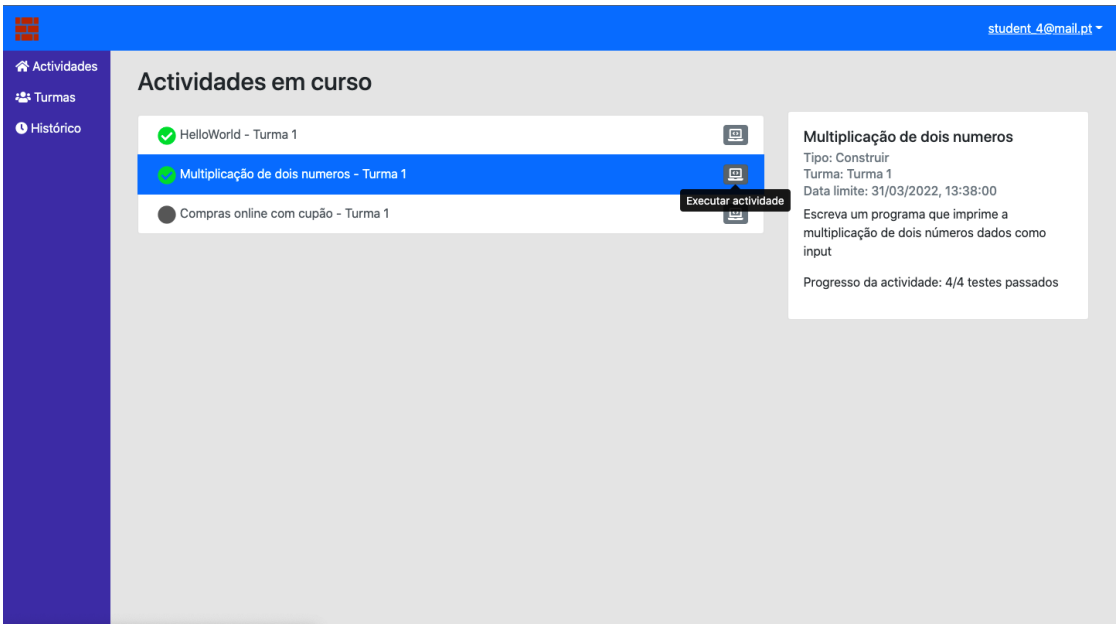
3.2.2 Funcionalidades do utilizador aluno

A componente do aluno permite:

- Executar actividades de programação: realizar actividades de programação que foram atribuídas às turmas que o aluno pertence.
- Consultar actividades: Visualizar actividades cujo prazo de realização já tenha terminado.
- Consultar as turmas: verificar a que turmas o aluno pertence e alguns detalhes sobre a turma.

Na Figura 3.24, é apresentado a interface da página principal do aluno, onde terá uma lista de actividades que foram atribuídas às turmas a que pertence e que se encontram no estado "aberto". As actividades nesta lista, no ecrã principal, estão dentro do prazo para serem submetidas. Actividades que estão "fechadas" podem ser acedidas a partir da barra de navegação lateral através do histórico de actividades.

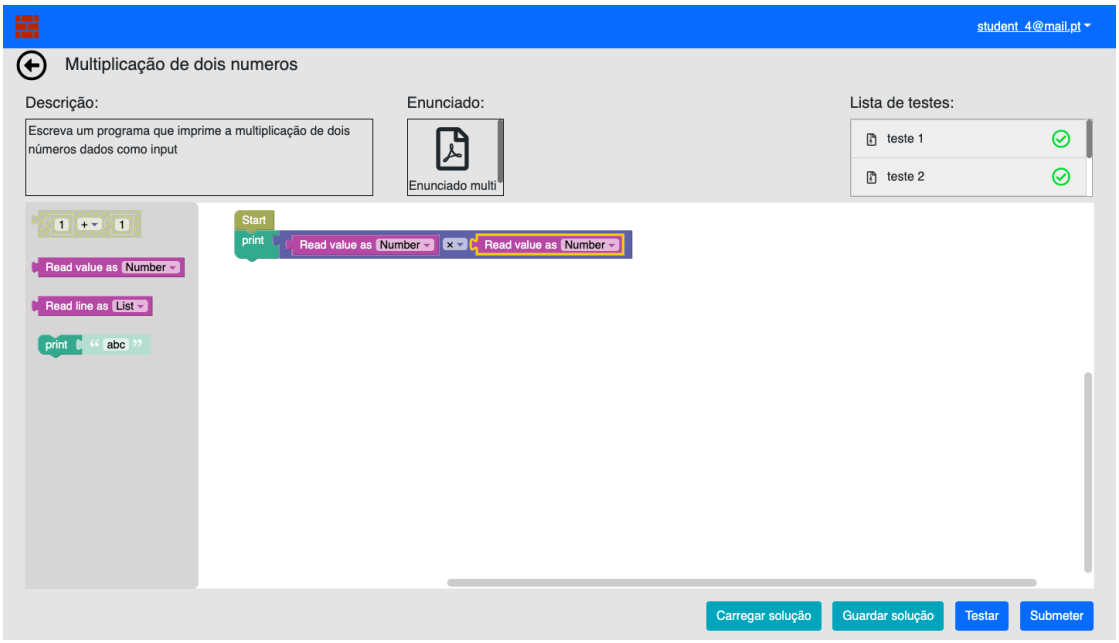
Figura 3.24: Interface da página principal do aluno



Execução actividades de programação

Na Figura 3.25, é apresentado o ecrã de resolução de uma actividade. Na parte inferior é ilustrada a área de trabalho da actividade. Na área de trabalho estará presente a *toolbox* com os blocos permitidos para a execução da actividade e, ao passar com o rato sobre um bloco, o número de blocos restantes permitidos para a construção do programa.

Figura 3.25: Interface da execução de uma actividade do aluno



A actividade poderá ser resolvida arrastando e parametrizando os blocos disponíveis na *toolbox* para a área de trabalho e encaixando os blocos de forma a obter o resultado descrito no enunciado disponibilizado. Na parte superior da área principal, o utilizador pode aceder ao enunciado, à documentação fornecida pelo professor ao criar a actividade e à lista de testes da actividade. Esta lista de testes, ao seleccionar um dos testes, permite aceder a dados relativos ao teste seleccionado. Estes dados consistem no resultado do teste, o resultado obtido pela execução do programa do aluno e o resultado esperado da execução do programa.

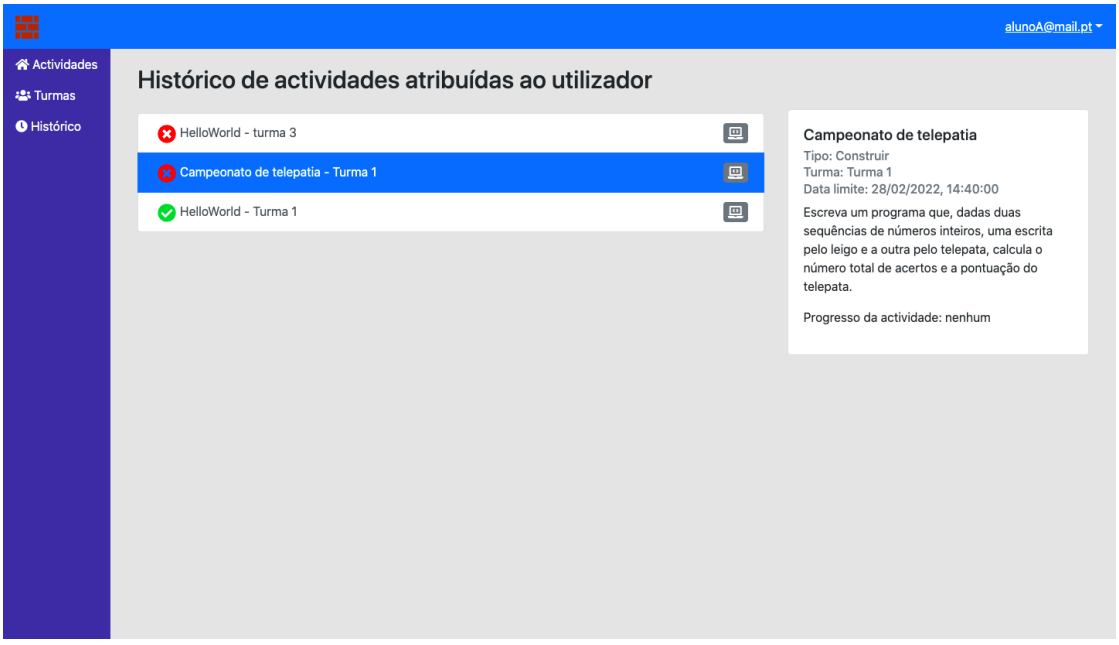
Enquanto o utilizador não carregar no botão de testar, em todos os testes estará indicado que estão por realizar. Qualquer alteração no programa presente na actividade coloca todos os testes com indicação que estão por realizar e, caso haja resultados, remove os resultados anteriores do teste. Quando o utilizador pressiona o botão *Testar*, a aplicação irá executar o programa, para cada teste, usando os valores presentes no ficheiro de entrada de cada teste. Após a execução, os resultados de cada execução do programa serão colocados junto ao seu respectivo teste, onde o utilizador pode observar lado-a-lado com o resultado esperado para o teste. Finalmente, o resultado da execução de cada teste é comparado com o resultado esperado do seu respectivo teste e, se forem iguais, o teste terá

indicação do resultado correcto, se forem diferentes, o teste terá indicação do resultado incorrecto.

Visualização de actividades de programação encerradas

A aplicação permite aos utilizadores visualizar e testar actividades já encerradas. Esta funcionalidade tem como objectivo aproveitar actividades que o aluno pode ou não ter realizado, de forma a poder aceder a recursos associados à actividade e poder testar soluções alternativas. Na Figura 3.26 é apresentado o separador do histórico de actividades, que contém uma lista de todas as actividades encerradas de todas as turmas que o aluno pertence. Cada item da lista exibida na Figura 3.26 é composto por um ícone relativo à última submissão do aluno à actividade, o nome da actividade, a turma à qual a actividade foi atribuída e um botão para visualizar a actividade. As actividades encerradas têm um ícone verde se a última submissão passou aos testes com sucesso, têm um ícone amarelo se a última submissão não passou a todos os testes com sucesso e tem um ícone vermelho se o aluno não chegou a realizar submissão à actividade quando se encontrava no estado "aberta". Ao seleccionar um dos itens da lista na Figura 3.26, são mostrados detalhes sobre a actividade como o nome da actividade, a data final, uma descrição da actividade e o número de testes bem sucedidos da última submissão.

Figura 3.26: Separador do histórico de actividades, na área do aluno

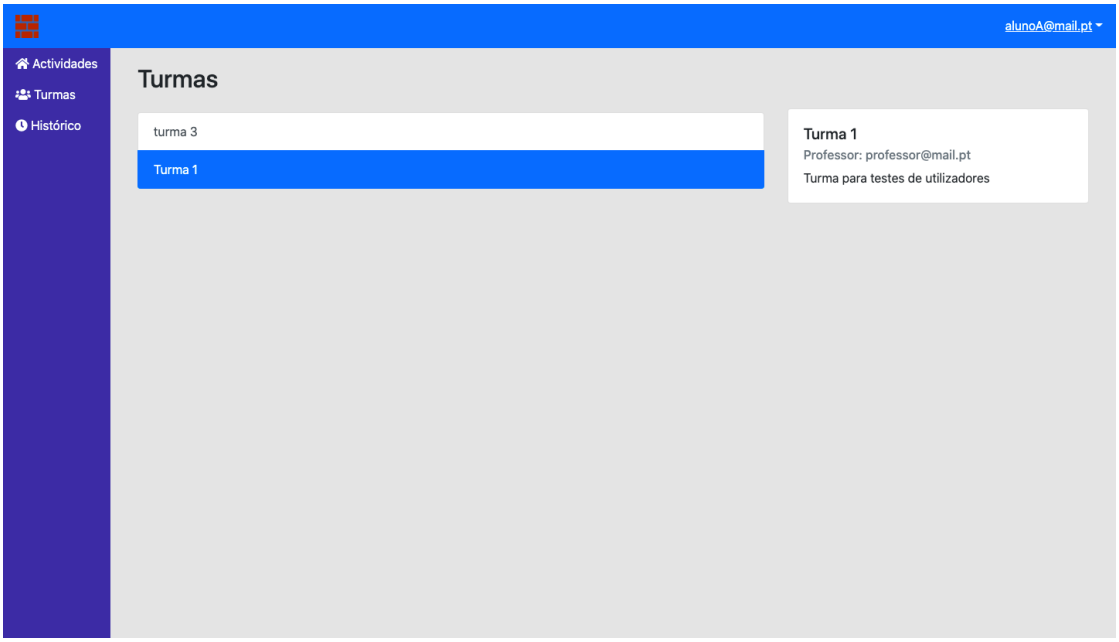


Consulta de turmas

A consulta de turmas pode ser acedida através do separador turmas da barra lateral de navegação. A área das turmas, apresentado na Figura 3.27, contém uma lista com

as turmas que o aluno pertence. Ao seleccionar uma das turmas presentes na lista, são exibidos alguns detalhes sobre a turma como o *e-mail* do professor e a descrição da turma. Esta funcionalidade tem como objectivo explicitar ao aluno as turmas ao qual pertence e, ao exibir os *e-mails* dos professores das turmas, permitir dar aos alunos uma forma de contactar os professores.

Figura 3.27: Separador das turmas, na área do aluno



3.3 Modelo de dados

A Figura 3.28 apresenta o modelo de dados do sistema *Bricks*.

A aplicação tem dois tipos de utilizadores, ´´aluno" e "professor".

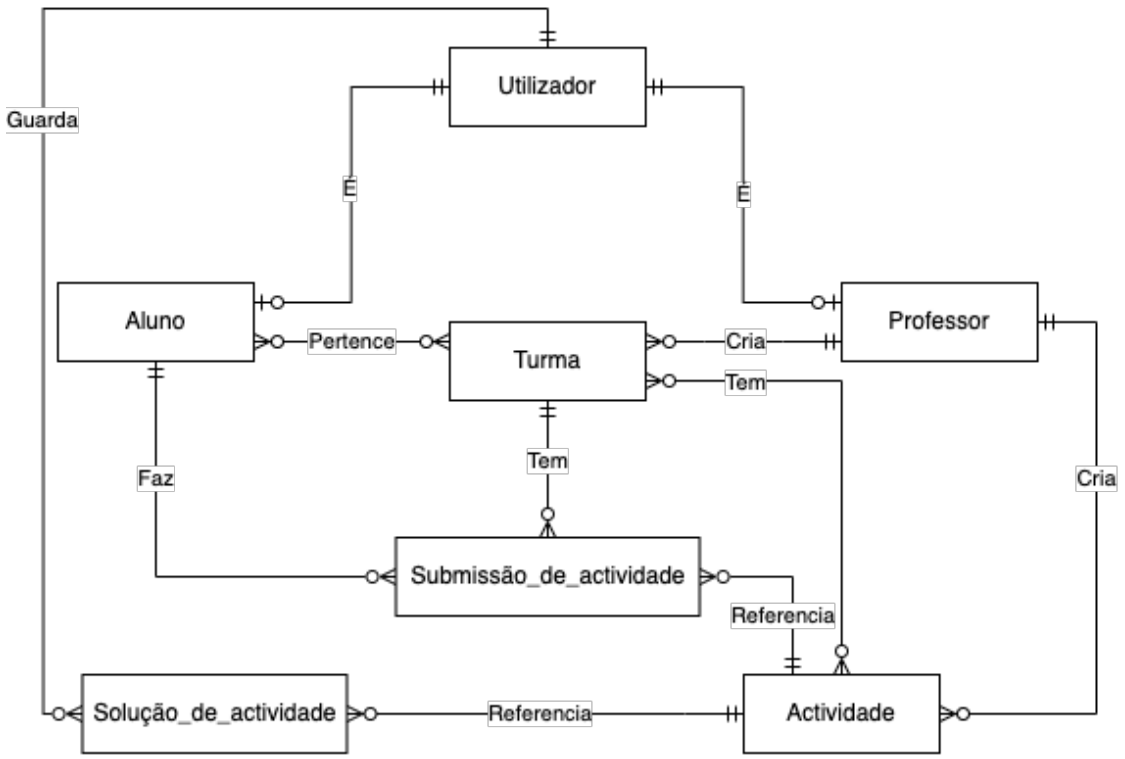
Na aplicação, um professor pode criar múltiplas turmas. Uma turma tem sempre um professor associado, que é o professor que a criou.

Um professor pode também criar múltiplas actividades. Uma actividade tem sempre um professor associado, que é o professor que a criou.

Uma turma é composta por vários alunos, e cada aluno pode pertencer a múltiplas turmas.

A entidade "Actividade" tem como objectivo organizar as actividades armazenadas no contexto da aplicação. Para tal, esta actividade tem dois atributos principais: o identificador da actividade e a seu identificador (localização) no sistema de ficheiros. A entidade também tem atributos como o título e a descrição, que servem de apoio ao cliente, de forma a não ser necessário enviar o arquivo da actividade sem ser estritamente necessário,

Figura 3.28: Diagrama do modelo de dados da aplicação



como por exemplo para obter listas de actividades. Na localização descrita no identificador (localização) é guardada uma referência para o sistema de ficheiros onde o arquivo da actividade foi armazenado.

A entidade "Solução_de_actividade" representa uma solução do aluno a uma atividade. Uma solução guardada pertence sempre a um utilizador e é sempre relativa a uma actividade. Uma actividade pode ou não ter soluções guardadas pelos utilizadores, sejam eles professores ou alunos. Uma solução de uma actividade é composta por uma referência a um utilizador, uma referência a uma actividade e a uma referencia ao sistema de ficheiros onde a solução é armazenada.

A entidade "Submissão_de_actividade" representa a submissão de uma solução de um aluno a uma actividade atribuída à turma ao qual o aluno pertence. A submissão guarda referências para o aluno que realizou, turma que pertence, actividade realizada, local onde o ficheiro de submissão ficou armazenado no sistema de ficheiros e a data de submissão. Apesar de os dados da solução serem iguais, o ficheiro de submissão é diferente do ficheiro gerido pela entidade "Solução_de_actividade" pois o aluno pode realizar múltiplas submissões à mesma actividade e todas as submissões ficam armazenadas, enquanto no caso das soluções de actividade apenas a última solução do utilizador (aluno ou professor) para uma determinada actividade fica armazenada.

3.4 Síntese

A aplicação *Bricks* é uma ferramenta que permite criar e executar actividades de programação por blocos. Para tornar a aplicação mais útil para o contexto do ensino, a aplicação tem funcionalidades como criação de turmas, atribuição de actividades a turmas e consulta de *feedback* de submissões.

De forma a ser utilizável no contexto do ensino, é importante que esta ferramenta seja acessível remotamente e que seja escalável para poder acomodar vários utilizadores em simultâneo.

4

Implementação da aplicação *Bricks*

Neste capítulo é descrito em detalhe o desenvolvimento e implementação da aplicação *Bricks* apresentada no capítulo anterior. Nomeadamente a arquitetura da aplicação, as tecnologias utilizadas e a estrutura das páginas dos clientes da aplicação.

4.1 Arquitectura da aplicação

O servidor da aplicação é composto por três camadas: a camada de comunicação, a camada de lógica e a camada de dados. A Figura 4.1 apresenta um esquema da arquitectura do servidor. O servidor foi implementado em *Node.js*. A aplicação é acessível através de um *browser web* usando o protocolo *HTTPS*. Os pedidos ao servidor são geridos na camada de comunicação do servidor. Para lidar com os pedidos dos clientes, o servidor utiliza a *framework Koa.js*. Esta *framework* implementa uma arquitectura *REST*.

A camada de lógica do servidor lida com o processamento dos dados recebidos através da camada de comunicação. Este processamento de dados inclui a gestão dos utilizadores, actividades e submissões de actividades. De forma a desempenhar estas funções, o servidor terá de armazenar dados numa base de dados e no sistema de ficheiros.

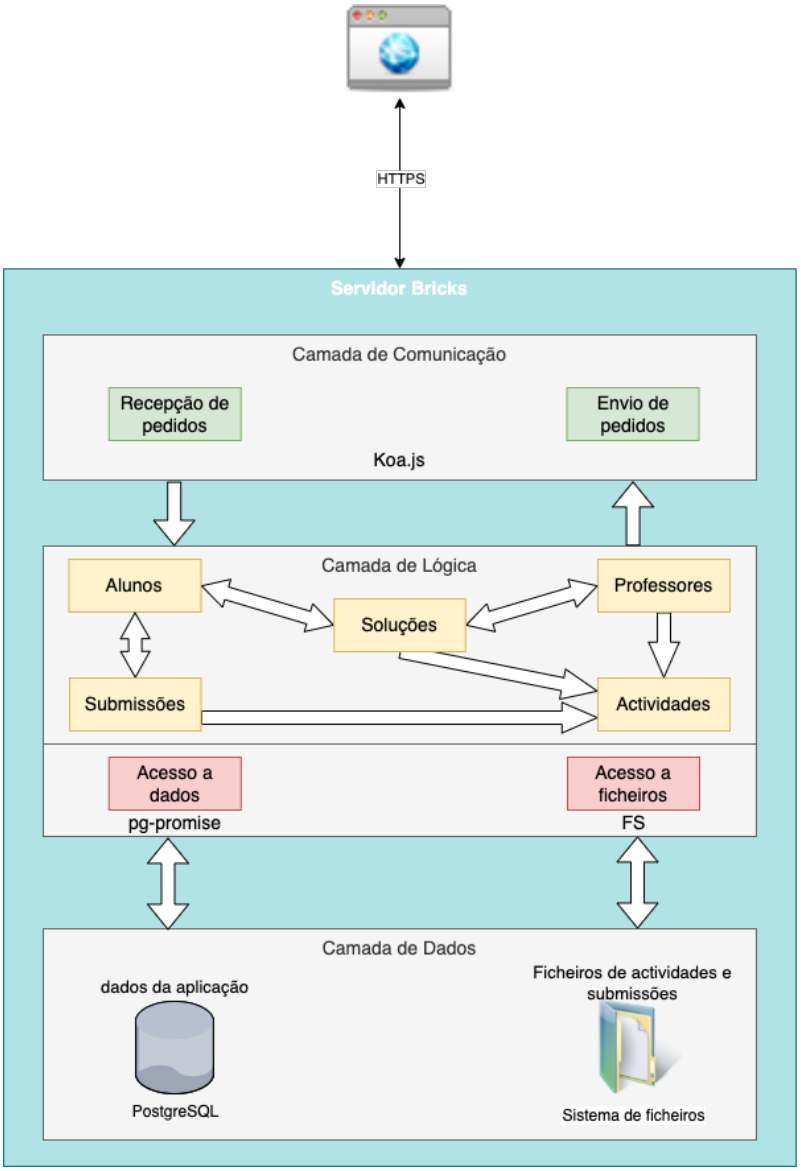
De forma a garantir as funcionalidades especificadas no capítulo anterior, foi necessário uma análise das *frameworks* existentes. Foi também necessário alguma atenção durante a implementação do servidor como o planeamento e implementação dos *endpoints* que o servidor disponibiliza.

Para iniciar a implementação do servidor foi primeiro necessário definir a linguagem e *frameworks* a utilizar.

4.1.1 Camada de comunicação

A comunicação entre servidor e cliente, durante o planeamento da arquitectura do sistema, foi planeado seguir uma arquitectura *REST*. Porém, existem bastantes opções de *frameworks* que suportam esta arquitectura. Como exemplo de pares linguagem-*frameworks*

Figura 4.1: Arquitectura do servidor *Bricks*



consideradas para a implementação do servidor temos *NodeJS* e *Koa.js*[1], *NodeJS* e *Express*[2] e *Python* e *Django*[3]. Todas estas opções seriam viáveis pelo que a escolha de *NodeJS* se deveu à preferência de utilização de apenas uma linguagem para o desenvolvimento do projecto, pois a biblioteca *Google Blockly* apenas disponibiliza *API* em *javascript*. A opção de *Koa.js* em detrimento de *Express* deveu-se a preferência pessoal.

[1] https://koajs.com/
[2] https://expressjs.com/
[3] https://www.djangoproject.com/

4.1.2 Camada de lógica

A camada de lógica consiste na parte do servidor responsável por:

- Autenticação dos pedidos ao servidor - O servidor, ao receber um pedido e antes de o realizar, tem de verificar se o utilizador que enviou o pedido existe e tem acesso à funcionalidade que invocou.

- Validação dos dados enviados com o pedido - O servidor, após autenticar o utilizador, verifica se os dados enviados com o pedido são válidos para realizar a funcionalidade invocada.

- Comunicação dos dados para a camada de dados - O servidor, após validar os dados enviados, invoca métodos da camada de dados para realizar operações sobre a base de dados e/ou sistema de ficheiros de acordo com a operação pedida (inserção, edição, remoção ou consulta).

No servidor, as operações estão separadas em módulos, cada um destes módulos responsável por gerir o seu tipo de pedidos. Estes módulos apenas consistem em grupos de operações que actuam sobre o mesmo alvo. Os módulos existentes no servidor são o módulo de alunos, o módulo de professores, o módulo de actividades, o módulo de soluções e o módulo de submissões. No módulo de alunos, por exemplo, encontramos as operações de criação de turmas (grupos de alunos) e consulta de actividades pertencentes a uma turma. No módulo de professores, encontramos operações como, por exemplo, a consulta de turmas que determinado professor criou ou consulta de actividades criadas por um professor. No módulo de actividades, por exemplo, encontramos operações como a criação de uma nova actividade ou a atribuição de uma actividade a uma turma. No módulo de soluções, encontramos operações como guardar a solução de um utilizador ou carregar uma solução previamente guardada. No módulo de submissões, por exemplo, encontramos operações como a criação de uma nova submissão, consulta de submissões de um determinado aluno ou consulta da última submissão de um aluno a uma actividade. Estes módulos têm como objectivo tornar o código mais organizado e, por conseguinte, mais fácil de compreender.

4.1.3 Camada de dados

De acordo o modelo de dados apresentado no capítulo anterior, o conjunto de dados necessário para o funcionamento da aplicação tem dependências entre as várias entidades. Estas dependências impactaram o rumo das opções para uma solução baseada em *SQL* em detrimento de um sistema de base de dados *NoSQL*. Vários sistemas de base de dados foram *SQL* tomados como potenciais candidatos, entre eles *MySQL* e *PostgreSQL*. Ambos os sistemas desempenhariam um papel adequado na aplicação pelo que a escolha final de utilizar o *PostgreSQL* foi baseada em familiaridade de utilização.

A base de dados constitui uma parte importante do servidor pois a modelação e implementação teve de ter em conta a arquitectura do sistema e as funcionalidades da aplicação. Na base de dados são armazenados os dados dos utilizadores, turmas, actividades e submissões. No caso das actividades e submissões, que utilizam o sistema de ficheiros, um dos elementos presentes na base de dados é o *filepath*, que faz referência ao lugar de armazenamento. O módulo de *Node.js* que o servidor utiliza para comunicar com a base de dados é o *pg-promise*[4].

Os ficheiros armazenados no sistema de ficheiros são os ficheiros que descrevem as actividades da aplicação e os ficheiros das submissões. As actividades são armazenadas em arquivos de extensão *zip*. As soluções e submissões de actividades são armazenadas em em ficheiros *json*. O módulo de *Node.js* utilizado para interagir com o sistema de ficheiros é o *fs*.

O ficheiro de submissão de uma actividade consiste num ficheiro de extensão JSON (Javascript Object Notation) que contém:

- O identificador da actividade.
- O identificador da turma.
- O identificador do aluno.
- Os testes da actividade.
- Os resultados dos testes da actividade.
- O programa submetido pelo aluno, que consiste numa definição *XML* (Extensible Markup Language) do conteúdo da área de trabalho e das variáveis utilizadas na área de trabalho.

A Figura 4.2 ilustra uma submissão à actividade exemplo, onde se pode observar todos os elementos que constituem o ficheiro de submissão.

4.2 *Google Blockly*

Para integrar programação por blocos na aplicação foi usada a biblioteca *Google Blockly*. Esta biblioteca permite criar um ambiente de trabalho de programas por blocos, onde o utilizador arrasta blocos de uma *toolbox* para uma área de trabalho e encaixa-os compondo um programa. A biblioteca trata de interfaces da área de trabalho e interfaces relativos aos blocos, juntamente com as suas respectivas traduções para código *javascript*. Para execução destes programas por blocos, é necessário utilizar um interpretador de código *javascript*. Na aplicação, a biblioteca utilizada para interpretação de código proveniente

[4]https://www.npmjs.com/package/pg-promise

Figura 4.2: Conteúdo do ficheiro de submissão para uma actividade


```
{
  "actID": "1059452106-1645969506249",
  "userID": "alunoA@mail.pt",
  "classID": "20",
  "actTests": [
    ["teste 1","input","5 3"],
    ["teste 1","output","15"],
    ["teste 2","input","10 25"],
    ["teste 2","output","250"],
    ["teste 3","input","1 1"],
    ["teste 3","output","1"],
    ["teste 4","input","9 9"],
    ["teste 4","output","81"]
  ],
  "testResults": [true,true,true,true],
  "testOutput": ["15","250","1","81"],
  "solution": {
    "workspace": "<xml xmlns=\"http://www.w3.org/1999/xhtml\"><block
      =\"text_print\" id=\"^vQKY]H%9[CAT%O{kMG~\"><value name=\"TEXT\
      =\"math_arithmetic\" id=\"!]b,hYSLfwBrd`;8VzLS\"><field name=\"
      =\"NUM\">1</field></shadow><block type=\"ler_pr_ximo_valor\" id
      ></value><value name=\"B\"><shadow type=\"math_number\" id=\"W|
      +y8w]D1m\"><mutation tp=\"Number\"></mutation><field name=\"typ
    "variables": "<variables xmlns=\"http://www.w3.org/1999/xhtml\"><
  }
}
```


dos blocos foi o *JS interpreter*[5]. A utilização destas duas bibliotecas permite a execução de qualquer programa por blocos que utilize os blocos presentes no *Google Blockly*.

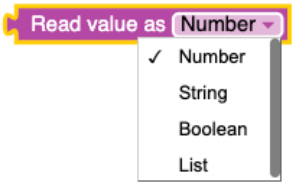
Na aplicação, as actividades de programação têm testes, que são compostos por ficheiros de entrada e saída. De forma a poder afectar os programas criados pelo utilizador usando estes ficheiros, foi necessário fazer algumas alterações, nomeadamente, criar blocos de leitura para que durante a realização das actividades, fosse possível um programa aceder aos valores de entrada dos testes.

No capítulo anterior foram apresentados os blocos de leitura criados para a aplicação. Para permitir o funcionamento destes blocos, o interpretador do código gerado pela tradução dos blocos tem que ter uma variável que guarde a posição do último valor lido. O bloco de leitura de valor lê o valor presente na posição guardada nesta variável e, ao terminar a leitura, avança a variável para a posição seguinte. Na Figura 4.3, é ilustrado o bloco e o código correspondente ao bloco de leitura de valor.

O bloco de leitura de linha lê a linha onde a posição actual da variável se encontra e, ao terminar a leitura, avança a posição da variável para o início da linha de entrada seguinte. Na Figura 4.4, é ilustrado o bloco e o código correspondente ao bloco de leitura

[5]https://github.com/NeilFraser/JS-Interpreter

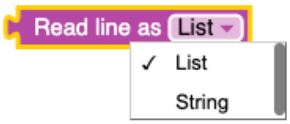
Figura 4.3: Bloco de leitura de valor, à esquerda, e a sua definição em *javascript*, à direita




```
Blockly.JavaScript['ler_pr_ximo_valor'] = function(block) {
      var code = 'executionInputArgsLength > inputArgsPointer ?
executionInputArgs[managePointers(0)] : "Excedeu os argumentos"';
      return [code, Blockly.JavaScript.ORDER_ASSIGNMENT];
    };
```

de linha.

Figura 4.4: Bloco de leitura de valor, à esquerda, e a sua definição em *javascript*, à direita




```
Blockly.JavaScript['ler_pr_xima_linha'] = function(block) {
      var dropdown_type = block.getFieldValue('type');
      var code;
      switch (dropdown_type) {
        case 'String':
            code = "lineEnds.length >= 1 ? managePointers(2) : 'EoF'";
            break;
        case 'Array':
            code = "lineEnds.length >= 1 ? managePointers(1) : 'EoF'";
            break;
      }
      var a = "";
      return [a + (code), Blockly.JavaScript.ORDER_ASSIGNMENT];
    };
```

Nas Figuras 4.3 e 4.4, onde se ilustrava as implementações dos blocos de leitura, é possível observar referencias à função *managePointers()*. Esta função tem como propósito permitir a utilização de ambos os blocos de leitura no mesmo programa.

Os encaixes dos blocos não depende apenas da forma do bloco, os tipos de bloco também influenciam a possibilidade de dois blocos se encaixarem. Por exemplo, um bloco de operação aritmética requer que ambos os blocos que são encaixados nele sejam do tipo "Número". Com este pormenor em mente, o *dropdown* do bloco de leitura de valor, que contém os tipos de blocos do *Google Blockly*, apenas encaixa em blocos dependendo do tipo escolhido. O *dropdown* do bloco de leitura de linha funciona de forma diferente, convertendo a linha de entrada lida no formato escolhido.

As soluções das actividades de programação, na aplicação, são compostas pelo bloco "start" seguidas do conjunto de blocos encaixados pelo utilizador. O comportamento do *Blockly* e do interpretador na ausência deste bloco é interpretar todo o conteúdo da área de trabalho ordenado da esquerda para a direita e de cima para baixo, incluindo a diagonal gerada por ambas as direcções. Para tal, de forma a facilitar a tornar os programas mais legíveis para os utilizadores, apenas os blocos que estiverem encaixados no seguimento deste bloco serão executados.

As actividades de programação podem ter limites de utilização aplicados nos seus

blocos. O ambiente de edição de programas do *Google Blockly*, apesar de não ter essa funcionalidade implementada, fornece forma de poder implementar uma solução com este fim, sendo esta os eventos sobre blocos. Cada vez que um bloco é arrastado da *toolbox* para a área de trabalho, removido da área de trabalho ou movido para uma nova localização, o bloco emite um evento a sinalizar a respectiva acção. Desta forma, a implementação dos limites de utilização de blocos consistiu em:

- Quando se abre uma actividade pela primeira vez, para cada bloco presente na *toolbox*, é atribuído o valor inicial do limite, caso exista, de forma a apresentar a mensagem, ao passar o ponteiro sobre o bloco, "Limite de utilização: X restantes". Caso não exista limite, a mensagem apresentada é Limite de utilização: sem limite".

- Quando um bloco sinaliza um evento de criação de bloco, se houver limites aplicados sobre o bloco, a mensagem apresentada pelo bloco é actualizada para número correcto de utilizações restantes e, caso não haja mais utilizações permitidas, o bloco na *toolbox* é colocado num estado desactivado até uma das suas instâncias seja removida.

- Quando um bloco sinaliza um evento de remoção de bloco, se houver limites aplicados sobre o bloco, a mensagem apresentada pelo bloco é actualizada para número correcto de utilizações restantes e se o bloco, na *toolbox*, estiver desactivado é então reactivado.

4.3 Estrutura de páginas do tipo aluno

O foco principal do cliente para utilizadores do tipo aluno é a execução de actividades de programação.

O utilizador do tipo aluno começa por introduzir as suas credenciais de acesso à aplicação (*Login* na Figura 4.5). Na figura, o código de cores agrupa as funcionalidades por página.

Após efectuar o *Login* na aplicação (transição de *Login* para *Dashboard* na Figura 4.5), o cliente começa por pedir ao servidor o conjunto de actividades ”abertas” para o utilizador, expondo-as numa lista presente no separador lateral de actividades, sendo este o separador principal do cliente do tipo aluno (*Separador actividades* na Figura 4.5). Cada item da lista de actividades ”abertas”, para além de fornecer dados sobre a actividade ao utilizador, tem um botão para executar a actividade.

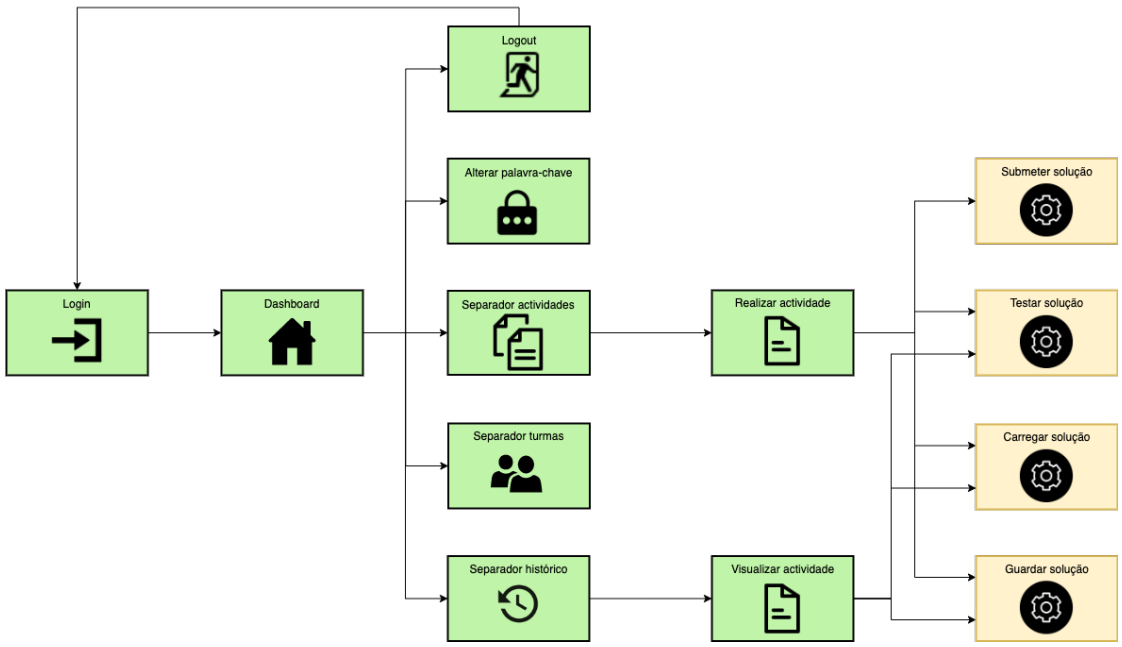
Ao pressionar para executar uma actividade, o servidor envia ao cliente a página da ferramenta de execução de actividades juntamente com o ficheiro da actividade (*Realizar actividade* na Figura 4.5). Na ferramenta de execução de actividades, o utilizador do tipo aluno pode aceder aos ficheiros de extensão PDF (*Portable Document Format*) correspondentes ao enunciado e documentação, caso exista. Após a leitura do enunciado, que deverá conter o objectivo da actividade, o utilizador do tipo aluno deve proceder à construção

da solução à actividade. Durante a construção da solução, o utilizador do tipo aluno pode guardar a solução que tem presente na área de trabalho (*Guardar solução* na Figura 4.5), enviando a solução para ser armazenada no servidor, pode carregar a última solução guardada (*Carregar solução* na Figura 4.5), pode testar a solução actual utilizando os testes da actividade (*Testar solução* na Figura 4.5) e pode submeter a solução actual (*Submeter solução* na Figura 4.5), finalizando a actividade. Os testes da actividade são realizados localmente, permitindo assim uma menor carga no servidor. Este detalhe é relevante para a potencial escalabilidade do sistema. Enquanto a actividade estiver ”aberta”, o aluno pode sempre submeter soluções, porém, o resultado que lhe é apresentado é sempre relativo à última submissão realizada.

No separador de turmas (*Separador turmas* na Figura 4.5), o cliente do aluno faz um pedido ao servidor que responde, enviando os dados sobre conjunto de turmas que o aluno se encontra inserido. Estes dados são colocados numa lista e apresentados ao utilizador do tipo aluno com objectivo meramente informativo.

O separador ”Histórico de actividades” (*Separador histórico* na Figura 4.5) tem o objectivo de apresentar uma lista de actividades ”encerradas”. As actividades são apresentadas de forma semelhante à lista de actividades ”abertas”, mencionada anteriormente, porém, apenas podem ser visualizadas. A visualização de actividades encerradas utiliza a ferramenta de execução de actividades pois, mesmo não permitindo a submissão, o utilizador pode editar soluções, guardar ou carregar soluções, visualizar os ficheiros *.pdf da actividade e testar a solução actual usando os testes da actividade.

Figura 4.5: Transições de interfaces da aplicação no cliente do tipo aluno

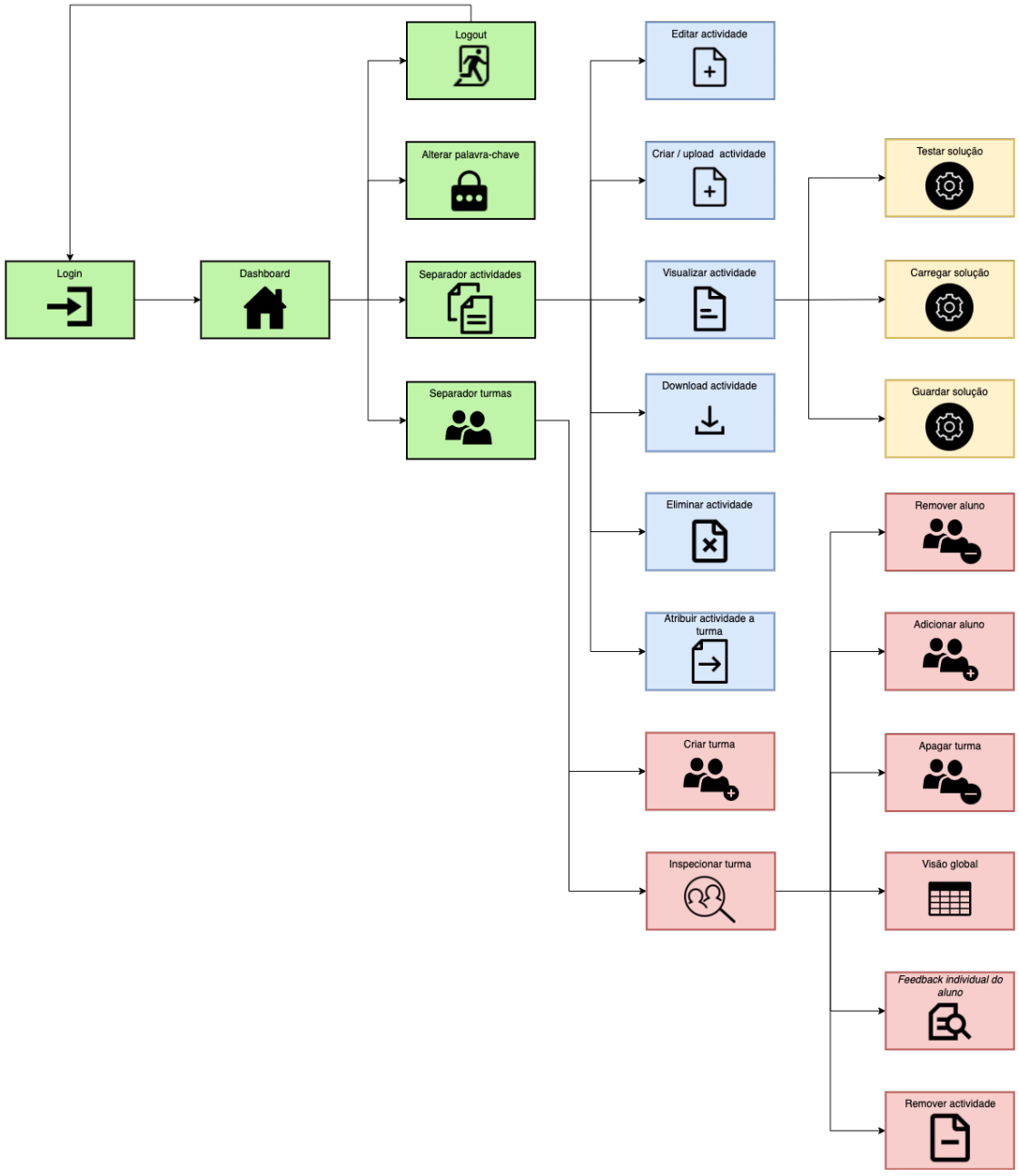


4.4 Estrutura de páginas do tipo professor

O foco principal do cliente do tipo professor é criar actividades de programação, visualizar actividades de programação, gerir turmas e obter *feedback* de execuções de actividades de utilizadores do tipo aluno que pertençam às suas turmas.

O utilizador do tipo aluno começa por introduzir as suas credenciais de acesso à aplicação (*Login* na Figura 4.6). Na figura, o código de cores agrupa as funcionalidades por página.

Figura 4.6: Transições de interfaces da aplicação no cliente do tipo professor



Após efectuar o *Login* na aplicação (transição de *Login* para *Dashboard* na Figura 4.6), o cliente começa por pedir ao servidor o conjunto de actividades criadas pelo utilizador do tipo professor, exibindo-as numa lista presente no separador lateral de actividades (*Separador actividades* na Figura 4.6), sendo este o separador principal do cliente do tipo professor. Cada item da lista de actividades criadas pelo professor, para além de fornecer dados sobre a actividade, tem um botão para mostrar as opções da actividade. Nestas opções, o professor pode aceder a funcionalidades como atribuição de uma actividade a turma, visualização, edição, remoção e *download* da actividade.

A visualização de uma actividade (*Visualizar actividade* na Figura 4.6) tem um funcionamento idêntico à visualização de actividades "encerradas" no cliente do tipo aluno, apenas não sendo possível submeter a solução da actividade.

A remoção de uma actividade (*Eliminar actividade* na Figura 4.6) faz um pedido ao servidor, removendo o ficheiro da actividade no servidor, assim como a sua respectiva referencia na base de dados. Uma actividade apenas pode ser removida se não tiver nenhuma atribuição a turma ou submissão realizada.

A edição de uma actividade (*Editar actividade* na Figura 4.6) utiliza a ferramenta de criação de actividades, preenchendo os campos com os dados actuais da actividade. Ao finalizar o processo, as alterações são enviadas ao servidor e devidamente substituídas.

O *download* de uma actividade (*Download actividade* na Figura 4.6) faz um pedido ao servidor pelo ficheiro da actividade e, quando recebe o ficheiro, faz um pedido ao *browser* para confirmar o *download* para o sistema de ficheiros do utilizador.

A associação de uma actividade (*Atribuir actividade a turma* na Figura 4.6) a uma turma envia um pedido ao servidor com o identificador da turma e os *timestamps* relativos à data de início e data de fim, de forma a ser criada a relação entre a actividade e a turma.

No topo da lista de actividades, o utilizador do tipo professor tem um botão para criar nova actividade (*Criar/upload actividade* na Figura 4.6), acedendo à ferramenta de criação de actividades. No final do processo de criação de actividades, o ficheiro da actividade é enviado ao servidor para processamento.

No separador lateral, o professor pode aceder ao separador de turmas (*Separador turmas* na Figura 4.6). Ao realizar esta acção, o cliente faz um pedido ao servidor para obter todas as turmas criadas pelo professor. Estas turmas são apresentadas numa simples lista. No topo desta lista, existe um botão para criar uma nova turma (*Criar turma* na Figura 4.6). Após introdução dos dados pedidos ao utilizador, estes dados são enviados ao servidor e uma nova turma é criada. Cada item da lista de turmas pode ser pressionado para inspeccionar a turma em detalhe. Ao pressionar numa das turmas (*Inspecionar turma* na Figura 4.6), o cliente pede ao servidor todos os dados relativos à turma como os alunos que pertencem, as actividades associadas e todas as submissões realizadas pelos alunos na turma. Com estes dados são geradas:

- **a lista de actividades associadas à turma**: em que cada actividade pode ser expandida para mostrar os dados relativos à associação e a estatística geral da actividade

relativa às submissões da turma.

- **a tabela de visão geral da turma**: em que cada coluna da tabela tem uma referência para as actividades atribuídas à turma e cada linha expõe o resultado da última submissão do aluno relativa à respectiva actividade (*Visão geral* na Figura 4.6).
- **a lista de alunos pertencentes à turma**: em que cada aluno pode ser expandido, exibindo uma lista com as actividades associadas à turma e o resultado da última submissão. Esta lista de actividades com a última submissão do aluno pode ser interagida, exibindo detalhes de todas as submissões do aluno relativas à actividade (*Feedback individual do aluno* na Figura 4.6).

Neste modo de inspecção de uma turma, o utilizador do tipo professor tem acesso a funcionalidades como adicionar alunos à turma (*Adicionar aluno* na Figura 4.6), remover a associação de um aluno (*Remover aluno* na Figura 4.6) ou actividade (*Remover actividade* na Figura 4.6) à turma e remoção da turma (*Apagar turma* na Figura 4.6).

4.5 Síntese

A aplicação *Bricks* é composta por um servidor *web* que é utilizada por professores e alunos. A comunicação com o servidor utiliza protocolo *HTTPS*. O servidor adopta uma arquitectura *REST*, disponibilizado *endpoints* para permitir aos clientes interagirem com o servidor. O servidor comunica com uma base de dados *PostgreSQL* para armazenar os dados da aplicação. A criação e execução de actividades utilizam o *Google Blockly* para gerar um ambiente de trabalho de programação por blocos. De forma a poder interagir com os ficheiros de entrada das actividades, foi necessário criar novos blocos para leitura.

A aplicação *Bricks* é uma ferramenta que permite a criação e realização de actividades de programação por blocos.

5

Avaliação da aplicação *Bricks*

De forma a avaliar a aplicação Bricks implementada, foi realizada uma avaliação da usabilidade da aplicação. Neste capitulo será descrito a metodologia de testagem, os resultados da testagem e uma análise de resultados.

5.1 Metodologia de avaliação

A avaliação desta dissertação foca-se na usabilidade da interface devido ao alinhamento com o objectivo principal de criar uma aplicação que permita criar e executar actividades de programação por blocos. De forma a identificar potenciais melhorias na interface da aplicação, é importante analisar as interacções dos utilizadores com o cliente da aplicação.

Como dito anteriormente, a aplicação tem dois clientes distintos: professor e aluno. Apesar de terem ambos interfaces *web*, cada tipo de utilizador tem, na aplicação, objectivos distintos.

No caso do professor, a avaliação da aplicação foi separada nas componentes de criação de actividades e gestão de actividades/turma. Esta separação deve-se à necessidade de os candidatos a avaliar a componente de criação de actividades terem que ter algumas competências de programação. Este requisito foi tomado como restrição aos candidatos desta componente pois, para a criação de uma actividade de programação, o utilizador comum terá que ter conhecimentos específicos para criar exercícios de programação e como testar o seu exercício. Para a componente de gestão de actividades/turmas, não existem restrições aos candidatos.

No caso do aluno, não existem restrições aos candidatos para serem eligíveis.

Para poder-se avaliar a usabilidade da aplicação, foram definidos e criados cenários para cada tipo de utilizador. No caso do professor foram criados dois tipos de cenários: criação de actividades e gestão de actividades/turmas. Estes cenários consistem em sequências/conjuntos de tarefas que podem ser consideradas operações comuns na aplicação. Após realização das tarefas, os utilizadores responderam aos questionários respectivos aos cenários que realizaram.

Todos os questionários tem uma sessão comum de perguntas. As questões presentes

nesta última secção são as 10 questões necessárias para calcular a escala de usabilidade do sistema (*System Usability Scale*)[6].

Escala de usabilidade do sistema (SUS)

O cálculo do valor da usabilidade do sistema é realizado obtendo a pontuação destas 10 questões. As questões de índice par são somadas e, de seguida, subtraidas por 5 (número de questões de índice par). As questões de índice impar são também somadas e, de seguida, do valor 25 (valor máximo de pontuação multiplicado por número de questões de índice ímpar) é subtraido o somatório do resultado das questões de índice ímpar. De seguida, ambos os valores obtidos são somados e multiplicados por 2.5, obtendo o valor da usabilidade do sistema para cada participante. Os valores de referência para classificação dos resultados obtidos são baseados na tabela de classificação de Sauro and Lewis[14], ilustrada na Tabela 5.1.

Tabela 5.1: Tabela de classificação de resultados do SUS simplificada

Resultado	Classificação
78.9 - 100	A
72.6 - 78.8	B
62.7 - 72.5	C
51.7 - 62.6	D
0 - 51.6	F

5.1.1 Cenários do professor - Criação de actividades

As seguintes tarefas foram desenhadas para avaliar a usabilidade das operações na aplicação que envolvessem a componente de criação de actividades do utilizador tipo Professor.

Tarefa 1 - Criação de actividade:

Descrição: "O utilizador quer criar uma actividade de programação. Os ficheiros necessários para a criação da actividade encontram-se no seguinte link fornecido ao utilizador. A actividade terá as seguintes características:

- A actividade terá como título "Multiplicação de dois números" Será do tipo "construir".
- Terá como descrição "Escreva um programa que imprime a multiplicação de dois números dados como input".
- O enunciado é o ficheiro que contém a descrição total da actividade e será o ficheiro "Enunciado multiplicação.pdf".

- O recurso é o material de apoio a esta actividade e encontra-se no ficheiro "TipsForCreatingABlockLanguage.pdf".

- Os testes, que servirão para validação das funcionalidades de uma dada solução, poderão ter um nome à escolha do utilizador e os ficheiros a inserir serão os pares de input e output com o mesmo número.

- Os blocos a seleccionar estão descritos na tabela fornecida ao utilizador.

O cenário termina quando o utilizador, após guardar a actividade, volta à área do professor na aplicação."

Objectivo: Este cenário tem como objetivo avaliar a interface de criação de actividades, uma das operações mais importantes da aplicação *Bricks*.

Tarefa 2 - Visualização e edição da solução de uma actividade:

Descrição: "O utilizador quer visualizar e editar uma solução para a actividade que criou na tarefa anterior. Caso não tenha consigo criar a actividade, use a actividade ´´HelloWorld", já existente. O cenário termina quando o utilizador, após guardar a sua solução da actividade, volta à área do professor na aplicação."

Objectivo: O objectivo deste cenário é avaliar, desde o ponto de vista do professor a edição de soluções à atividade, assim como a visualização da mesma.

Tarefa 3 - Fazer download da actividade:

Descrição: "O utilizador quer realizar o ´´download" da actividade que acabou de criar. Ou caso não tenha conseguido criar, pode fazer download de uma actividade existente. O cenário termina quando o utilizador tiver o ficheiro zip da actividade no seu disco local."

Objectivo: Esta tarefa tem como objectivo avaliar se o utilizador acha útil o propósito da aplicação permitir aos utilizadores fazer download de actividades.

5.1.2 Cenários do professor - Gestão de actividades/turmas

Complementando a componente de criação de actividades, as seguintes tarefas são relativas à componente de gestão de actividades/turmas, abrangindo assim as restantes funcionalidades da aplicação relativas ao utilizador tipo Professor.

Tarefa 1 - Criação de turma:

Descrição: "O utilizador quer criar uma nova turma para agrupar os seus alunos e poder atribuir actividades. O nome e descrição da turma serão à escolha do utilizador e os alunos pertencentes à turma estão disponíveis no link fornecido. O cenário termina quando o utilizador, após criação da turma, verifica que a nova turma está criada na área do professor na aplicação."

Objectivo: Esta tarefa permite avaliar se a interface de criação de turma é fácil de utilizar.

Tarefa 2 - Criação de actividade através dum zip:

Descrição: "O utilizador quer adicionar uma actividade à aplicação. O utilizador transfere uma das actividades (um dos ficheiros *.zip) presente na pasta que se encontra no link fornecido. O cenário termina quando o utilizador identifica a nova actividade na sua lista."

Objectivo: O objectivo de incluir esta tarefa é avaliar a utilidade do método alternativo de adicionar actividades à aplicação sem ser necessário conhecimentos prévios sobre construção de actividades de programação e analisar se o processo é intuitivo.

Tarefa 3 - Associação de actividade à turma:

Descrição: "O utilizador atribuir a actividade criada, ou outra lá existente à turma que criou. A data de início será o dia e hora presente e a data de fim será dois dias depois da data de início. O cenário termina quando o utilizador, depois de associar a actividade à turma, verifica que a turma tem a actividade associada."

Objectivo: Esta tarefa tem como objectivo avaliar o processo de atribuição de actividades a turmas.

Tarefa 4 - Visualizar número de alunos que completaram uma actividade:

Descrição: "O utilizador quer visualizar se já houve alunos a completar a actividade "HelloWorld"que atribuiu à turma "Turma 1". O cenário termina quando o utilizador conseguir obter os dados que precisa."

Objectivo: O objectivo desta tarefa é avaliar se o processo para alcançar os dados necessários é demasiado complexo e se os dados necessários têm um formato apelativo e fácil de compreender.

Tarefa 5 - Visualização do estado global de uma turma:

Descrição: "O utilizador quer visualizar globalmente a turma "Turma 1". O cenário termina quando o utilizador conseguir obter os dados que precisa."

Objectivo: Semelhante ao objectivo da tarefa anterior, o objectivo é avaliar se esta forma de visualizar os dados é de fácil compreensão e vantajosa para o utilizador.

5.1.3 Cenários do aluno

As seguintes tarefas têm como objectivo determinar a usabilidade do cliente tipo Aluno e constituem operações comuns para este tipo de utilizador.

Tarefa 1 - Alteração da password:

Descrição: "O utilizador quer modificar a palavra-chave de acesso à sua conta. Depois de entrar na aplicação Bricks com as credenciais dadas fornecidas ao utilizador, este deve proceder à alteração da palavra-chave. A nova palavra-chave deverá ser "pass". O cenário termina no final deste processo."

Objectivo: Esta tarefa constitui uma operação comum para o utilizador do tipo Aluno pois quando a criação das contas de aluno é feita pelo utilizador do tipo professor, as contas são geradas com uma palavra-chave por omissão, tornando-se assim, idealmente, a primeira operação que um novo utilizador deveria realizar. Esta tarefa tem o objectivo avaliar se o processo de alteração de palavra-chave é intuitivo.

Tarefa 2 - Realização de actividade:

Descrição: "O utilizador quer realizar a actividade de programação "Multiplicação de dois números". O cenário termina quando o utilizador, após uma submissão, volta à sua área na aplicação."

Objectivo: Esta tarefa é uma das operações mais importantes para este tipo de utilizador, constituindo uma das funcionalidades principais da aplicação. Como tal, é importante a análise das interacções dos utilizadores com a interface relativo a esta tarefa. O objectivo desta tarefa consiste em avaliar o ambiente fornecido ao utilizador para obtenção de dados relativos à actividade, para edição da solução da actividade e de *feedback* individual dado no momento de avaliação do programa/solução.

Tarefa 3 - Consultar actividade encerrada (Histórico):

Descrição: "O utilizador quer consultar uma actividade de programação que se encontra encerrada. O cenário termina quando o utilizador, depois de visualizar a actividade, volta à área do aluno na aplicação."

Objectivo: O objectivo desta tarefa é avaliar se, na aplicação, o acesso a actividades que já estejam encerradas é um processo intuitivo e útil para os alunos.

Tarefa 4 - Consultar turmas:

Descrição: "O utilizador quer consultar as turmas ao qual está inserido para confirmar se pertence à Turma 1. O cenário termina quando o utilizador confirma que pertence à Turma 1."

Objectivo: Esta tarefa tem como objectivo avaliar a utilidade do separador das turmas no cliente do tipo Aluno, assim como determinar se os dados presentes nesta interface têm um formato intuitivo.

5.2 Resultados da avaliação

Nesta secção são apresentados os resultados dos cenários avaliados no sistema *Bricks*, juntamente com as respectivas caracterizações do grupo de teste. Cada cenário tem um questionário associado, que segue a seguinte estrutura:

- Caracterização do utilizador
- Questões relativas às tarefas do cenário
- Questões relativas à aplicação
- Questões relativas à usabilidade da aplicação

Os questionários usados encontram-se no anexo A.

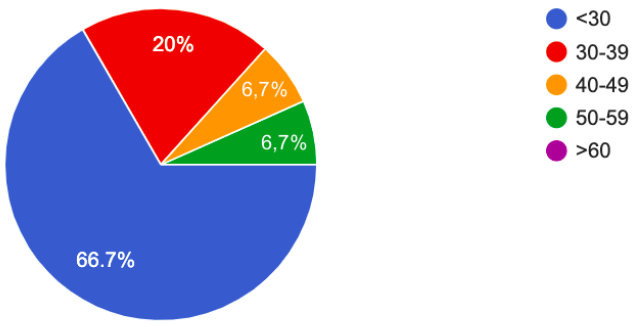
5.2.1 Componente de criação de actividades

Esta subsecção é dedicada à apresentação dos resultados dos cenários do professor para a componente de criação de actividades.

5.2.1.1 Caracterização do grupo de teste

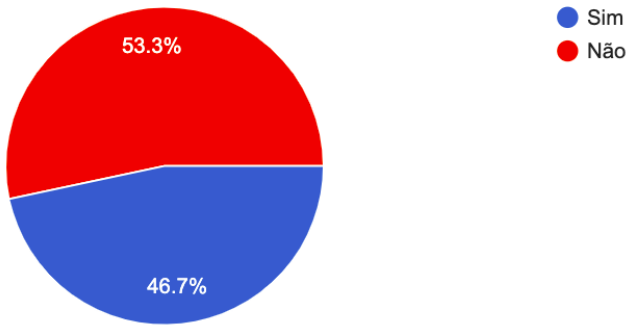
Este grupo de teste foi composto por 15 participantes. Os participantes tinham idades entre os 25 e os 60 anos, apresentadas pelo gráfico exibido na Figura 5.1.

Figura 5.1: Gráfico circular com os dados relativos à idade dos participantes



Os participantes neste grupo tinham experiência em programação, porém, nem todos tinham experiência a leccionar programação. Dos participantes deste grupo de teste, apenas cerca de 45% já foi professor de uma disciplina curricular de programação. Estes dados podem ser observados na Figura 5.2.

Figura 5.2: Gráfico circular com os dados relativos à experiência a leccionar dos participantes

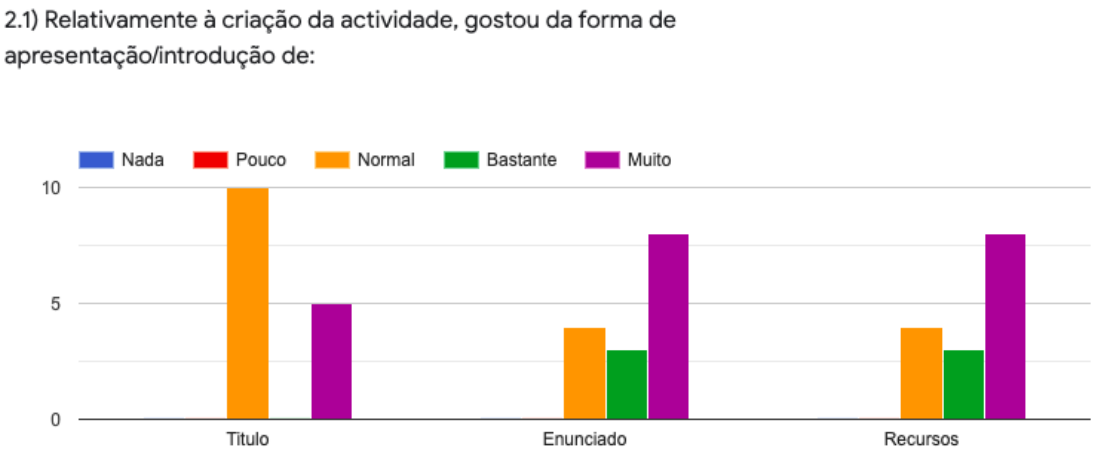


5.2.1.2 Questões relativas às tarefas do cenário

De forma a obter informação sobre a utilização da aplicação, no contexto do cenário do professor para a criação de actividades, os participantes foram questionados sobre as tarefas realizadas.

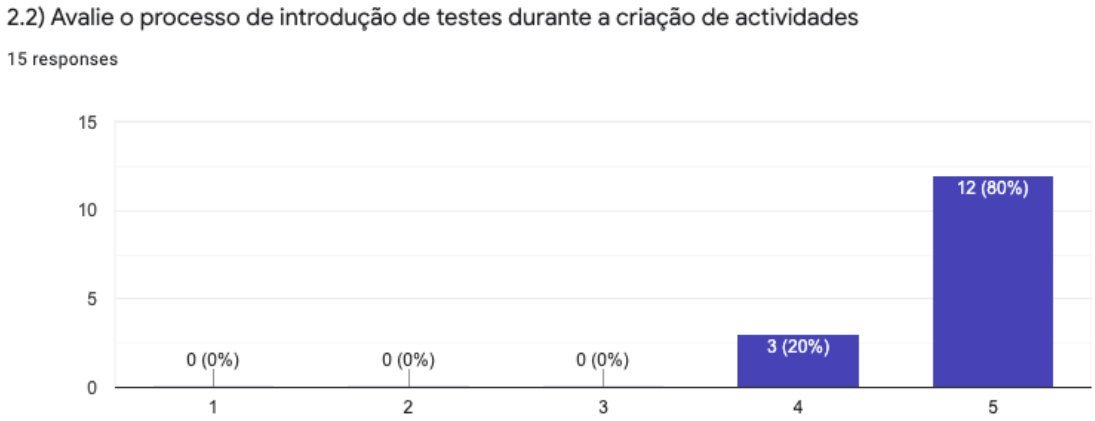
Os resultados da avaliação referente à introdução do título, enunciado e recursos adicionais na criação duma atividade são apresentados na Figura 5.3. Estes resultados indicam que a forma de introdução de ficheiros implementada é intuitiva e proporciona uma boa experiência de utilização.

Figura 5.3: Respostas obtidas para apresentação/introdução de titulo, enunciado e recursos



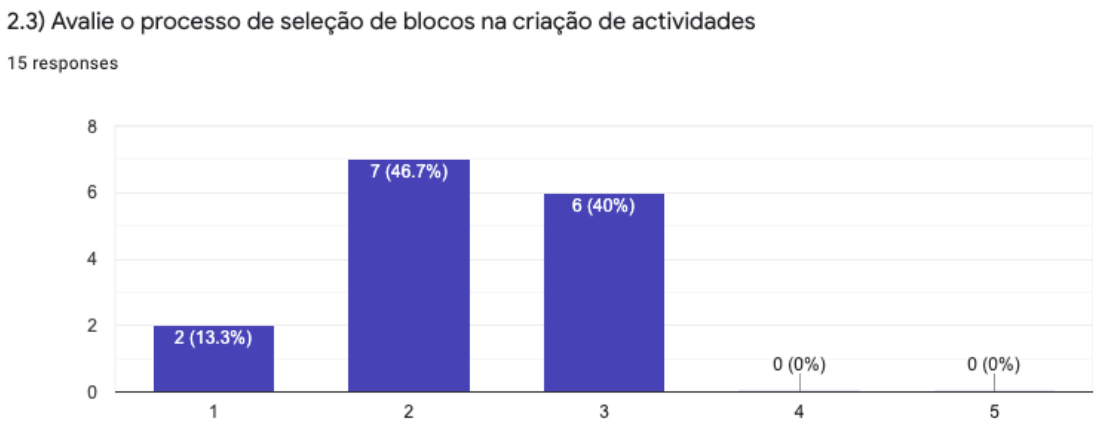
Outra avaliação realizada foi se o processo de introdução de testes durante a criação de uma actividade seria simples ou se seria demasiado complicado. A escala das respostas nesta avaliação iniciava-se em "Muito difícil" (1) escalando a "Muito fácil" (5). Na Figura 5.4 são apresentados os resultados obtidos, os quais indicam que o procedimento de introdução de testes é apropriadamente simples.

Figura 5.4: Respostas obtidas para a introdução dos testes da actividade



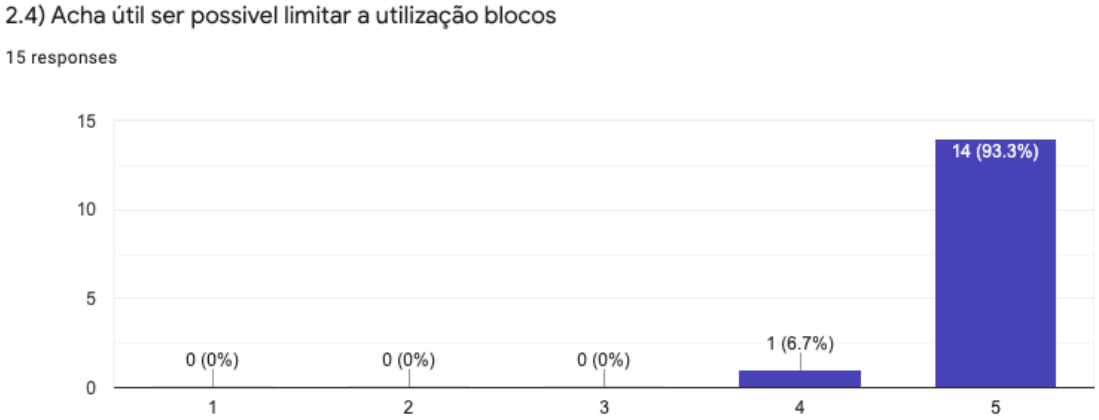
Avaliou-se também se o processo de selecção dos blocos durante a criação de uma actividade seria um processo apropriadamente complexo. A escala das respostas a esta questão iniciava-se em "Muito difícil" (1) escalando a "Muito fácil" (5). Na Figura 5.5 são apresentados os resultados obtidos, os quais mostram que os participantes sentiram dificuldades durante esta parte da tarefa o que indica que o processo de selecção de blocos deve ser melhorado.

Figura 5.5: Respostas obtidas para a selecção de blocos na criação da actividade



A utilidade da funcionalidade de limitar a utilização de blocos de uma actividade, usou-se tal a escala das respostas a esta questão iniciava-se em "Pouco útil" (1) escalando a "Muito útil" (5). Na Figura 5.6 são apresentados os resultados, os quais foram favoráveis, o que indica que os participantes reconheceram positivamente esta funcionalidade.

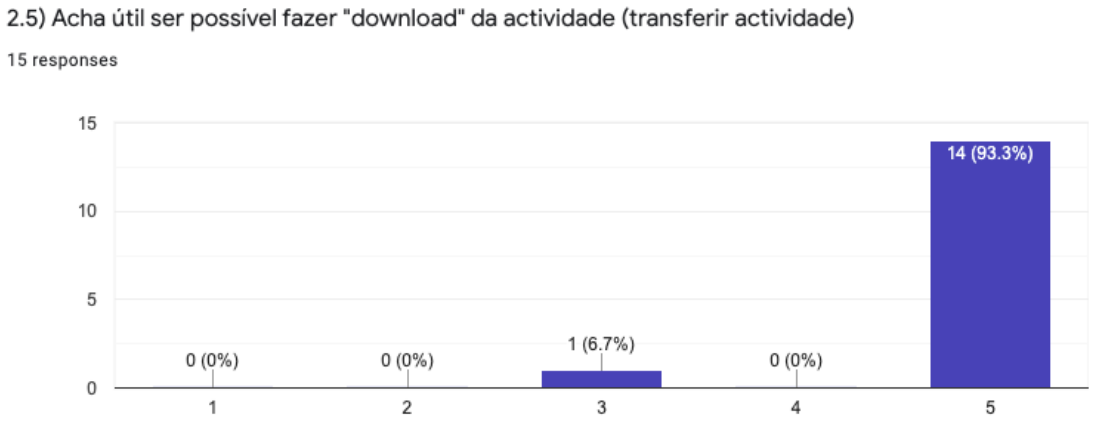
Figura 5.6: Respostas obtidas para a limitação dos blocos na actividade



Quanto à avaliação da utilidade da funcionalidade de fazer *download* de actividades, usou-se uma escala que iniciava-se em "Pouco útil" (1) escalando a "Muito útil" (5). Na Figura 5.7 são apresentados os resultados os quais foram favoráveis, o que indica que os participantes reconheceram positivamente esta funcionalidade.

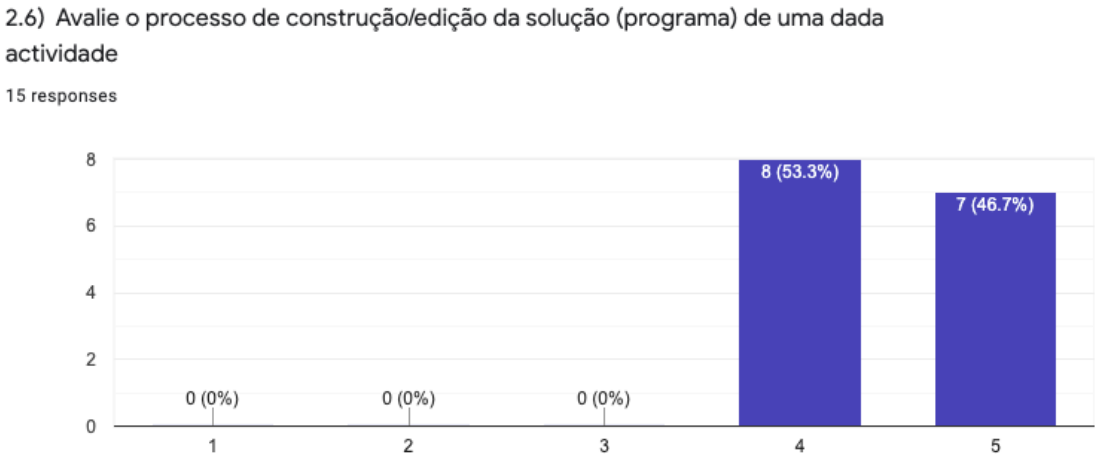
A última avaliação realizada neste cenário constava de avaliar a utilização de um ambiente de programação por blocos seria intuitivo e fácil de utilizar, do ponto de vista do professor. A escala das respostas a esta questão iniciava-se em "Muito difícil" (1) escalando a "Muito fácil" (5). Na Figura 5.8 são apresentados os resultados obtidos, os

Figura 5.7: Respostas obtidas para a realizar download da actividade



quais foram favoráveis, indicando que o ambiente de construção de soluções a actividades de programação é intuitivo do ponto de vista do professor.

Figura 5.8: Respostas obtidas para a edição de soluções para a actividade



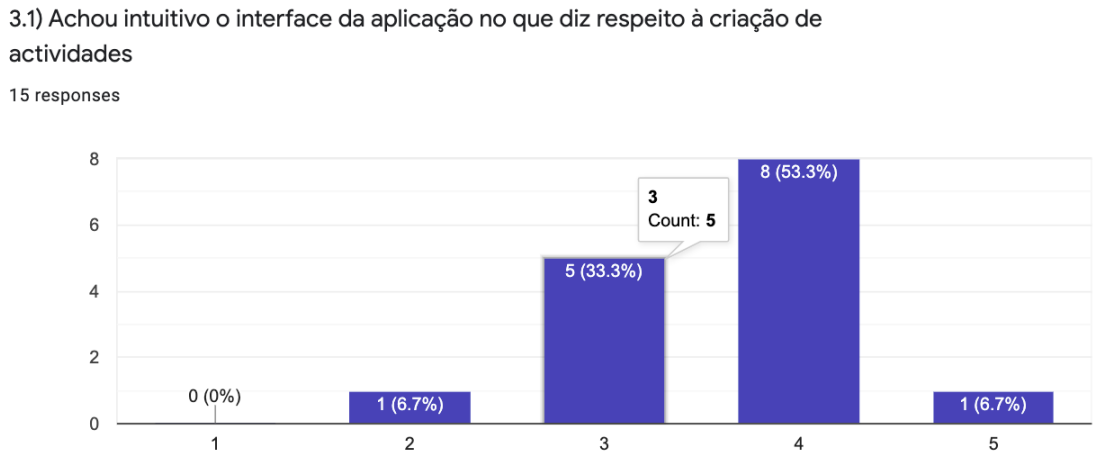
5.2.1.3 Questões relativas à aplicação

De forma a obter informação geral sobre a utilização da aplicação, os participantes do cenário de teste foram questionados sobre a sua experiência de utilização da aplicação. Ao contrário do conjunto de questões anterior, estas questões não se focam apenas nas tarefas do cenário.

Os resultados da avaliação referente à criação de actividades de programação na aplicação é um processo intuitivo. A escala das respostas a esta avaliação iniciava-se em "Pouco intuitivo" (1) escalando a "Muito intuitivo" (5). Na Figura 5.9 são apresentados os

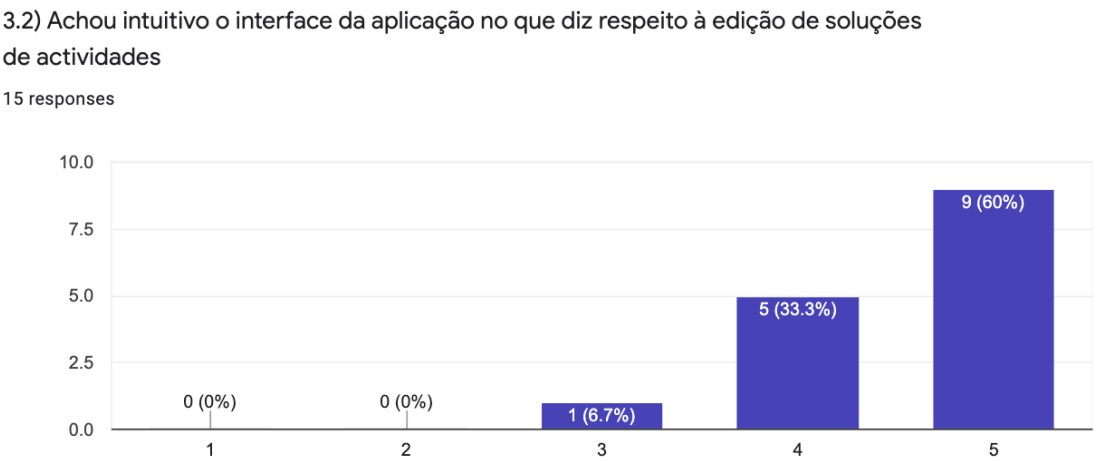
resultados obtidos, os quais indicam que a interface da aplicação é intuitivo, com algum espaço para melhorias.

Figura 5.9: Respostas obtidas para a avaliação da interface no que diz respeito à criação de actividades



Outra avaliação realizada foi se a edição de soluções a actividades de programação na aplicação é um processo intuitivo. A escala das respostas a esta avaliação iniciava-se em "Pouco intuitivo" (1) escalando a "Muito intuitivo" (5). Na Figura 5.10 são apresentados os resultados obtidos, os quais mostram que foram favoráveis, indicando que o processo de edição de soluções a actividades de programação é intuitivo.

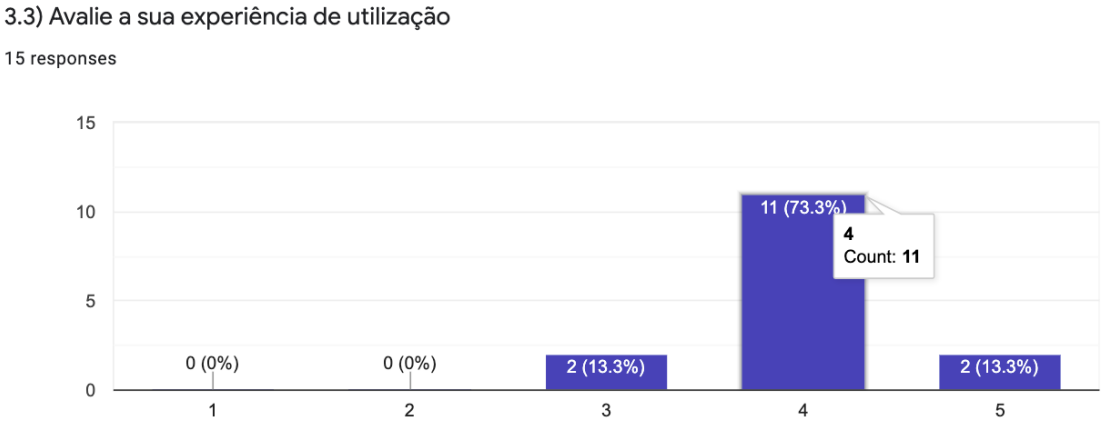
Figura 5.10: Respostas obtidas para a avaliação da interface no que diz respeito à edição de soluções



Avaliou-se também se a experiência de utilização da aplicação, enquanto realizava o

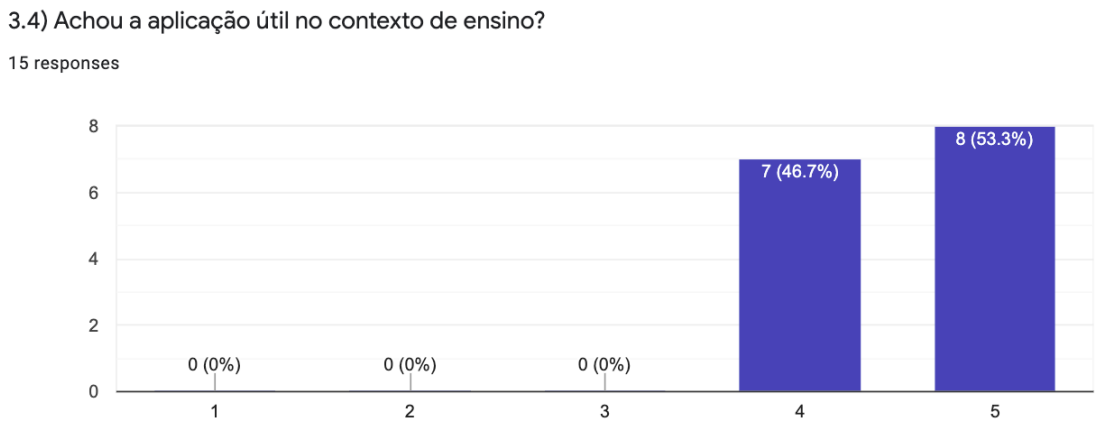
cenário, foi positiva. A escala das respostas a esta avaliação iniciava-se em "Não gostei" (1) escalando a "Gostei muito" (5). Na Figura 5.11 são apresentados os resultados obtidos, os quais indicam que os participantes tiveram uma experiência de utilização favorável, porém, segundo resultados a questões anteriores, os resultados terão sido condicionados pela experiência menos favorável proporcionada pela dificuldade de selecção de blocos durante a criação da actividade de programação proposta no cenário.

Figura 5.11: Respostas obtidas para a avaliação da experiência de utilização



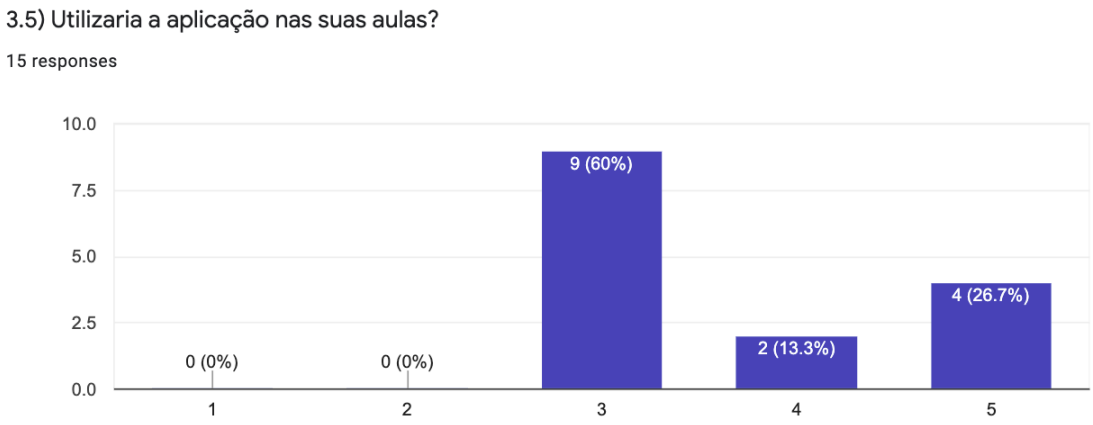
Avaliou-se também se a existe utilidade para esta aplicação, num contexto real. A escala das respostas a esta avaliação iniciava-se em "Pouco útil" (1) escalando a "Muito útil" (5). Na Figura 5.12 são apresentados os resultados obtidos, os quais mostram que os participantes reconhecem aplicação prática desta aplicação no contexto do ensino.

Figura 5.12: Respostas obtidas para a utilidade da aplicação no contexto de ensino



A última avaliação realizada neste cenário constava de avaliar se, conceptualmente, esta aplicação poderia ser adoptada como ferramenta no ensino. A escala das respostas a esta avaliação iniciava-se em "Nunca" (1) escalando a "Muitas vezes" (5). Na Figura 5.13 são apresentados os resultados obtidos, os quais indicam que os participantes reconhecem alguma aplicabilidade dentro do contexto específico a cada participante.

Figura 5.13: Respostas obtidas para a opinião pessoal de aplicabilidade



5.2.1.4 Questões relativas à usabilidade da aplicação

Após realização do cálculo da usabilidade do sistema para cada participante utilizando os resultados da última secção do questionário, procedemos a uma análise destes dados. Na Figura 5.14 são apresentados os resultados do cálculo do valor de usabilidade extraído do questionário de cada participante. Com estes dados foi obtida uma média para os resultados de 91.83 pontos e uma mediana de 90 pontos.

Utilizando a tabela de referência (ver Tabela 5.1), os resultados obtidos foram favoráveis, encontrando-se na melhor classificação. A Figura 5.15 apresenta a contagem de resultados de participantes por classificação.

5.2.2 Componente de gestão de actividades/turmas

Esta subsecção é dedicada à apresentação dos resultados dos cenários do professor para a componente de gestão de actividades e turmas.

5.2.2.1 Caracterização do grupo de teste

Este grupo de teste foi composto por 15 participantes. Os participantes tinham idades entre os 25 e os 60 anos, apresentadas pelo gráfico exibido na Figura 5.16.

Considerando a avaliação da componente de gestão de actividades e turmas, foi questionado aos participantes se já teriam utilizado algum ambiente com funcionalidades de

Figura 5.14: Resultados do cálculo do valor de usabilidade de cada participante para o cenário do professor (Criação de actividades)

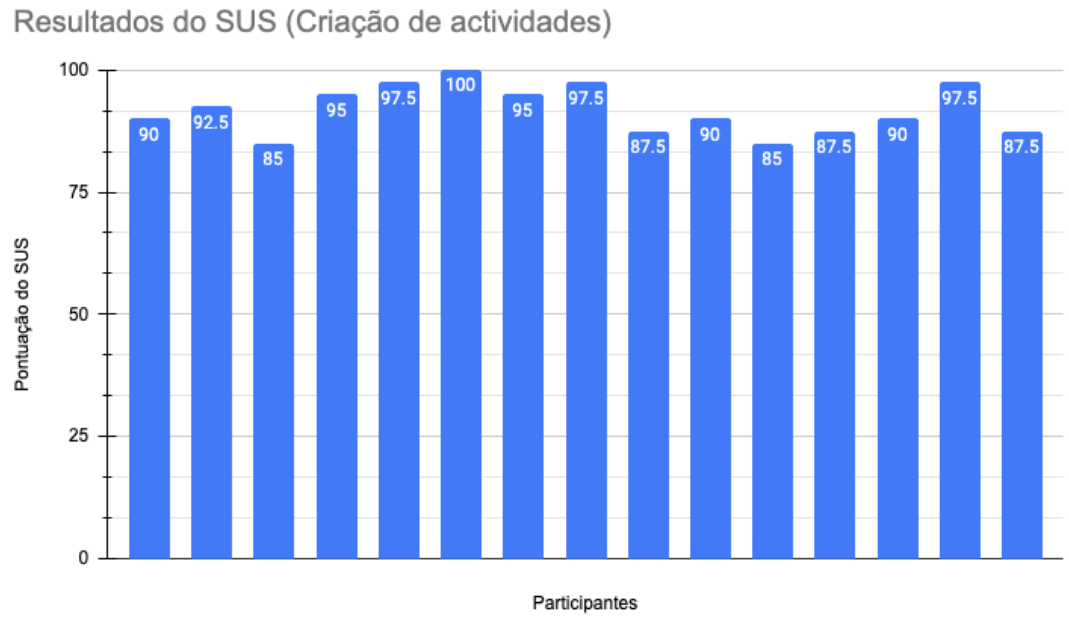


Figura 5.15: Contagem de resultados de participantes por classificação obtida para o cenário do professor (Criação de actividades)

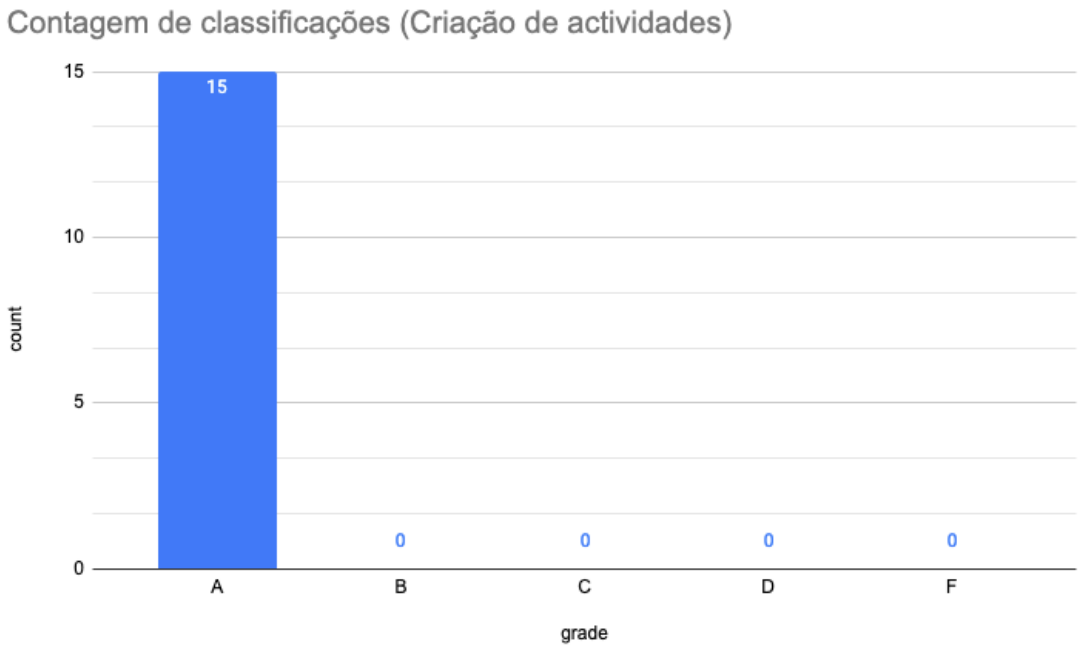
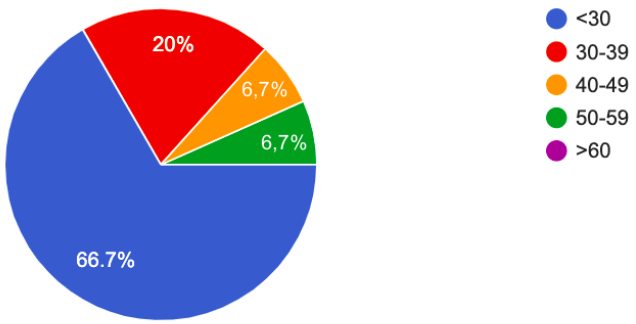
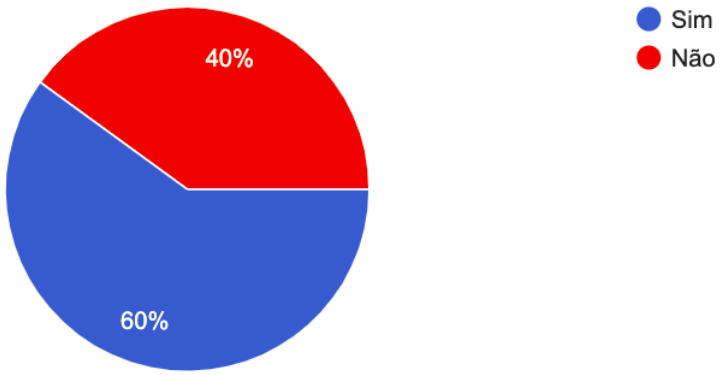


Figura 5.16: Gráfico circular com os dados relativos à idade dos participantes



gestão de exercícios ou conjuntos de alunos como o *Moodle* ou o *Mooshak*. Como pode ser observado no gráfico da Figura 5.17, 40% dos participantes já teve experiência de utilização de algum ambiente com funcionalidades de gestão de exercícios ou turmas.

Figura 5.17: Gráfico circular com os dados relativos à experiência dos participantes relativa a ambientes de gestão/atribuição de actividades a alunos



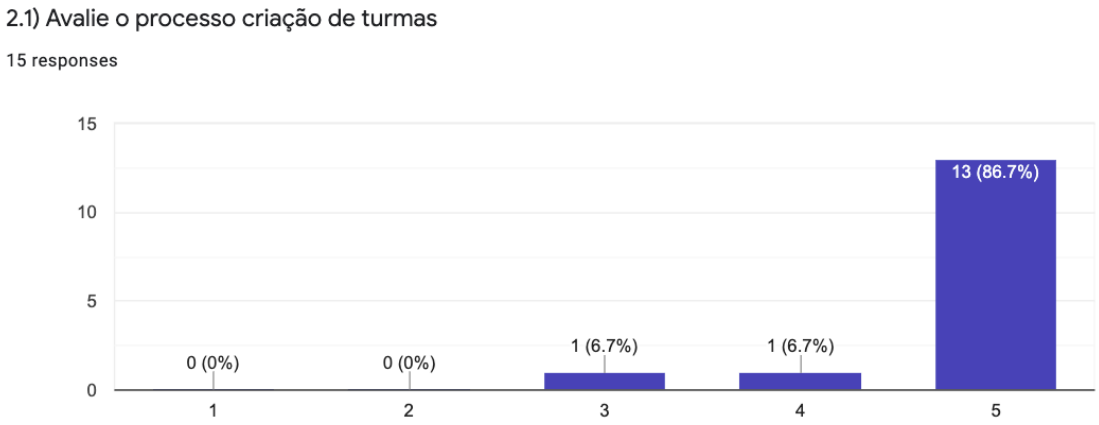
5.2.2.2 Questões relativas às tarefas do cenário

De forma a obter informação sobre a utilização da aplicação, no contexto do cenário do professor para a gestão de turmas e actividades, os participantes foram questionados sobre as tarefas realizadas.

Foi avaliado se o processo de criação de uma turma seria uma experiência positiva de utilização. A escala das respostas a esta avaliação iniciava-se em "Pouco intuitivo" (1) escalando a "Muito intuitivo" (5). Na Figura 5.18 são apresentados os resultados

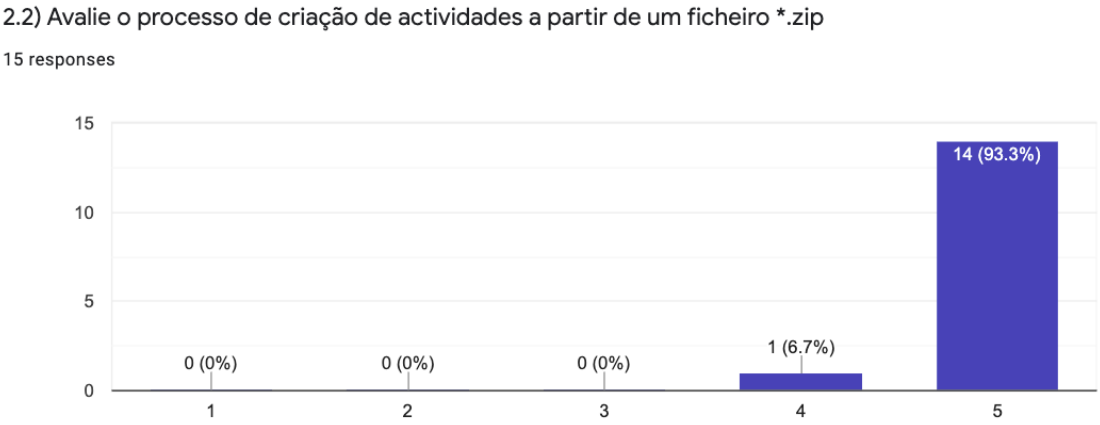
obtidos, os quais indicam que o processo de criação de turmas implementado é intuitivo e proporciona uma boa experiência de utilização.

Figura 5.18: Respostas obtidas para a avaliação do processo de criação de turmas



Avaliou-se também se a criação de actividades a partir de um arquivo de extensão ZIP, que contenha uma actividade, seria um processo demasiado complexo. A escala das respostas a esta avaliação iniciava-se em "Muito difícil" (1) escalando a "Muito fácil" (5). Na Figura 5.19 são apresentados os resultados obtidos, os quais indicam que o processo de criação de actividades a partir de um ficheiro implementado é simples e intuitivo.

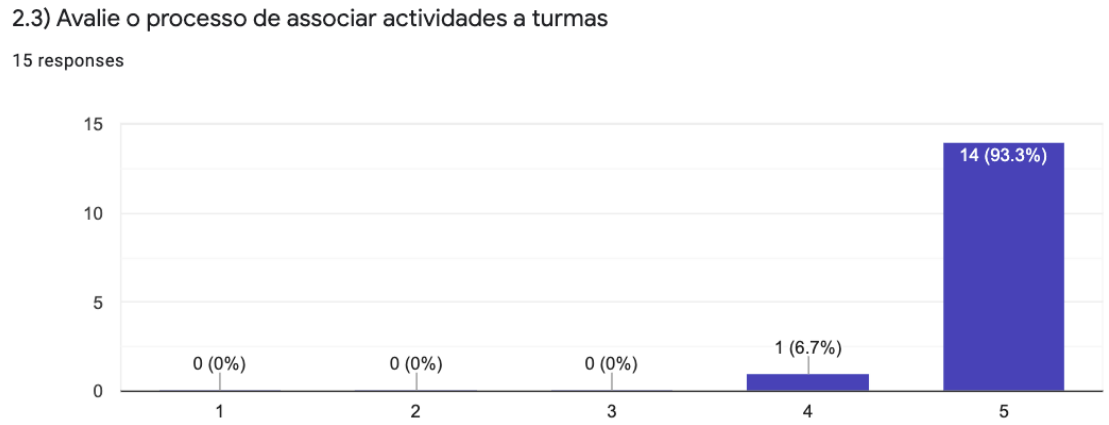
Figura 5.19: Respostas obtidas para a avaliação do processo de criação de actividades a partir do ficheiro



Outra avaliação realizada foi determinar a dificuldade do processo de atríbuição de uma actividade a uma turma. A escala das respostas a esta avaliação iniciava-se em "Muito difícil" (1) escalando a "Muito fácil" (5). Na Figura 5.20 são apresentados os resultados

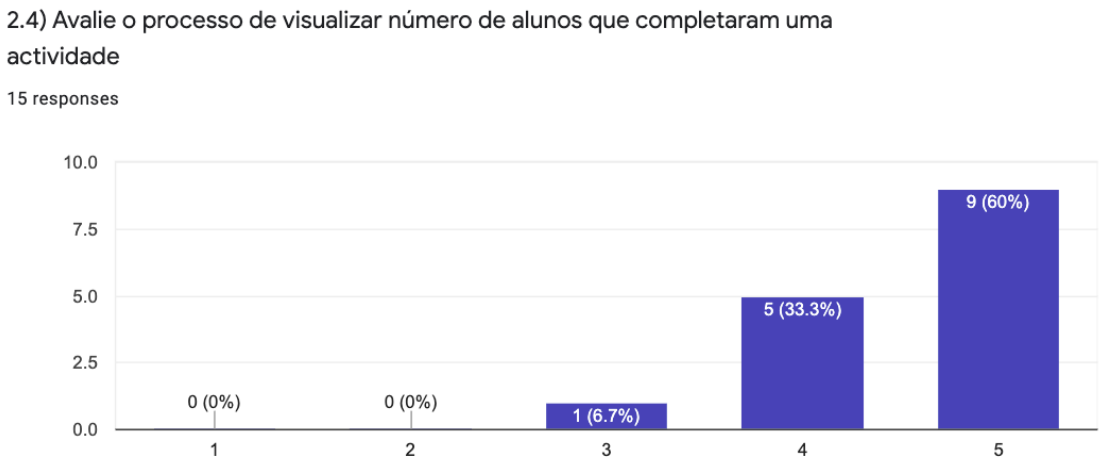
obtidos, os quais mostram que a implementação do processo para atríbuir actividades a uma turma não foi alvo de dificuldades pelos participantes do cenário, indicando que o processo não é demasiado complexo.

Figura 5.20: Respostas obtidas para a avaliação do processo associação de uma actividade a uma turma



Também se realizou avaliação para determinar se a visualização dos dados das actividades relativos a uma turma seria um processo pouco intuitivo. A escala das respostas a esta avaliação iniciava-se em "Muito difícil" (1) escalando a "Muito fácil" (5). Na Figura 5.21 são apresentados os resultados obtidos, os quais indicam que o processo de visualizar os dados das actividades relativos a uma turma implementado é simples e intuitivo.

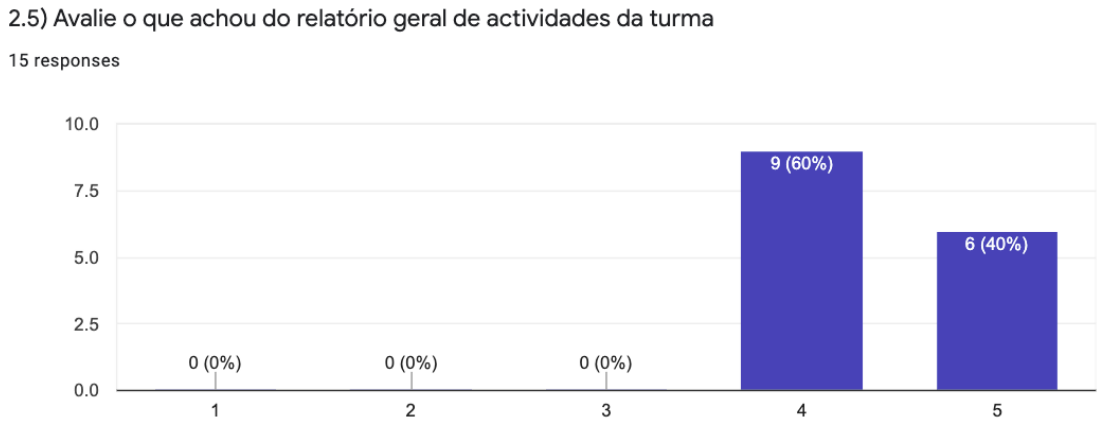
Figura 5.21: Respostas obtidas para o processo de visualização de alunos a completar uma actividade



Avaliou-se também se a funcionalidade de visualizar um relatório geral de uma turma,

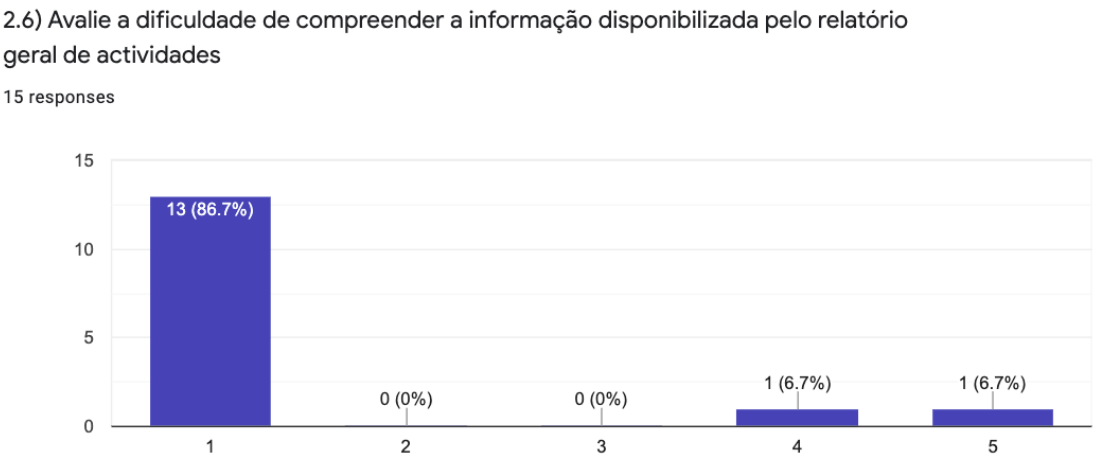
contendo uma tabela exibindo que alunos fizeram quais actividades, seria vantajosa para os utilizadores do tipo professor. A escala das respostas a esta avaliação iniciava-se em "Pouco útil" (1) escalando a "Muito útil" (5). Na Figura 5.22 são apresentados os resultados obtidos, os quais mostram que os utilizadores reconheceram a utilidade desta funcionalidade.

Figura 5.22: Respostas obtidas para opinião sobre o relatório geral da turma



A última avaliação realizada neste secção constava de availar a dificuldade de compreender os dados presentes na tabela do relatório geral da turma. A escala das respostas a esta avaliação iniciava-se em "Pouco difícil" (1) escalando a "Muito difícil" (5). Na Figura 5.23 são apresentados os resultados obtidos, os quais indicam que os dados da tabela do relatório geral da turma são apresentados num formato simples e de fácil compreensão.

Figura 5.23: Respostas obtidas para a avaliação da informação disponibilizada pelo relatório geral

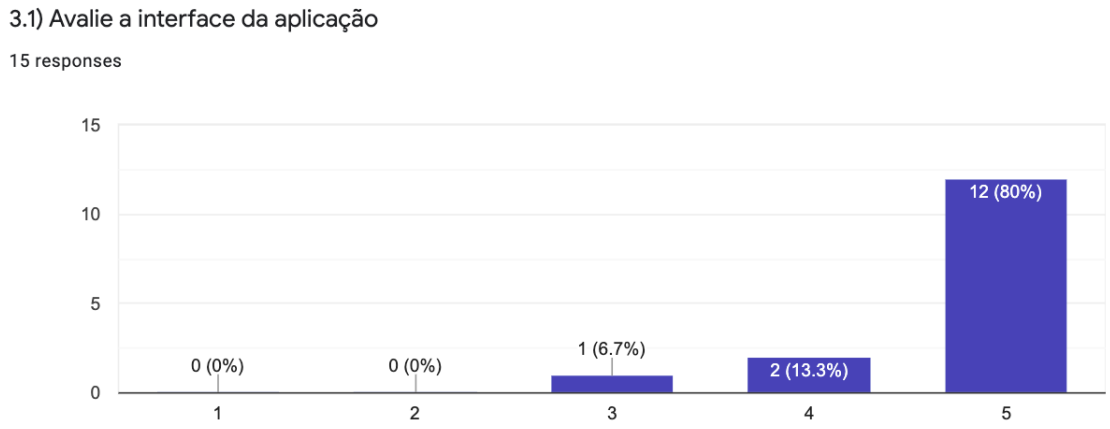


5.2.2.3 Questões relativas à aplicação

De forma a obter informação geral sobre a utilização da aplicação, os participantes do cenário de teste foram questionados sobre a sua experiência de utilização da aplicação. Ao contrário do conjunto de questões anterior, estas questões não se focam apenas nas tarefas do cenário.

Foi avaliado se a interface da aplicação iria impedir uma experiência positiva de utilização. A escala das respostas a esta avaliação iniciava-se em "Pouco intuitivo" (1) escalando a "Muito intuitivo" (5). Na Figura 5.24 são apresentados os resultados obtidos, os quais foram favoráveis, indicando que a interface da aplicação é intuitivo.

Figura 5.24: Respostas obtidas para a avaliação da interface da aplicação



Avaliou-se também se a experiência de utilização da aplicação, enquanto realizava o cenário, foi positiva. A escala das respostas a esta avaliação iniciava-se em "Não gostei" (1) escalando a "Gostei muito" (5). Na Figura 5.25 são apresentados os resultados obtidos, os quais mostram que os participantes tiveram uma experiência de utilização favorável.

A última avaliação realizada neste secção constava de availar se a existe utilidade para esta aplicação, num contexto real. A escala das respostas a esta avaliação iniciava-se em "Pouco útil" (1) escalando a "Muito útil" (5). Na Figura 5.26 são apresentados os resultados obtidos, os quais indicam que os participantes reconhecem aplicação prática desta aplicação no contexto do ensino.

5.2.2.4 Questões relativas à usabilidade da aplicação

Após realização do cálculo da usabilidade do sistema para cada participante utilizando os resultados da última secção do questionário, procedemos a uma análise destes dados. Na Figura 5.27 são apresentados os resultados do cálculo do valor de usabilidade extraído do questionário de cada participante. Com estes dados foi obtida uma média para os resultados de 93.83 pontos e uma mediana de 95 pontos.

Figura 5.25: Respostas obtidas para a avaliação da experiência de utilização

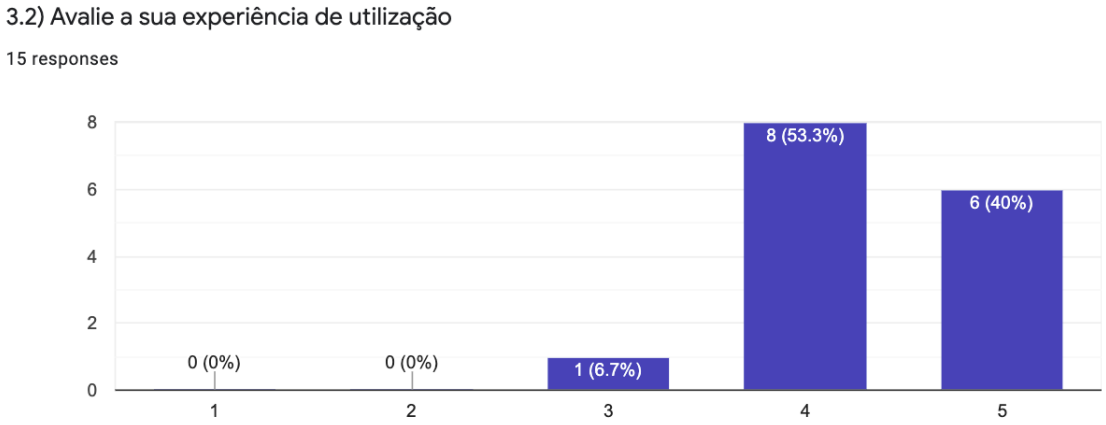
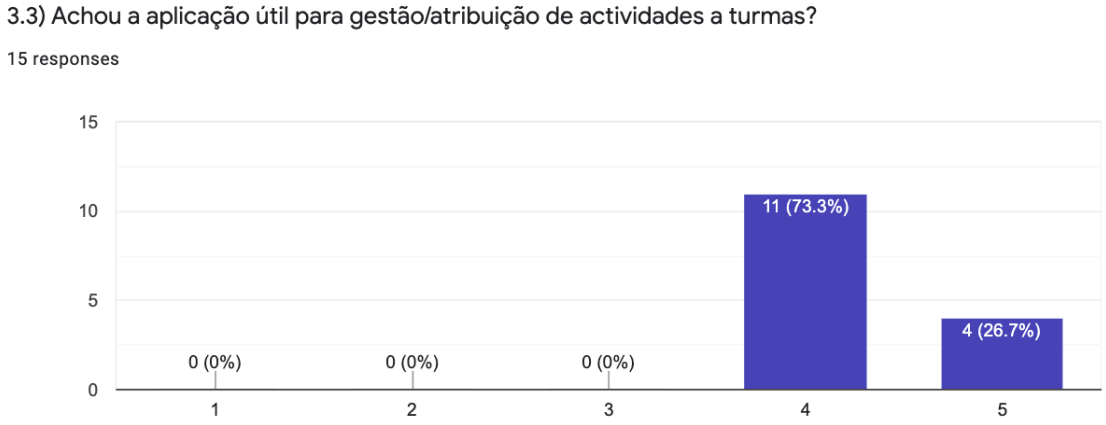


Figura 5.26: Respostas obtidas para opinião sobre a utilidade da aplicação



Utilizando a tabela de referência (ver Tabela 5.1), os resultados obtidos foram favoráveis, encontrando-se maioritáriamente na melhor classificação com a excepção do resultado de um participante que avaliou a aplicação com classificação *B*. A Figura 5.28 apresenta a contagem de resultados de participantes por classificação.

5.2.3 Componente do aluno

Esta subsecção é dedicada à apresentação dos resultados dos cenários do aluno.

5.2.3.1 Caracterização do grupo de teste

Este grupo de teste foi composto por 25 participantes. Os participantes tinham idades entre os 15 e os 60 anos, apresentadas pelo gráfico exibido na Figura 5.29.

Figura 5.27: Resultados do cálculo do valor de usabilidade de cada participante para o cenário do professor (Gestão de actividades/turmas)

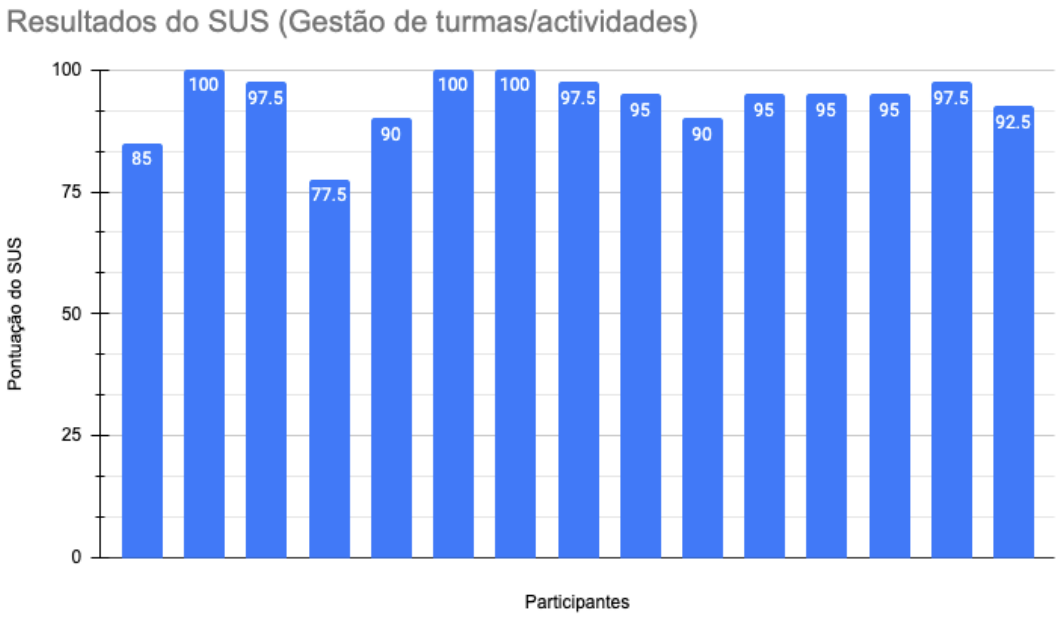


Figura 5.28: Contagem de resultados de participantes por classificação obtida para o cenário do professor (Gestão de actividades/turmas)

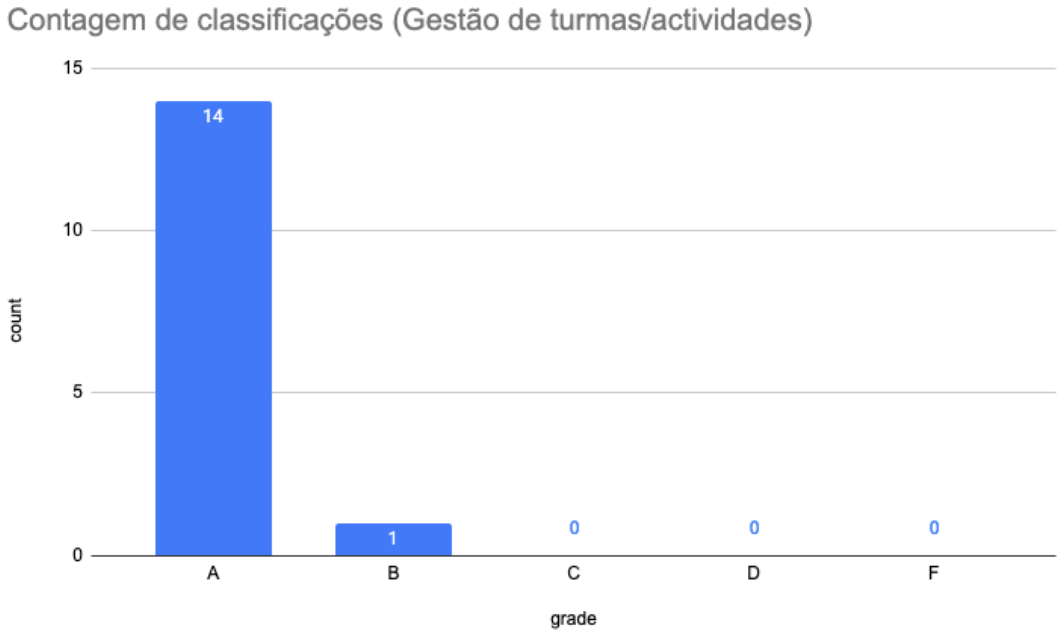
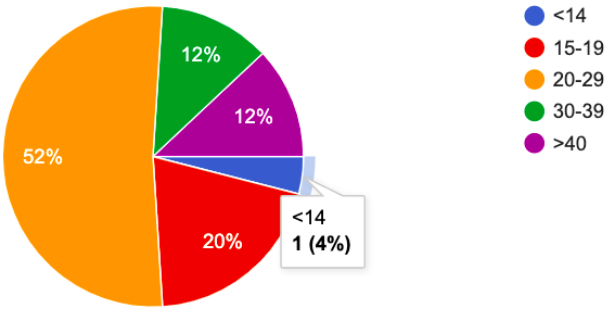
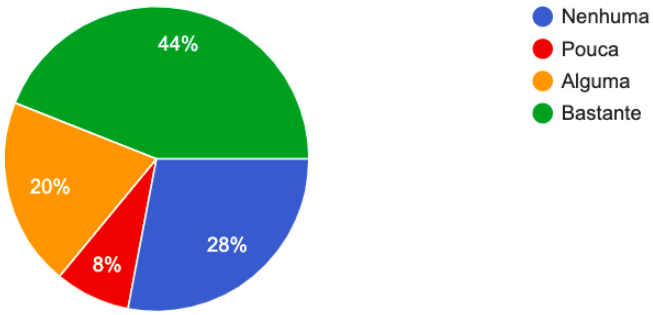


Figura 5.29: Gráfico circular com os dados relativos à idade dos participantes



Para os cenários do aluno, seria útil obter perspectivas de utilização com base na experiência em programação. Para tal, questionámos os participantes deste grupo de teste sobre a sua experiência em programação. Idealmente, o caso óptimo seria ter uma distribuição equitativa sobre a escala usada para determinar a experiência em programação. Na figura 5.30 é exibido o gráfico com a distribuição do grupo de teste em relação à experiência em programação

Figura 5.30: Gráfico circular com os dados relativos à experiência dos participantes em programação



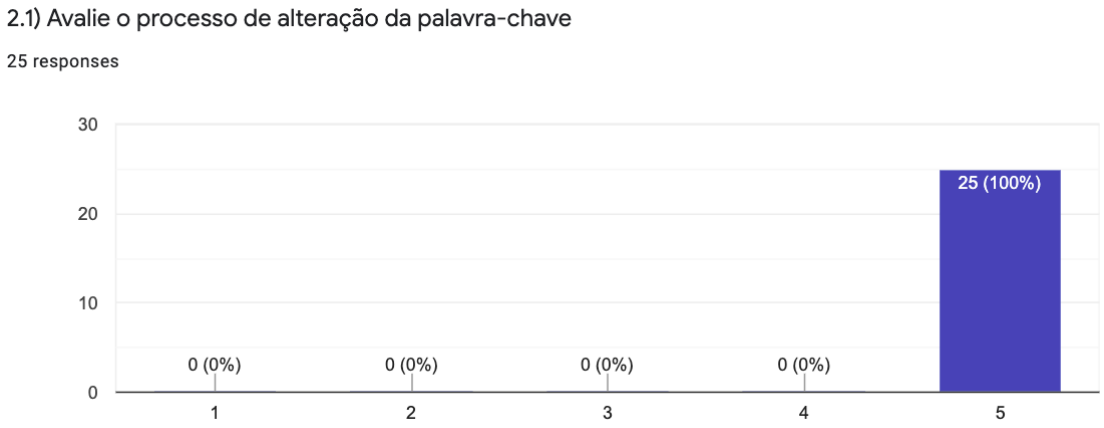
5.2.3.2 Questões relativas às tarefas do cenário

De forma a obter informação sobre a utilização da aplicação, no contexto do cenário do aluno, os participantes foram questionados sobre as tarefas realizadas.

As contas do tipo aluno, na aplicação, podem ser criadas por utilizadores do tipo professor. Estas contas são geradas com uma palavra-chave por omissão. Por utilizar uma palavra-chave por omissão, a alteração da palavra-chave será uma das primeiras acções a

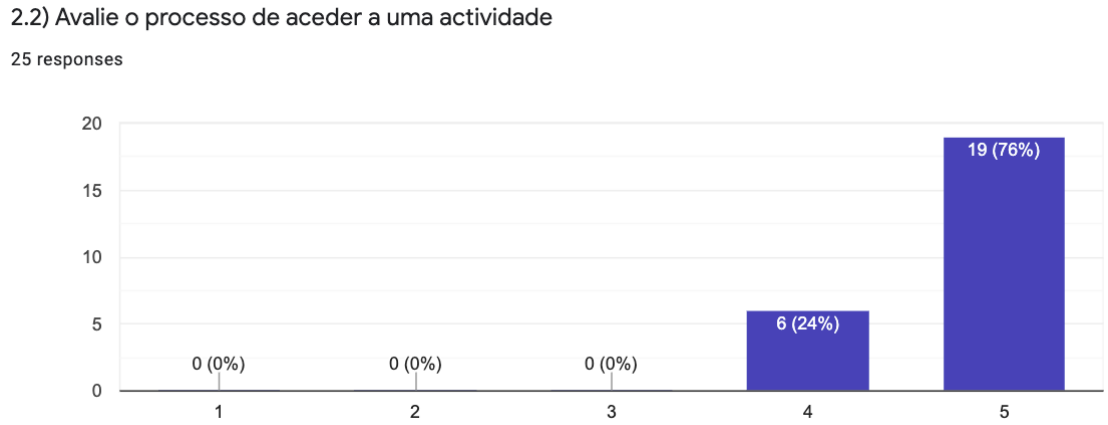
realizar quando um novo utilizador do tipo aluno entra na aplicação. O objectivo desta primeira avaliação é determinar se o processo de alteração de palavra-chave necessita de simplificação. A escala das respostas a esta avaliação iniciava-se em "Muito difícil" (1) escalando a "Muito fácil" (5). . Na Figura 5.31 são apresentados os resultados obtidos, os quais foram muito positivos indicando que o processo de alteração de palavra-chave é simples, intuitivo e não necessita de ser melhorado.

Figura 5.31: Respostas obtidas para a avaliação do processo de alteração da palavra-chave



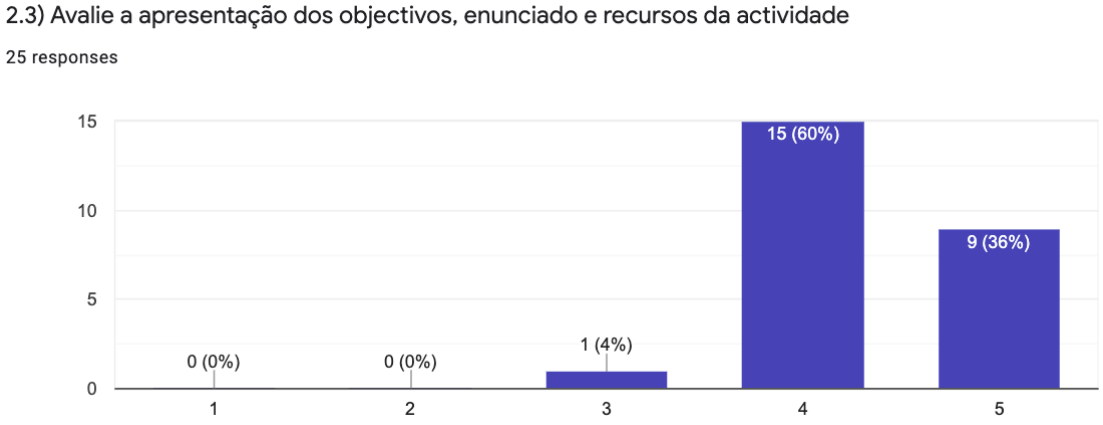
Também se realizou avaliação para availar a dificuldade do utilizador do tipo aluno aceder a uma actividade de programação para a realizar. A escala das respostas a esta avaliação iniciava-se em "Muito difícil" (1) escalando a "Muito fácil" (5). Na Figura 5.32 são apresentados os resultados obtidos, os quais mostram que os utilizadores não encontraram dificuldades a aceder a uma actividade.

Figura 5.32: Respostas obtidas para a avaliação do processo de acesso a uma actividade



Foi também avaliado se o objectivo, o enunciado e os recursos de uma actividade, durante a sua realização, estão exibidos de uma forma intuitiva e de simples acesso. A escala das respostas a esta avaliação iniciava-se em "Pouco intuitivo" (1) escalando a "Muito intuitivo" (5). Na Figura 5.33 são apresentados os resultados obtidos, os quais indicam que os utilizadores consideraram a apresentação do objectivo, enunciado e recursos das actividades intuitivo e de simples utilização.

Figura 5.33: Respostas obtidas para a avaliação da apresentação dos objectivos, enunciado e recursos de uma actividade



Outra avaliação realizada foi de forma a determinar a dificuldade de construção de soluções a actividades utilizando o editor implementado. A escala das respostas a esta avaliação iniciava-se em "Muito difícil" (1) escalando a "Muito fácil" (5). Na Figura 5.34 são apresentados os resultados obtidos, os quais mostram que o processo de construção de soluções é simples, não requerendo uma curva de aprendizagem muito acentuada.

A última avaliação realizada nesta secção teve como objectivo availar a dificuldade do utilizador conseguir distinguir o propósito dos dois separadores diferentes de actividades na aplicação. A escala das respostas a esta avaliação iniciava-se em "Muito difícil" (1) escalando a "Muito fácil" (5). Na Figura 5.35 são apresentados os resultados obtidos, os quais indicam que a maioria os utilizadores compreendeu a diferença entre os dois separadores de actividades, na aplicação.

5.2.3.3 Questões relativas à aplicação

De forma a obter informação geral sobre a utilização da aplicação, os participantes do cenário de teste foram questionados sobre a sua experiência de utilização da aplicação. Ao contrário do conjunto de questões anterior, estas questões não se focam apenas nas tarefas do cenário.

A primeira avaliação desta secção teve como objectivo de compreender se a interface da aplicação iria impedir uma experiência positiva de utilização. A escala das respostas a

Figura 5.34: Respostas obtidas para a avaliação do processo de construção/edição da solução de uma dada actividade

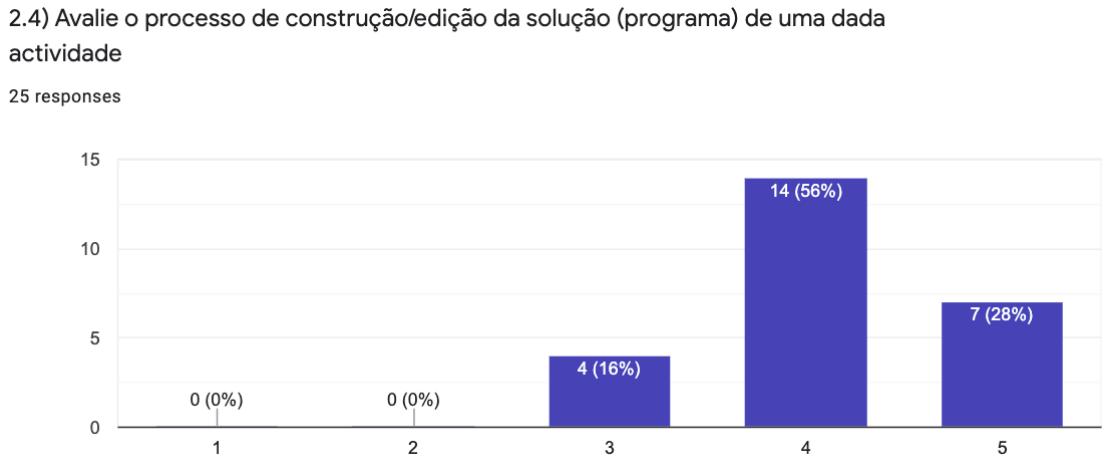
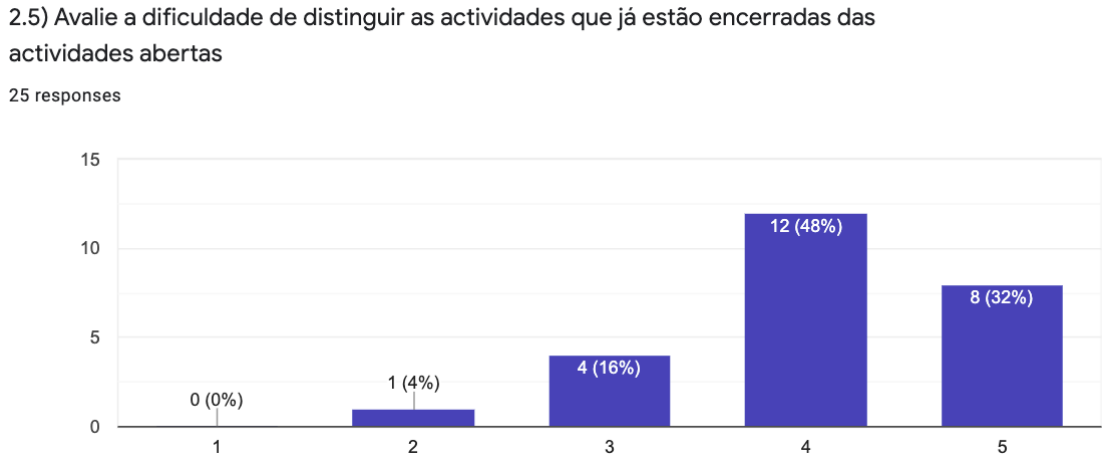


Figura 5.35: Respostas obtidas para a avaliação da distinção entre actividades abertas e encerradas



esta avaliação iniciava-se em "Pouco intuitivo" (1) escalando a "Muito intuitivo" (5). Na Figura 5.36 são apresentados os resultados obtidos, os quais foram favoráveis, indicando que a interface da aplicação é intuitivo.

Foi também avaliado se a experiência de utilização da aplicação, enquanto realizava o cenário, foi positiva. A escala das respostas a esta avaliação iniciava-se em "Não gostei" (1) escalando a "Gostei muito" (5). Na Figura 5.37 são apresentados os resultados obtidos, os quais mostram que os participantes tiveram uma experiência de utilização favorável.

A última avaliação realizada nesta secção teve como objectivo de compreender se a existe utilidade para esta aplicação, num contexto real. A escala das respostas a esta avaliação iniciava-se em "Pouco útil" (1) escalando a "Muito útil" (5). Na Figura 5.38 são

Figura 5.36: Respostas obtidas para a avaliação da interface da aplicação

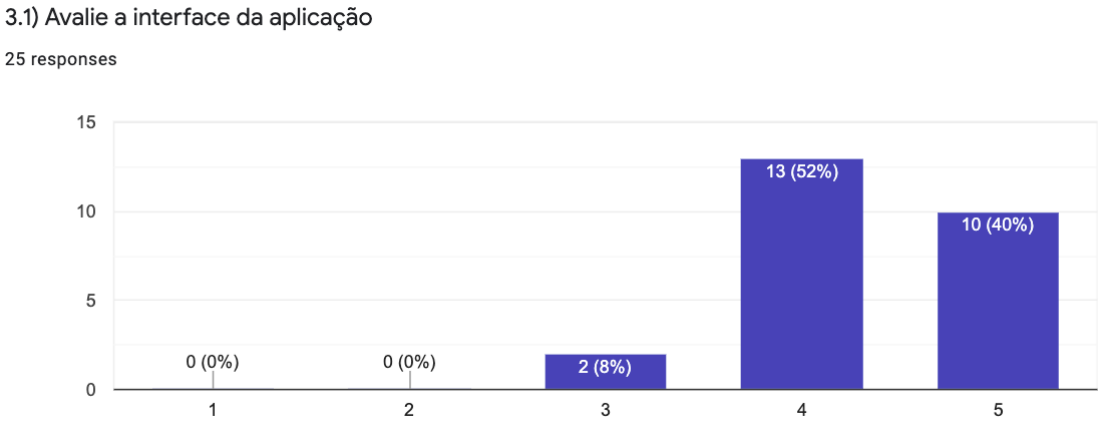
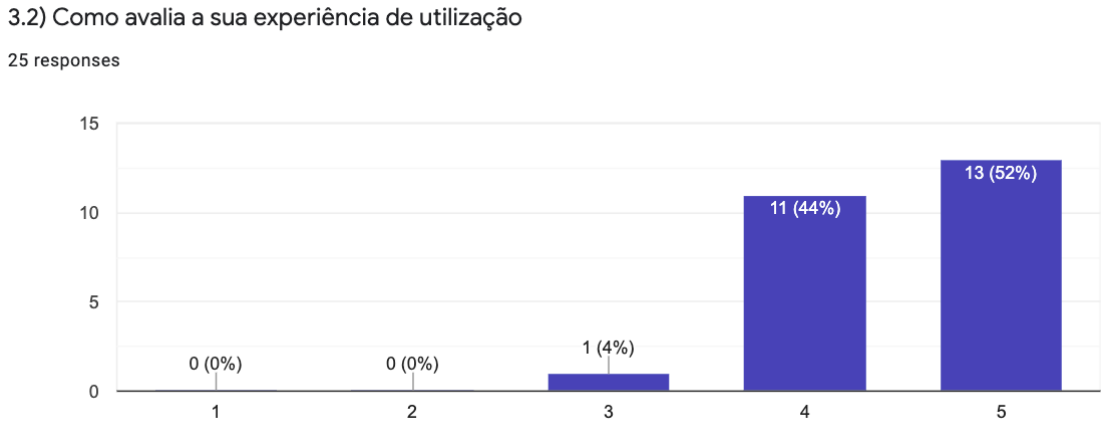


Figura 5.37: Respostas obtidas para a avaliação da experiência de utilização



apresentados os resultados obtidos, os quais indicam que os participantes reconhecem aplicação prática desta aplicação no contexto do ensino.

5.2.3.4 Questões relativas à usabilidade da aplicação

Após realização do cálculo da usabilidade do sistema para cada participante utilizando os resultados da última secção do questionário, procedemos a uma análise destes dados. Na Figura 5.39 são apresentados os resultados do cálculo do valor de usabilidade extraído do questionário de cada participante. Com estes dados foi obtida uma média para os resultados de 91.2 pontos e uma mediana de 95 pontos.

Utilizando a tabela de referência (ver Tabela 5.1), os resultados obtidos foram favoráveis, encontrando-se maioritáriamente na melhor classificação com a excepção do

Figura 5.38: Respostas obtidas para a avaliação da utilidade da aplicação

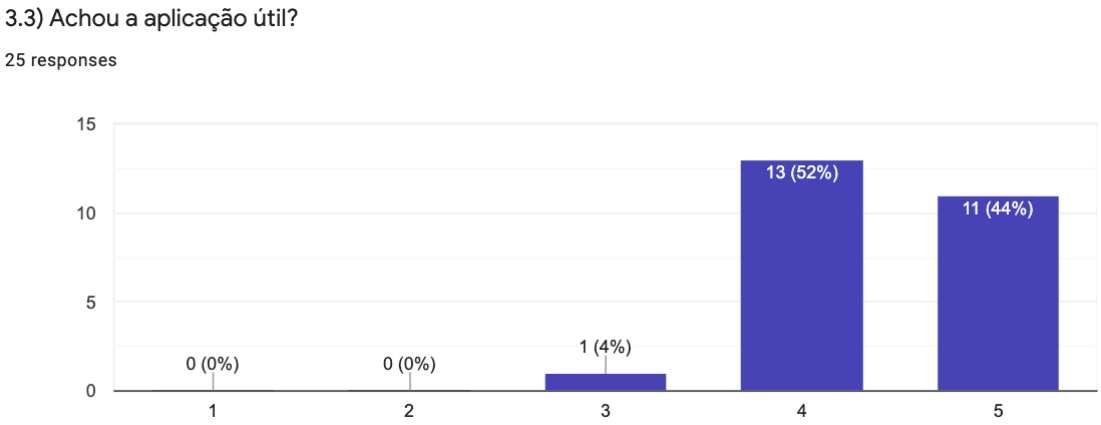
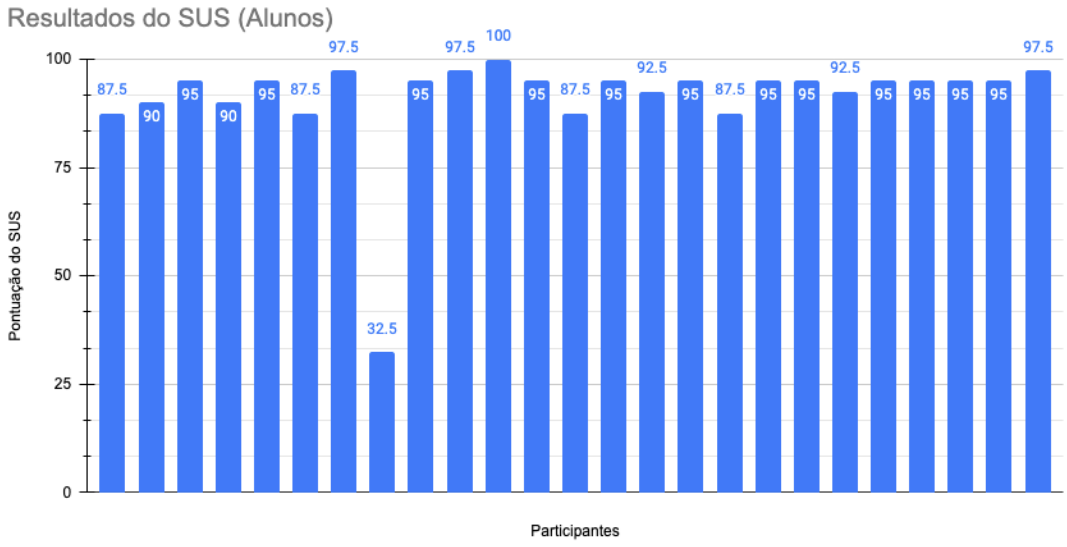


Figura 5.39: Resultados do cálculo do valor de usabilidade de cada participante para o cenário do aluno

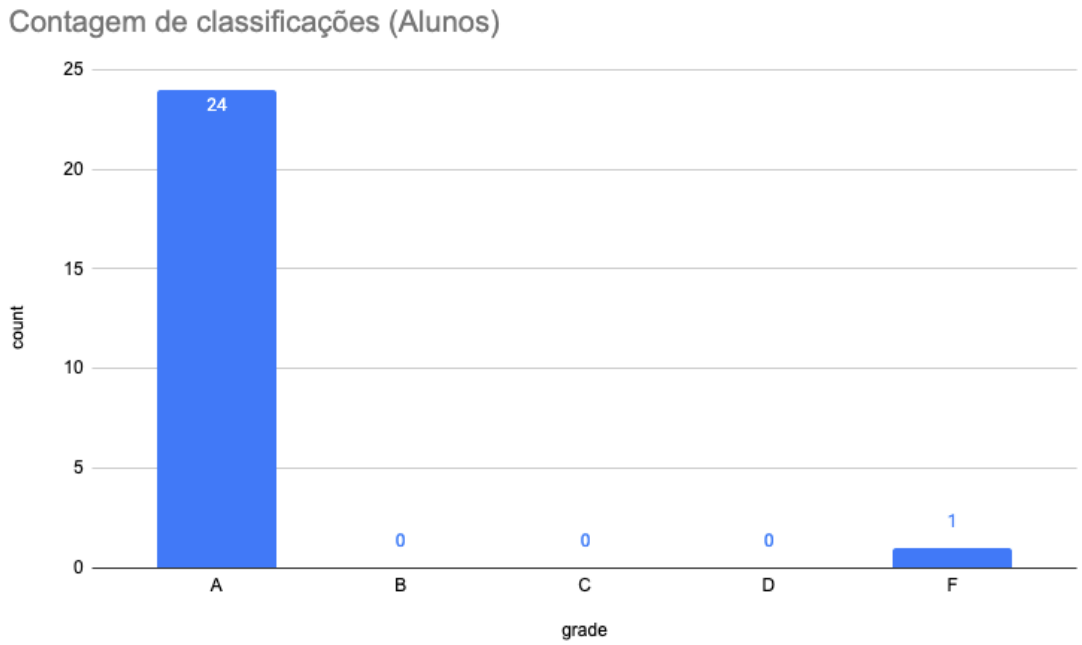


resultado de um participante que avaliou a aplicação com classificação *F*. A Figura 5.40 apresenta a contagem de resultados de participantes por classificação.

5.3 Síntese

Para avaliar a utilização da aplicação, foram concebidos alguns cenários para cada componente da aplicação. Estes cenários consistiram em tarefas de utilização, as quais são as actividades centrais da aplicação. Acompanhando os cenários, foram construidos questionários sobre a utilização da aplicação. Estes questionários consistiam em três grupos de

Figura 5.40: Contagem de resultados de participantes por classificação obtida para o cenário do aluno



questões: Questões sobre o cenário efectuado, questões sobre a utilização da aplicação e questões da escala de usabilidade do sistema.

De um ponto de vista geral, a aplicação foi considerada intuitiva e fácil de utilizar por ambos os tipos de utilizadores (professores e alunos). No que diz respeito ao método de construção de actividades, este deve ser melhorado, especificamente o processo de selecção de blocos. Em relação à edição de soluções, os participantes do tipo professor tiveram uma opinião favorável, indicando que o processo implementado para construção/edição de soluções é intuitivo. Os participantes do tipo aluno, quando questionados sobre a edição de soluções, não sentiram muita dificuldade a adaptarem-se ao ambiente implementado. O *feedback* das submissões do aluno dado aos professores teve boas recepções dos utilizadores com acesso aos dados de forma intuitiva e com os dados apresentados num formato fácil de interpretar. No que diz respeito à gestão de turmas (grupos de alunos) os dados obtidos durante a avaliação mostram que os processos de criação e atribuição de actividades de turmas são intuitivos e fáceis de utilizar e que os utilizadores de teste reconhecem utilidade no contexto do ensino para a aplicação desenvolvida.

6

Conclusões

Após o desenvolvimento, implementação e avaliação da aplicação *Bricks*, foi possível alcançar um conjunto de conclusões que serão apresentadas neste capítulo. Serão também apresentadas neste capítulo algumas melhorias e possíveis funcionalidades futuras que foram discutidas durante o desenvolvimento da aplicação.

6.1 Conclusões

Nesta dissertação foi proposta uma aplicação, denominada *Bricks*, para o desenvolvimento e execução de actividades de programação por blocos.

Esta aplicação está desenhada para funcionar num contexto *web*, suportada por um servidor leve e escalável, permitindo múltiplos acessos concorrentes.

Esta aplicação disponibiliza ferramentas para criação e execução de actividades de programação por blocos. Para o contexto da aprendizagem de programação, esta aplicação é uma forma divertida de aprender competências de programação. A utilização de uma linguagem de programação por blocos nas actividades de programação permite também, ao utilizador que tente resolver a actividade, um foco maior sobre a lógica do programa que constrói, sem haver uma preocupação com erros de sintaxe comuns em linguagens textuais.

Esta aplicação fornece também uma forma de criar actividades de programação com objectivos definidos por professores utilizadores da aplicação permitindo assim que esta aplicação possa adaptar a diferentes contextos do ensino.

A execução de actividades, na aplicação, permite aos utilizadores do tipo aluno obterem *feedback* imediato sobre a validação das suas soluções utilizando os testes da actividade, podendo comparar o resultado das suas soluções com o resultado esperado do teste. A integração de testes nas actividades tornou o conceito de actividades de programação, na aplicação *Bricks*, bastante versátil, destacando-se de outras aplicações de programação por blocos, que não possuem esta característica de avaliação de programas.

A aplicação permite a gestão de alunos, agrupando-os em turmas. A estas turmas, o

professor pode atribuir actividades e obter *feedback* das resoluções das actividades atribuídas. Este *feedback* pode ser exibido de formas diferentes, desde uma visão geral sobre a turma a detalhes de uma submissão específica a uma actividade.

Durante a fase de testes, os questionários que acompanhavam cada cenário visaram cobrir as principais funcionalidades de forma a detectar potenciais melhorias na implementação. A última secção de cada questionário continha um conjunto de questões cujo objectivo era pontuar a usabilidade do sistema. Apesar de ter sido possível detectar algumas melhorias como, por exemplo, a simplificação do processo de selecção de blocos durante a criação de actividades, os resultados dos testes de usabilidade indicaram que, em todos os cenários de utilização, a aplicação *Bricks* é intuitiva, fácil de utilizar, consistente e não possui uma curva de aprendizagem acentuada.

A aplicação *Bricks* permite portanto:

- Dois tipos de utilizadores: Professor e Aluno.
- Criação de actividades de programação por blocos.
- Atribuir actividades de programação a grupos de alunos para serem resolvidas.
- Execução de actividades de programação criadas na aplicação.
- Consultar *feedback* de resoluções de actividades.
- Consultar actividades de programação já resolvidas.

De acordo com estas funcionalidades enunciadas, os objectivos propostos nesta disseração foram alcançados.

6.2 Trabalho futuro

Nesta secção são contempladas algumas ideias sobre possíveis desenvolvimentos sobre a aplicação no âmbito do tema desta dissertação.

Existem algumas melhorias no design do interface de forma a não ser apenas funcional mas também apelativo. Para além da actualização do design, nesta categoria também pode ser enquadrada a alteração à selecção de blocos durante a criação de actividades.

Podem ser também realizadas optimizações que podem oferecer melhor desempenho de funcionamento da aplicação, nomeadamente, a utilização de uma *framework* mais dinâmica para gerar os interfaces dos clientes. A aplicação, no seu estado actual, foi maioritariamente desenvolvida com recurso muito reduzido à *framework Bootstrap*. A utilização de outras *frameworks* como *React*, *Vue.js* ou *Angular* seriam uma potencial melhoria de desempenho, assim como uma garantia de um interface consideravelmente mais reactivo do que o implementado.

O interface do editor de soluções de actividades pode também ser alvo de um melhor aproveitamento do espaço disponível. A distribuição actual visa proporcionar o maior

espaço possível para o utilizador compor a sua solução durante a execução de actividades, porém, pode haver um melhor equilíbrio na distribuição dos elementos presentes no interface de execução de actividades.

Para além disso, podem ser adicionados novos tipos de actividades para complementar o criador de actividades e fornecer à aplicação uma maior variedade de exercícios de programação.

Outra funcionalidade interessante que poderia ser implementada seria poder agrupar conjuntos de actividades em fichas de trabalho. Este conceito de fichas de trabalho pode ser útil por questões de organização de actividades.

Bibliografia

[1] *About Moodle - MoodleDocs*. URL: https://docs.moodle.org/310/en/About_Moodle (ver p. 15).

[2] *About Snap!* URL: https://snap.berkeley.edu/about (ver p. 11).

[3] *About SublimeLinter — SublimeLinter 3.4.24 documentation*. URL: http://www.sublimelinter.com/en/v3.10.10/about.html (ver p. 17).

[4] A. Begel e M. Resnick. "LogoBlocks: A Graphical Programming Language for Interacting with the World". Em: 2000 (ver p. 6).

[5] *Blockly | Google Developers*. URL: https://developers.google.com/blockly/ (ver p. 20).

[6] J. Brooke. "SUS: A quick and dirty usability scale". Em: *Usability Eval. Ind.* 189 (nov. de 1995) (ver p. 58).

[7] *CodeRunner*. URL: https://coderunner.org.nz/ (ver p. 16).

[8] *CodeRunner question type - MoodleDocs*. URL: https://docs.moodle.org/39/en/CodeRunner_question_type (ver p. 15).

[9] P. Feijóo-García e F. De la Rosa. "RoBlock – Web App for Programming Learning". Em: *International Journal of Emerging Technologies in Learning (iJET)* 11 (dez. de 2016), p. 45. DOI: 10.3991/ijet.v11i12.6004 (ver p. 2).

[10] *GitHub - trun/flashblocks*. URL: https://github.com/trun/flashblocks (ver p. 7).

[11] *Introduction to Blockly | Google Developers*. URL: https://developers.google.com/blockly/guides/overview (ver p. 20).

[12] B. Jost et al. "Graphical Programming Environments for Educational Robots: Open Roberta - Yet Another One?" Em: *2014 IEEE International Symposium on Multimedia*. Taichung, Taiwan: IEEE, 2014, pp. 381–386. DOI: http://doi.org/10.1109/ISM.2014.24 (ver pp. 2, 5).

[13] J. Leal e F. Silva. "Mooshak: A Web-based multi-site programming contest system". Em: *Software Practice Experience* 33 (mai. de 2003), pp. 567–581. DOI: 10.1002/spe.522 (ver p. 12).

[14] J. R. Lewis. "The System Usability Scale: Past, Present, and Future". Em: *International Journal of Human–Computer Interaction* 34.7 (2018), pp. 577–590. DOI: 10.1080/10447318.2018.1455307. eprint: https://doi.org/10.1080/10447318.2018.1455307. URL: https://doi.org/10.1080/10447318.2018.1455307 (ver p. 58).

[15] *Manual de Submissão Mooshak*. URL: https://moodle.fct.unl.pt/pluginfile.php/130729/mod_resource/content/0/labs/ManualSubmissaoMooshak.pdf (ver p. 13).

[16] M. Mataric. *In The Robotics Primer*. Cambridge, Massachusetts: The MIT Press, 2007 (ver p. 2).

[17] *Moodle plugins directory: Virtual programming lab*. URL: https://moodle.org/plugins/mod_vpl (ver p. 16).

[18] *moodle-qtype$_c$oderunner/authorguide.mdatmastertrampgeek/moodle–qtype$_c$oderunnerGitHub*. URL: https://github.com/trampgeek/moodle-qtype_coderunner/blob/master/authorguide.md (ver p. 15).

[19] *Mooshak*. URL: https://mooshak.dcc.fc.up.pt/~mooshak-test/ (ver p. 14).

[20] *Mooshak's help*. URL: https://mooshak.dcc.fc.up.pt/~mooshak-test/cgi-bin/execute/1607133560631020 8?help+./procedures/before/contest (ver p. 13).

[21] *Mooshak's help*. URL: https://mooshak.dcc.fc.up.pt/~mooshak-test/cgi-bin/execute/1607133560631020 8?help+./procedures/before/problems (ver p. 13).

[22] ". Resnick et al. ""Scratch: programming for all"". Em: *Communications of the ACM* 52.11 (2009). ISSN: 0001-0782. DOI: https://doi.org/10.1145/1592761 (ver pp. 2, 11).

[23] *Scratch - About*. URL: https://scratch.mit.edu/about (ver p. 9).

[24] *Scratch - Developers*. URL: https://scratch.mit.edu/developers (ver p. 21).

[25] *Scratch Blocks is a library for building creative computing interfaces. - LLK/scratch-blocks*. URL: https://github.com/LLK/scratch-blocks (ver p. 21).

[26] *Scratch Statistics - Imagine, Program, Share*. URL: https://support.code.org/hc/en-us/articles/204784827-What-is-Code-org- (ver p. 8).

[27] *Virtual Machine used to represent, run, and maintain the state of programs for Scratch 3.0 - LLK/scratch-vm*. URL: https://github.com/LLK/scratch-vm (ver p. 21).

[28] *VPL - Virtual Programming Lab - Support*. URL: https://vpl.dis.ulpgc.es/index.php/support (ver p. 18).

[29] *VPL - Virtual Programming Lab - What is VPL?* URL: https://vpl.dis.ulpgc.es/index.php/about/what-is-vpl (ver p. 16).

[30] D. Weintrop e U. Wilensky. "Comparing Block-Based and Text-Based Programming in High School Computer Science Classrooms". Em: *ACM Trans. Comput. Educ.* 18.1 (out. de 2017). DOI: 10.1145/3089799. URL: https://doi.org/10.1145/3089799 (ver p. 2).

[31] *What is Code.org?* URL: https://scratch.mit.edu/statistics/ (ver p. 11).

[32] J. M. Wing. "Computational thinking". Em: *Communications of the ACM* 49.3 (2012). ISSN: 0001-0782. DOI: https://doi.org/10.1145/1118178.1118215 (ver p. 1).

[33] ". M. Wing". ""Computational thinking benefits society"". Em: *"40th Anniversary Blog of Social Issues in Computing"* ("2014/1/10") (ver p. 1).

I

Anexo A

I.1 Questões presentes no Questionário relativo ao cenário do aluno

1. 1.1 Qual é a sua idade?
   1.2 Qual a sua experiência com programação?
2. 2.1 Avalie o processo de alteração da palavra-chave
   2.2 Avalie o processo de aceder a uma actividade
   2.3 Avalie a apresentação dos objectivos, enunciado e recursos da actividade
   2.4 Avalie o processo de construção/edição da solução (programa) de uma dada actividade
   2.5 Avalie a dificuldade de distinguir as actividades que já estão encerradas das actividades abertas
3. 3.1 Avalie a interface da aplicação
   3.2 Como avalia a sua experiência de utilização
   3.3 Achou a aplicação útil?
4. 4.1 Achei que gostaria de utilizar esta aplicação frequentemente
   4.2 Achei que esta aplicação é demasiado complexa
   4.3 Achei que esta aplicação é fácil de utilizar
   4.4 Achei que iria precisar de ajuda de um técnico para utilizar esta aplicação
   4.5 Achei que as funcionalidades desta aplicação estão bem integradas
   4.6 Achei que a aplicação tem muitas inconsistências
   4.7 Achei que a maior parte das pessoas aprenderiam a utilizar esta aplicação rapidamente
   4.8 Achei que a utilização desta aplicação seria muito complicada

4.9 Achei que estive confiante ao utilizar esta aplicação

4.10 Achei que precisei de aprender muito antes de conseguir utilizar a aplicação

I.2 Questões presentes no 1. Questionário relativo aos cenários do professor (gestão de turmas/actividades)

1. 1.1 Qual é a sua idade?

   1.2 Tem alguma experiência com ambientes de gestão/atribuição de actividades a alunos?

2. 2.1 Avalie o processo criação de turmas

   2.2 Avalie o processo de criação de actividades a partir de um ficheiro *.zip

   2.3 Avalie o processo de associar actividades a turmas

   2.4 Avalie o processo de visualizar número de alunos que completaram uma actividade

   2.5 Avalie o que achou do relatório geral de actividades da turma

   2.6 Avalie a dificuldade de compreender a informação disponibilizada pelo relatório geral de actividades

3. 3.1 Avalie a interface da aplicação

   3.2 Avalie a sua experiência de utilização

   3.3 Achou a aplicação útil para gestão/atribuição de actividades a turmas?

4. 4.1 Achei que gostaria de utilizar esta aplicação frequentemente

   4.2 Achei que esta aplicação é demasiado complexa

   4.3 Achei que esta aplicação é fácil de utilizar

   4.4 Achei que iria precisar de ajuda de um técnico para utilizar esta aplicação

   4.5 Achei que as funcionalidades desta aplicação estão bem integradas

   4.6 Achei que a aplicação tem muitas inconsistências

   4.7 Achei que a maior parte das pessoas aprenderiam a utilizar esta aplicação rapidamente

   4.8 Achei que a utilização desta aplicação seria muito complicada

   4.9 Achei que estive confiante ao utilizar esta aplicação

   4.10 Achei que precisei de aprender muito antes de conseguir utilizar a aplicação

I.3 Questões presentes no 1. Questionário relativo aos cenários do professor (criação de actividades)

1. 1.1 Qual é a sua idade?

   1.2 Já leccionou alguma cadeira de programação

2. 2.1 Relativamente à criação da actividade, gostou da forma de apresentação/introdução de: [Titulo], [Enunciado] e [Recursos]

   2.2 Avalie o processo de introdução de testes durante a criação de actividades

   2.3 Avalie o processo de seleção de blocos na criação de actividades

   2.4 Acha útil ser possivel limitar a utilização blocos?

   2.5 Acha útil ser possível fazer "download"da actividade (transferir actividade)?

   2.6 Avalie o processo de construção/edição da solução (programa) de uma dada actividade

3. 3.1 Achou intuitivo o interface da aplicação no que diz respeito à criação de actividades

   3.2 Achou intuitivo o interface da aplicação no que diz respeito à edição de soluções de actividades

   3.3 Avalie a sua experiência de utilização

   3.4 Achou a aplicação útil no contexto de ensino?

   3.5 Utilizaria a aplicação nas suas aulas?

4. 4.1 Achei que gostaria de utilizar esta aplicação frequentemente

   4.2 Achei que esta aplicação é demasiado complexa

   4.3 Achei que esta aplicação é fácil de utilizar

   4.4 Achei que iria precisar de ajuda de um técnico para utilizar esta aplicação

   4.5 Achei que as funcionalidades desta aplicação estão bem integradas

   4.6 Achei que a aplicação tem muitas inconsistências

   4.7 Achei que a maior parte das pessoas aprenderiam a utilizar esta aplicação rapidamente

   4.8 Achei que a utilização desta aplicação seria muito complicada

   4.9 Achei que estive confiante ao utilizar esta aplicação

   4.10 Achei que precisei de aprender muito antes de conseguir utilizar a aplicação