isec
Engenharia

MESTRADO EM INFORMÁTICA E SISTEMAS

DEFINITIVO

Estudo e aplicação dos princípios e práticas *DevOps* no processo de ensino-aprendizagem de programação

Autor

André Filipe Santos Vicente

Orientador

João Carlos Costa Faria da Cunha

INSTITUTO POLITÉCNICO
DE COIMBRA

INSTITUTO SUPERIOR
DE ENGENHARIA
DE COIMBRA

Coimbra, julho 2022

isec
Engenharia

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA E SISTEMAS

Estudo e aplicação dos princípios e práticas *DevOps* no processo de ensino-aprendizagem de programação

Relatório de Trabalho de Projeto para a obtenção do grau de Mestre em Informática e Sistemas

Especialização em Tecnologias da Informação e do Conhecimento.

Autor

André Filipe Santos Vicente

Orientador

João Carlos Costa Faria da Cunha

INSTITUTO POLITÉCNICO
DE COIMBRA

INSTITUTO SUPERIOR
DE ENGENHARIA
DE COIMBRA

Coimbra, julho 2022

Agradecimentos

Em primeiro, gostaria de agradecer ao professor Doutor João Cunha pelo seu apoio, disponibilidade e orientação.

Agradeço ao professor Doutor Francisco Pereira por toda a ajuda, disponibilidade e cooperação ao longo deste projeto. Agradeço também à professor Doutor Anabela Gomes pela ajuda e disponibilidade durante o projeto.

Agradeço a todos os professores e alunos da licenciatura em Engenharia Informatica e Mestrado em Informática e Sistemas, que de maneira direta ou indireta participaram e/ou se disponibilizaram a participar neste projeto.

Agradeço à minha família e a todos os colegas e amigos que encontrei ao longo deste percurso.

E agradeço aos meus colegas e amigos do trabalho que de forma direta ou indireta ajudaram na conciliação deste projeto de mestrado com o trabalho diário na empresa.

A todos o meu muito obrigado.

Abstract

Teaching and Learning programming languages in high-education, especially for the first-year students is a major challenge due to two main factors: the ratio of students/teachers in these courses are usually very high, not allowing a greater proximity between teachers and students; these students have distinct backgrounds and logical reasoning, and therefore make progress at very different paces.

With the ever-increasing demand from the industry for professionals with programming skills, the high-education institutions continue to increase their offers in these areas, pressuring the teachers for more positive results with even more students.

This demand for more efficiency and effectiveness is common in every sector of the industry, including software. DevOps is a software development methodology largely adopted by the industry, that promotes the continuous improvement of the software development process. This methodology simplifies the problems across all development stages by removing grey areas between the team and promoting cooperation supported by the automation of tasks through tools and allowing developers to focus on what matters.

This project explores the possibility to improve the teaching-learning programming languages process, by recurring to the principles of DevOps to create an improved method able to individualize learning and reduce teacher effort.

A Moodle-based platform was developed with several mechanisms that automate tasks such as providing learning materials and exercises to each student, controlling the performance of the work, and providing feedback to students and teachers. This allows teachers to receive quick feedback and to focus on what really matters, like the difficulties of the students or the preparation of new learning resources, and let the students practice and study at their own pace.

A preliminary experience took place with a teacher and students of an introductory course in Programming. Although preliminary, the results obtained are promising and with positive feedback from the teacher and the students.

Keywords: Teaching-learning programming languages, DevOps, Moodle, Teaching efficiency

Resumo

O ensino-aprendizagem de linguagens de programação no ensino superior, especialmente para os estudantes do primeiro ano, é um grande desafio devido, essencialmente, a dois fatores: a proporção alunos/professores nestes cursos é normalmente elevada, não permitindo uma maior proximidade entre professores e estudantes, e os diferentes *backgrounds* e raciocínio lógico que os estudantes possuem, levando-os a progredir a ritmos muito diferentes.

Com a crescente procura da indústria por profissionais com conhecimentos de programação, as instituições de ensino superior continuam a aumentar a sua oferta nestas áreas, pressionando os professores a obter resultados cada vez mais positivos com turmas com cada vez mais estudantes.

Esta exigência de mais eficiência é comum a todos os sectores da indústria. O *DevOps* é uma metodologia de desenvolvimento de software largamente adotada por esta indústria, que promove a melhoria contínua do processo de desenvolvimento de software. Esta metodologia simplifica os problemas nas fases de desenvolvimento, removendo áreas cinzentas entre a equipa e promovendo a cooperação apoiada pela automatização de tarefas através de ferramentas.

Este projeto pretende explorar a possibilidade de melhorar o processo do ensino-aprendizagem de linguagens de programação, recorrendo aos princípios do *DevOps* para criar um método melhorado que visa individualizar a aprendizagem e reduzir o esforço do professor.

Foi desenvolvida uma plataforma baseada no *Moodle* com vários mecanismos para automatizar tarefas como o fornecimento de materiais e exercícios de aprendizagem, permitindo que os professores disponham de *feedback* rápido sobre o desempenho dos alunos nas atividades. Assim, podem concentrar-se nas dificuldades dos alunos e na preparação de novos recursos de aprendizagem.

Uma experiência preliminar teve lugar com um professor e estudantes de uma unidade curricular introdutória de Programação. Embora preliminares, os resultados obtidos com a plataforma e método desenvolvidos são promissores tendo obtido *feedback* positivo por parte do professor e dos estudantes.

Palavras-Chave: Ensino e aprendizagem de programação; *DevOps*; *Moodle*; Métodos de ensino; Eficiência no ensino de programação.

Índice

Lista de Figuras

Lista de Tabelas

Definições e Acrónimos

B2B – *Business to Business* – Empresas que vendem produtos ou prestam serviços a outras empresas, pessoas jurídicas.

B2C – *Business to Consumer* – Empresas que vendem produtos ou prestam serviços ao consumidor final.

IaC – *Infrastructure as Code* – Utilizar programação e código para criar e gerir infraestrutura e serviços

IP – Unidade curricular de Introdução à Programação

ISEC – Instituto Superior de Engenharia de Coimbra

LEI – Licenciatura em Engenharia Informatica

LMS – *Learning Managment System* – Sistema/Plataforma de ensino online que disponibiliza, gere e permite a criação de diversos recursos direcionados à aprendizagem e é projetado para a modalidade de ensino a distância.

P – Unidade curricular de Programação

POO – Unidade curricular de Programação Orientada a Objetos

SIGQ – Sistema Interno de Garantia da Qualidade do Politécnico de Coimbra

SLA – *Service Level Agreement*– Acordo de Nível de Serviço, é um documento realizado entre cliente e fornecedor que visa garantir a qualidade dos serviços prestados perante um determinado parâmetro (exemplo: disponibilidade de serviço igual a 99%, tempo de resposta máximo a incidentes de 24 horas)

1 Introdução

Ensinar e aprender programação constitui um dos maiores desafios na ciência da computação e é muitas vezes um "processo" complexo, quer para os alunos que têm dificuldades em aprender, quer para os professores nas estratégias que aplicam no ensino, especialmente em aulas introdutórias de programação [1] [2]. Têm surgido diversas pesquisas relacionadas com o propósito de identificar as estratégias, métodos ou ferramentas mais corretas e de maior eficácia que possam ser utilizadas para ajudar os alunos a aprender mais facilmente e os professores a ensinar com mais eficiência. Existem também diversos estudos no sentido de medir as taxas de retenção e desistência destes cursos em especial das unidades curriculares introdutórias de programação [2] [3]. Assim, torna-se evidente que os alunos que começam a programar nem sempre possuem as habilidades necessárias para interpretar e criar estratégias de resolução para os exercícios, ficando facilmente frustrados e desmotivados [4]. Com a elevada procura de profissionais de informática [5] [6] [7] [8] por parte da industria, também os cursos superiores desta área abrem mais vagas para alunos. No entanto, um pouco por todo o mundo, existem elevadas taxas de insucesso e de abandono em cursos de ciências da computação e ligados à programação [2] [3]. São exemplos destas afirmações a Universidade de Coimbra [9] no curso de Engenharia Informática e o Instituto Superior de Engenharia de Coimbra (ISEC) também na licenciatura de Engenharia Informática (LEI) [10]. Por exemplo o ISEC, possui taxas de sucesso a disciplinas como Introdução à Programação, de apenas 58%; Programação, com 53%; Programação Orientada a Objetos, com 59%, para o curso de Engenharia Informática na vertente diurna. Na vertente pós-laboral estes números mantêm-se baixos com taxas de sucesso de 47% à unidade curricular de Introdução à Programação, 45% a Programação e Programação Orientada a Objetos com 67%.

Existem ainda outras abordagens para ensinar e despertar interesse na programação, como *gamification* ou *bootcamps*. Na *gamification* é alcançada motivação através de concursos ou jogos com recompensas que ajudam a perceber, a gerar interesse e empatia pelos conceitos da programação. Os *bootcamps* realizam uma abordagem diferente, sendo normalmente usados por pessoas que pretendem mudar de área ou adquirir conhecimento conciso sobre uma linguagem de programação. São cursos intensivos, com processos e ferramentas bem definidos, que exigem o máximo empenho dos seus alunos, mas o seu sucesso nem sempre é efetivo, os conhecimentos adquiridos necessitam de ser cimentados e enquadrados dentro das *stacks* tecnológicas. [11]

Na indústria existem também diversas abordagens que pretendem otimizar o processo de desenvolvimento e aumentar a sua eficácia. O conceito *DevOps* tem-se afirmado e crescido cada

vez mais nas metodologias mais adotadas pela indústria de software [12]. Representa uma convergência entre diversas filosofias de gestão e é caracterizado por uma melhoria contínua de processos e interligação de todas as fases de um projeto.

Melhorar o tempo despendido em cada fase de desenvolvimento; melhorar a qualidade do produto ou serviço; melhorar a satisfação do cliente e gerar menos desperdício são os objetivos principais desta metodologia.

É pretendido aplicar a metodologia, conceitos e práticas *DevOps* à melhoria das práticas de ensino-aprendizagem de linguagens de programação, criando uma plataforma assente em fluxogramas, ferramentas e processos que possam ser melhorados iterativamente. Permitindo aos alunos obter conhecimentos de programação de maneira mais acessível, concisa e adaptada às suas necessidades individuais. Assim como, auxiliar os métodos de ensino-aprendizagem, ajudando os docentes em tarefas repetitivas e/ou outras, podendo estes concentrarem-se em aspetos pedagógicos e no apoio aos alunos.

1.1 Motivação para o tema do projeto

A metodologia *DevOps* surgiu no meu percurso profissional associada à melhoria contínua conseguida pelo uso de automatismos e ferramentas para realizar tarefas repetitivas.

Sendo uma abordagem em expansão, foi conquistando a sua afirmação na indústria. Rapidamente passou a existir uma tendência em pensar que o *DevOps* consiste maioritariamente em ferramentas e/ou programas para melhorar e agilizar toda e qualquer tarefa. No entanto, esta metodologia é focada nas pessoas - em especial no feedback - e colaboração continua para descobrir os pontos, processos e fluxos que poderiam funcionar melhor ou de maneira mais rápida e menos repetitiva.

A aprendizagem de programação foi também um processo que marcou o meu percurso académico e que é essencial no percurso de qualquer informático. Sabendo que aprender a programar demora o seu tempo e depende maioritariamente da pessoa que está a aprender, a eficácia do processo está também dependente dos processos que são usados.

Apesar dos professores fazerem todos os possíveis pelo sucesso dos alunos, fatores críticos como resultados de aprovação abaixo do esperado e um número crescente de alunos em turmas de programação, demonstra que este processo pode beneficiar de melhorias com base no feedback entre as partes envolvidas e na otimização de tarefas demorosas e/ou repetitivas.

Assim, motivado também pelo meu orientador, decidi tentar adaptar esta abordagem da indústria - com a qual trabalho diariamente - ao processo de ensino-aprendizagem de programação no intuito de tentar auxiliar os professor e alunos da minha instituição.

1.2 Objetivo

Conforme referenciado anteriormente, o objetivo deste projeto é alcançar um processo e plataforma que apoie o ensino-aprendizagem de programação, utilizando os princípios, ferramentas e filosofias da metodologia *DevOps* - uma das metodologias mais usada na indústria, para resolver e ajudar em diversas questões deste processo. É pretendido assim automatizar, tornar mais acessível e facilitar tarefas relevantes para professores e alunos, melhorando os métodos e metodologias usados através do alívio de tarefas repetitivas, ajudar na resolução de exercícios, revisões teóricas e na recolha de resultados das tarefas realizadas, através da plataforma.

A Tabela 1, pretende clarificar as metas a atingir até ao objetivo exposto anteriormente, assim como elaborar mais um pouco sobre cada um delas.

Tabela 1 - Tabela de objetivos e metas do projeto

Objetivos e metas	
OB1 -Plataforma e sistema de ajuda e suporte ao ensino-aprendizagem de programação	Definição de um sistema de suporte ao ensino/aprendizagem de programação, que permita melhorar a eficácia e eficiência formativa.
M1- Fundamentação teórica	Descrição da metodologia *DevOps*, os seus processos e as suas melhores práticas (ler artigos, livros)
	Estudo das metodologias já consideradas no ensino da programação (ler artigos, publicações)
M2- Pedagogia do ensino de programação e de fatores importantes	Recorrendo à fundamentação anterior e aos aspetos pedagógicos desta área criar fluxogramas, *root cause diagrams, fishbone diagrams*, gráficos que possam analisar e reunir os aspetos chave.
M3- Análise entre o processo de ensino de programação e o ciclo DevOps	Criar paralelismo entre ciclo *DevOps* e o ciclo de aprendizagem de programação, identificando pontos de automatização e melhoria.
M4- Análise do ensino de programação no ISEC	Análise das taxas de aprovação e medidas planeadas para cadeiras introdutórias de programação no ISEC. Juntar esta análise ao 'M2' para criar áreas especificas de atuação atendendo ao ambiente em causa (ISEC).
M5- Análise de ferramentas online que cumpre o pressuposto de ensino de programação	Análise de ferramentas não relacionadas com *DevOps*, que implementam aprendizagem de programação, de maneira a recolher funcionalidades, interação e abordagens utilizadas.
M6 -Análise de requisitos e elaboração de funcionalidades	Definição de requisitos, entregas e funcionalidade para a plataforma (criar versões)
M7 -*Testes e análise de resultados*	Testar a plataforma com voluntários/alunos, obtendo *feedback* e analisando os resultados obtidos.
	OB – Objetivo, M – Meta/Milestone

É possível ver ainda a respetiva hierarquia de tarefas através da Figura 1.

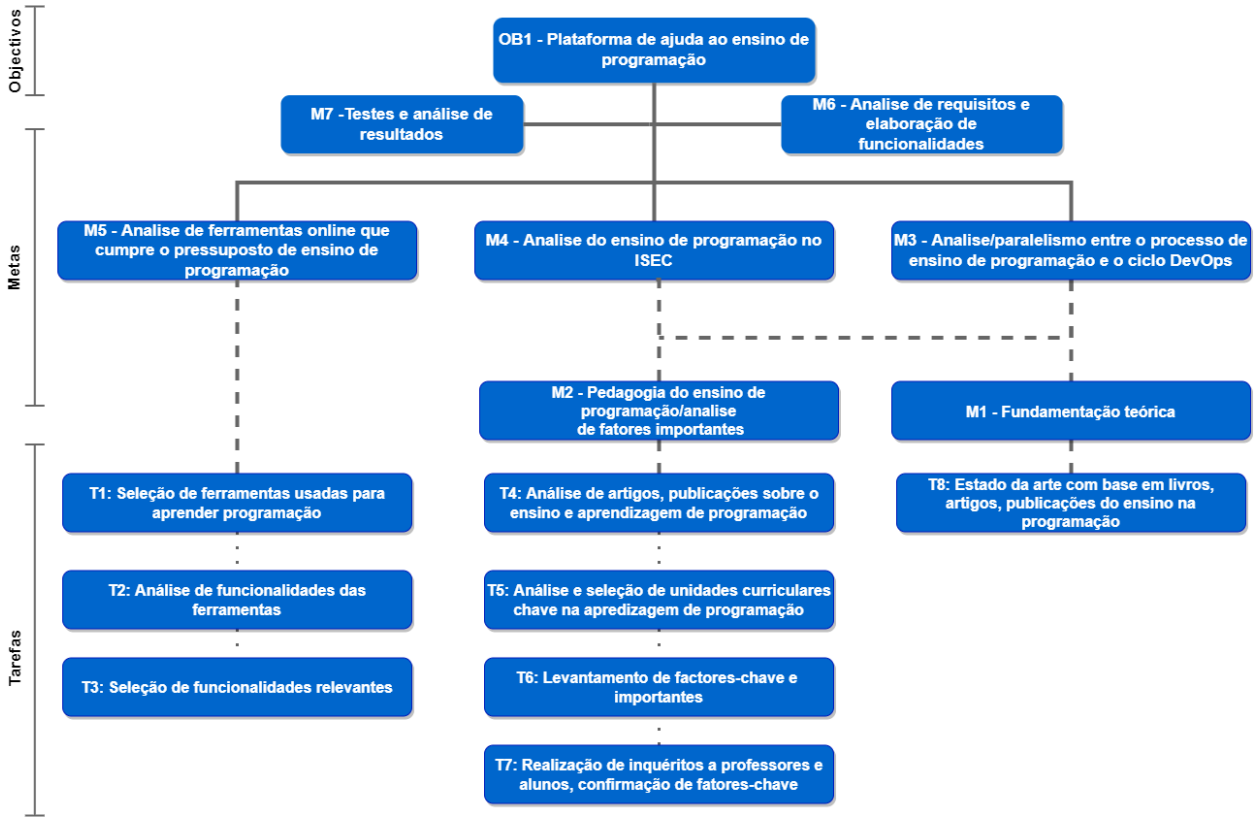


Figura 1 – Diagrama hierárquico de metas, objetivos e tarefas do projeto

1.3 Estrutura do relatório

Este documento está organizado da seguinte forma: no capítulo 2 é analisado o ensino de programação e os métodos que são genericamente aplicados nas aulas, a metodologia *DevOps* e as suas características. É depois traçada uma analogia entre o ensino-aprendizagem de programação e o *DevOps*, comparando os dois processos.

No capítulo 3 são analisadas algumas unidades curriculares do LEI-ISEC de programação e os seus resultados nos anos letivos 2018/2019 e 2019/2020, sendo depois levantados com base nesta análise os possíveis pontos chave na aprendizagem de programação por parte dos alunos.

No capítulo 4 é feita uma análise das plataformas online existentes que permitem a aprendizagem de programação por parte dos seus utilizadores e das ferramentas que podem ser utilizadas para constituir a plataforma que pretende ser criada. Após isto são também especificados os requisitos necessários para a mesma, sendo por fim feita a escolha da ferramenta que vai ser utilizada.

No capítulo 5 é traçado um plano de implementação de funcionalidades para a plataforma, sendo depois apresentadas as diversas fases de experimentação assim como seus resultados.

Por fim, no capítulo 6 são apresentadas as conclusões acerca do trabalho desenvolvido e sugestões, que poderão ser realizadas como trabalho futuro.

2 *DevOps* e o Ensino-aprendizagem da programação

A pedagogia é um tópico que estuda a aprendizagem e ensino de disciplinas e/ou tópicos escolares [13]. Diversos estudos foram desenvolvidos, obtendo avanços em diversas áreas como Português, Matemática ou Ciências sobre as formas e processos mais eficazes de ensino para replicar e transmitir o conhecimento específico de cada área [14]. Algumas áreas vão ainda sofrendo alterações nas suas características como é o caso da informática, que apesar de ser uma disciplina mais nova em comparação com outras, tem sofrido alterações na sua definição e identidade ao longo do tempo.

Assim, podemos dizer que aprender e ensinar são atividades especificas com desafios distintos que dependem de quem as realiza. Aprender a programar é uma atividade praticamente "artística" e cheia de caminhos alternativos para a resolução de cada problema. É um processo que consiste na aquisição e utilização de pensamento abstrato para formar estruturas lógicas que se traduzem depois em linhas de código de uma determinada linguagem. São estes os dois maiores desafios para programar: aprender e dominar a sintaxe de uma linguagem de programação e adquirir pensamento abstrato. No entanto, existem ainda outras variáveis. Programar requer escrever linhas de código, testar, fazer *debug* e *troubleshooting* do código, tarefas que envolvem estratégias distintas e dependentes do *background*, experiência e abstração de cada pessoa [1].

O uso de ferramentas como acelerador e impulsionador de mudança e aprendizagem é algo que pode ajudar a melhorar um processo. Anteriormente, foi realçado que a aprendizagem de programação poderá usar outros princípios e ferramentas para melhorar o seu processo, assim também podem ser traçados paralelismos e estratégias recorrendo à *toolchain DevOps*, para criar automatismos e ferramentas que suportem os processos do método a usar no ensino-aprendizagem de programação.

2.1 Ensino de programação

A informática/computação passou a existir como tema de aprendizagem em 1949/1950, com a junção de algoritmo teórico, lógica matemática e a invenção do armazenamento de programas em dispositivos eletrónicos surgindo a programação através da linguagem *Assembly* [16]. Desde aí que abrange cada vez mais áreas e continua a evoluir nos paradigmas e número de linguagens para programar.

Atualmente, o ensino de programação continua a evoluir ao ritmo da tecnologia e das necessidades da indústria.

2.1.1 Estudos, resultados e tendências dos últimos anos

Diversos autores e artigos citam elevadas taxas de retenção existentes em cursos superiores, um pouco por todo o mundo, ligados ao ensino da informática [17] [18].

Surgiram novas tendências, como o uso de aspetos de *gamification*[1] para melhorar a interação e interesse dos alunos pelos tópicos e exercícios, a automação recorrendo ao uso de ferramentas automáticas de avaliação de código para permitir a obtenção de *feedback* após submissão de exercícios ou o uso de mecanismos como *code review*[2], entre outros, para identificar erros e ajudar o programador a seguir as melhores práticas.

Entre as plataformas mais usadas com o intuito de apoiar o ensino-aprendizagem, estão as plataformas *Learning Managment System* (*LMS)*. Uma plataforma LMS é um software que ajuda a criar, gerir, organizar e fornecer materiais de estudo e treino de um determinado assunto para seu público.

Importa ainda realçar que, no que diz respeito a estudos ou artigos dos resultados de programação no ensino superior, estes tendem a ser limitados e difíceis de encontrar. Quer por não existirem muitas publicações sobre resultados negativos nas unidades curriculares de Programação do ensino superior, ou por muitos dos artigos deste tópico utilizarem e citarem os meus estudos e dados[2].

2.1.2 Abordagens no ensino de programação

O método de ensino de programação pode variar dependendo do tipo de aulas ou abordagem que se pretende realizar. Podem considerar-se dois tipos de abordagem que são genericamente *lecturer-oriented* e *student-oriented* [19].

- **Lecturer-oriented**

  Esta abordagem é direcionada ao conteúdo lecionado e à aprendizagem por memorização dos conteúdos expostos pelo professor. Normalmente é usada em aulas introdutórias, com grande número de alunos, ou com alunos com pouco conhecimento dos tópicos mostrados ou *background* de programação. Este tipo de abordagem permite apenas a aprendizagem básica sobre os tópicos apresentados.

  Para ser eficaz é necessário haver componentes adicionais de discussão entre os alunos, questões e exercícios para cimentar os conhecimentos expostos. Esta abordagem pode

[1] Gamification consiste na utilização de técnicas e estratégias associadas aos jogos
[2] *Code review* ou *peer review* é a prática de fazer revisões no código desenvolvido, identificando possíveis problemas ou bugs e oferecendo feedback.

não ser a mais eficaz para ser usada sozinha no sujeito da engenheira, mais especificamente, na área de aprendizagem de programação. Os alunos, especialmente aqueles que começam a programar, necessitam de uma interligação forte com a componente prática de maneira a conseguirem adquirir os conhecimentos teóricos em práticos e a não se sentirem frustrados na resolução de exercícios.

- **Student-oriented**

  Esta abordagem é direcionada aos alunos e às suas ações. A aprendizagem é conseguida através da interação entre alunos e o seu envolvimento e participação na resolução de exercícios e atividades de programação. O professor e o contexto de sala de aula funcionam como facilitadores enquanto os alunos resolvem exercícios, expõem dúvidas e interagem entre si. Desta forma será mais fácil ir ao encontro das dúvidas, dificuldades e espectativas desenvolvendo mais autonomia e interesse nos alunos.

  Esta abordagem apresenta ainda desafios como conseguir responder corretamente às necessidades e dificuldades de todos os alunos, assim como conseguir motivá-los a interagir e participar na aula. Pode não ser a abordagem mais indicada para alunos com pouco conhecimento ou iniciantes.

A abordagem mais adequada para a aprendizagem de programação seria a *student-oriented*, uma vez que cada aluno possui dificuldades distintas. No entanto, está abordagem pode ser incomportável em unidades curriculares ou aulas onde o número de alunos é elevado ou em que a participação e interesse dos alunos seja baixo.

2.1.3 Metodologias no ensino de programação

Apesar dos poucos estudos existentes da forma e processo correto de ensino de programação, têm sido identificadas algumas metodologias aplicadas no processo [1]. Embora nem todas as metodologias sejam indicadas para qualquer aluno, a escolha da metodologia mais adequada beneficia a aprendizagem [20]. Algumas das metodologias mais comuns são [20] [1]:

- **Statement-oriented**

  Ensinar uma determinada linguagem de programação através do leccionamento de toda a sua sintaxe. Não importa a ordem pela qual é ensinada. Esta metodologia não promove nem diferencia a sintaxe ou estruturas lógicas mais importantes, nem se todos os elementos ensinados serão usados.

- **Applying as a tool**

  Focado em ensinar o algoritmo e estruturas lógicas do processo de programação, usa pseudocódigo e fluxogramas. Com ajuda das regras de sintaxe de uma determinada linguagem e respetivas convecções, programar poderá ser posteriormente automatizado.

- **Task type-oriented**

  Ensinar através da resolução de exercícios e tarefas de programação. Os elementos da linguagem necessários e exemplos introdutórios são introduzidos antes de cada exercício, não seguindo qualquer plano. Pode por vezes ser usado em conjunto com outras metodologias.

- **Language-oriented**

  Ensinar uma determinada linguagem de programação e os seus elementos numa ordem lógica.

- **Action-oriented**

  Ensinar através do uso de linguagens de baixo nível. Assim é possível mostrar o funcionamento detalhado das operações de programação.

- **Sample task-based**

  Ensinar através do uso de exemplos que contêm todos os elementos da linguagem e sintaxe necessários. Se não forem praticados várias vezes os elementos ensinados podem ser esquecidos com facilidade.

As metodologias anteriores cobrem aspetos distintos da aprendizagem de programação, mas cada uma delas possui falhas. A metodologia ou conjunto de metodologias a aplicar deve depender do nível dos alunos e também da dificuldade da unidade curricular de programação (unidades curriculares de programação avançada vão aplicar métodos diferentes das aulas de programação introdutórias).

2.2 DevOps

A internet disponibiliza inúmeros serviços, programas e aplicações a utilizadores e clientes. Com as crescentes necessidades dos clientes, *Service Level Agreement* (*SLAs*)[3] a cumprir e um mercado cada vez mais competitivo, esses serviços necessitam de ser mantidos e constantemente adaptados por equipas especializadas. Assim, podemos afirmar que além da área de desenvolvimento que se foca no design e desenvolvimento de software, existe claramente uma área que lida com o ciclo de vida inteiro de um produto/serviço, a sua operação, melhoramento e inclusive a sua descontinuação e final de vida.

Com o evoluir da indústria, tecnologias e metodologias, também a área de gestão e operação de serviços foi mudando. Começando pela metodologia *Lean*, proveniente do sistema de produção da Toyota, focado na criação de valor para o cliente ao construir um sistema robusto, eficiente e em constante atualização. Passando pelo manifesto *Agile*, que define um conjunto de princípios e valores claros e simples, para dividir em fases mais curtas e ágeis o processo de desenvolvimento de software e pela necessidade de tornar a infraestrutura cada vez mais ágil derivado da constante necessidade de mudança e de entrega de novos produtos e funcionalidades. Isto leva a que abordagens tradicionais como equipas de *sysadmins*, que operavam os sistemas e respondiam a eventos e problemas à medida que ocorriam, tendo a sua área de *expertise* exclusivamente na parte operacional, a transformar-se em abordagens mais preventivas. Estas abordagens mais adaptáveis à mudança e à utilização da automatização de tarefas com o intuito e de criar sistemas mais autónomos e robustos, levam ao surgimento de *Site Reliable Engineering* ou *SRE* [21].

As práticas *SRE* evoluíram ainda com a necessidade de eliminar e afinar a interação entre a parte operacional e parte de desenvolvimento de um projeto, dividindo todo o processo em fases. Com o surgimento de práticas como *continuous delivery (CI)*[4], *continuous deploy (CD)*[5] e o conceito de pipeline, surgiu assim – como hoje conhecemos – o *DevOps*.

O *DevOps*, constituído genericamente por 8 fases conforme mostra a Figura 2, junta a *expertise* operacional com a *expertise* de desenvolvimento de software, motivados pela rápida necessidade de inovação e adaptação. Cada ciclo de fases *DevOps* leva ao começo de outro ciclo formando uma sucessão infinita de iterações.

[3] *Service Level Agreement* é um acordo realizado entre cliente e fornecedor que visa garantir a qualidade dos serviços prestados perante um determinado parâmetro

[4] *Continuous Delivery* é um conjunto de práticas cujo objetivo é a configuração e o teste automático de novo código num determinado ambiente posterior ao ambiente de "produção". A passagem do novo código entre ambientes é manual.

[5] *Continuous Deploy* constitui a mesma prática que o *continuous delivery* mas que utiliza um *deploy* automático entre ambiente caso os testes sejam passados com sucesso.

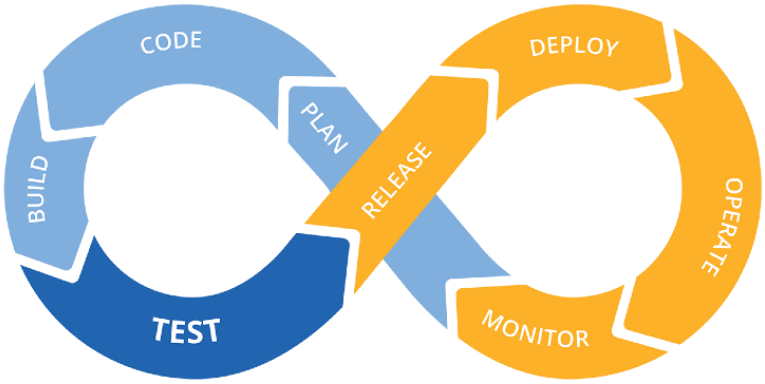


Figura 2 – Ciclo *DevOps*[6]

O conceito *DevOps* pode ser dividido ainda em dois níveis: princípios e ferramentas. Para criar um sistema escalável e ágil são envolvidas as partes operacionais e de desenvolvimento apoiadas em princípios como a automação, *CI*, *CD*, *infrastructure as code (IaC)*[7], centralização de configurações, monitorização, configuração de indicadores e métricas de performance ou arquiteturas de micro-serviços, implementadas por ferramentas como o *Docker*, *Git*, *Ansible*, *Jenkins*, *Nagios* entre outras [22].

Aplicar a filosofia *DevOps* significa que a equipa operacional e a equipa de desenvolvimento deixam de apenas desempenhar o seu papel tradicional e passam a preocupar-se com o produto na sua totalidade. Desde o seu desenvolvimento, operacionalização até à sua entrega, percebendo todos os processos envolventes.

Esta abordagem não deve ser adotada pelo simples uso de ferramentas. Para conseguir retirar valor desta metodologia é necessário entender os seus princípios e também a área de negócio ao qual se pretende aplicar os seus mecanismos e os seus problemas. Assim, é possível apoiar o desenvolvimento ágil, largamente adotado na indústria, e apoiar os seus valores na entrega de software de forma mais rápida e com melhorias ao longo de todo o ciclo de desenvolvimento do produto.

Entre as vantagens na aplicação do *DevOps* estão:

- **Colaboração**

  Aumentar a colaboração entre os membros de todas as equipas envolvidas em cada projeto. Isto permitirá evitar ao máximo: tempo despendido na comunicação e passagem de tarefas entre equipas. Assim, o tempo que cada equipa passa em espera é menor, permitindo melhorar a motivação e foco nas atividades a realizar. Ao deixar de ter áreas de conhecimento tão distintas entre as equipas, derivado da sua colaboração, é possível

[6] https://awkwardgen.com/wp-content/uploads/2021/07/pngegg-1.png
[7] *Infrastructure as Code* é à transformação de instruções de instalação em código que pode ser corrido sobre uma qualquer máquina e/ou plataforma

minimizar também a indisponibilidade de membros da equipa com os conhecimentos necessários para uma determinada tarefa.

- **Escalabilidade**

  Através da automatização de tarefas e dos automatismos existentes em diversas fases do projeto, é possível despender mais tempo na manutenção, evolução e escalabilidade do sistema e dos serviços. É também aconselhado o uso de arquiteturas de micro serviços, em substituição do tradicional uso de serviços monolíticos, o que permite ter mais versatilidade e facilidade na expansão de serviços e reformulação de arquiteturas.

- **Flexibilidade**

  O uso de micro serviços permite ter pequenos módulos configuráveis e de fácil expansão. Assim, é possível modificar, substituir ou atualizar mais agilmente serviços evitando *downtime* ou indisponibilidade, recorrendo à redundância entre módulos com *load balancing*, uma vez que facilmente podem ser lançados novos módulos/serviços. Estas características permitem uma maior flexibilidade às arquiteturas.

- **Fiabilidade**

  Com os mecanismos de automatização corretamente implementados e havendo colaboração entre todas as partes do projeto e o cliente é possível, mediante feedback, corrigir problemas ou bugs encontrados e recuperar mais rapidamente o sistema ou desenvolver uma nova versão, caso seja necessário.

- **Velocidade**

  Todos os princípios e vantagens realçadas anteriormente visam acrescentar mais versatilidade ao processo e diminuir o tempo de resposta em cada fase, acrescentado velocidade aos processos usados.

- **Cultura empresarial segura**

  Para incentivar uma cultura organizacional, com todas as vantagens referidas anteriormente, extremamente aberta à mudança e transformação, é necessário aceitar e permitir erros humanos inerentes ao processo de desenvolvimento e à operação de serviços. Os sistemas tornam-se grandes e com o tempo é difícil prever com total exatidão todos os seus outputs e resultados. O *DevOps* fomenta uma cultura regenerativa que aprende

com os seus erros, reconhece que eles são uma constante e que podem acontecer. Assim, o medo de cometer erros não prevalece nas equipas, sendo cultivado um ambiente seguro e apto à mudança.

- **Feedback**

  Obter feedback de todas as pessoas envolvidas em cada fase do ciclo de vida de um produto/serviço é fundamental para melhorar o processo e atender as necessidades. Este é um dos aspetos base para a aplicação de um sistema que é melhorado iterativamente e a base da metodologia *DevOps*.

A metodologia *DevOps* é uma junção de práticas, princípios e ferramentas para aumentar a capacidade de fornecer serviços mais rapidamente, de forma mais adaptada e fiável, do que utilizando os processos tradicionais de desenvolvimento.

2.3 Analogia entre ciclo de *DevOps* e ciclo de aprendizagem de programação

Apesar do *DevOps* ser uma metodologia especificamente usada na indústria e na distribuição de um produto onde existe uma aplicação ou funcionalidade, um cliente, uma equipa de desenvolver e suporte, os seus princípios poderão ser usados em outros processos/desafios.

A aprendizagem de programação é um processo que, segundo as análises e estudos realizados nos capítulos 2 e 3, possui diversos aspetos passiveis de melhoria e otimização.

No ensino de programação o professor cria um plano que contenha todos os elementos necessários de sintaxe, algoritmos, estruturas, bibliotecas e outros, para a aprendizagem de uma linguagem de programação traçando um plano inicial. Procura depois obter *output* sobre o processo de aprendizagem dos alunos e a efetividade dos conteúdos que lecionou. O objetivo é que o máximo de alunos atinjam o nível de competências em programação, definidos nos objetivos de aprendizagem da UC através dos conhecimentos transmitidos.

O professor, para além de expor a informação, também orienta os alunos, adaptando o seu processo de ensino ao feedback que vai recebendo. No entanto, a adaptação do processo usado pelo professor a cada aluno é praticamente impossível.

Além do grande número de alunos, o que impossibilita a individualização, nem todos os alunos vão expor a suas dúvidas ou dar feedback ao professor. O professor não dispõe de qualquer outro dado com o qual possa verificar as dificuldades de cada aluno.

Na aprendizagem de programação o aluno pretende aprender ao máximo os elementos de sintaxe de uma linguagem de programação e adquirir o conhecimento que é transmitido. Recebe o plano de estudos do professor assim como os seus inputs, e tem que se adaptar ao decorrer do plano e ao leccionamento das aulas. O seu objetivo é normalmente obter aprovação e não dar feedback sobre as suas dificuldades ou necessidades naquilo que está a aprender e no seu processo de aprendizagem.

Como descrito anteriormente, o *DevOps* pretende encurtar o ciclo de desenvolvimento de um produto desenvolvendo *features*, *fixes* e *updates* alinhados com os objetivos do negócio, necessidades e feedback do cliente. Uma abordagem que permite tornar o software mais resiliente e com menos bugs através de práticas como *continuous development*, *continuous integration*, *continuous testing*, *continuous deployment* e *continuous monitoring*. Para tornar o produto mais adaptado às necessidades do cliente é necessário envolvê-lo e obter o seu feedback, requisitos, prioridades e necessidades, assim como o seu compromisso e empenho.

Na aprendizagem de programação o cliente, neste caso – o aluno - não possui a visão total do plano de aprendizagem ou dos requisitos necessários. No entanto, sabe as suas capacidades assim como as suas dificuldades. O professor tem visibilidade do plano de aprendizagem, contudo, não da totalidade das dificuldades que são encontradas pelos alunos - o seu cliente. Esse feedback, que seria necessário, nem sempre é obtido. Quer pela enorme quantidade de alunos, quer pela sua desmotivação, desinteresse e não frequência das aulas ou dificuldade na resolução de exercícios pelos mais diversos motivos.

Este processo de ensino-aprendizagem de programação pode então consistir num conjunto de iterações, tal como acontece num projeto da indústria, alinhadas e encadeadas por um plano, por tópicos ou por feedback obtido, que são realizadas pelo aluno e administradas/operadas pelo professor. O aluno resolve os exercícios dados pelo professor de maneira a obter conhecimento. Em alguns conteúdos programáticos e tópicos terá mais dificuldades, logo deverá fazer mais exercícios, ou seja, mais iterações até atingir o objetivo de aprendizagem definido. Noutros conteúdos terá menos dificuldades, necessitando de resolver menos exercícios e menos iterações. Após ter alcançado um dos objetivos poderá avançar para outros, num processo de aprendizagem apoiado no leccionamento de matéria que segue o plano de estudos e em resoluções de exercícios alinhadas segundo cada conteúdo programático, tópico ou dificuldade.

As metodologias de ensino de programação abordadas anteriormente definem a forma como pode ser executado o plano e os métodos que se pretendem usar. Assim, podemos considerar que este processo as características apresentadas na Tabela 2.

Na sua generalidade, o plano de requisitos de aprendizagem de programação é dividido em pequenas tarefas de programação que se resumem à resolução de exercícios (prática de programação). Por exemplo, programar usando a linguagem C passa por aprender tipos de variáveis, operações básicas, ciclos, vetores entre outros elementos, até todos os elementos da linguagem de programação serem contemplados.

Tabela 2 - Âmbito do *DevOps* e do Ensino de programação

	Indústria - DevOps	*Ensino de programação*
Scope	Projeto ou tema	Ensino e aprendizagem de programação
Produto	Aplicação ou serviço	Aprender a programar
Cliente alvo	*Business to Business (B2B)* ou *Business to Consumer (B2C)*	Alunos
Stakeholders	Cliente ou Equipa interna	Professores

A Tabela 3, representa uma comparação entre aquilo que é realizado na indústria, em cada uma das fases da metodologia *DevOps*, e uma proposta de adaptação do processo de aprendizagem de programação apoiado em ferramentas e automatismos.

Tabela 3- *DevOps* no ensino da programação

	Indústria - DevOps	***Ensino de programação***
Plan	Requisitos e feedback são recolhidos das partes interessadas (*stakeholders*, cliente) para criar o *roadmap* do produto. O *roadmap* é depois dividido em tarefas que são planeadas e priorizadas segundos *sprints* antes de começar a próxima fase.	O professor: Identifica necessidades. Traça o plano de estudos em tópicos e exercícios. O aluno: Planeia o seu estudo para a semana e/ou para o próximo tópico com base no plano elaborado pelo professor.
Code	Codificação das tarefas planeadas anteriormente. São realizadas reuniões diárias para promover cooperação, interajuda e sincronizar do estado das tarefas que estão em curso.	O aluno: Estuda os conteúdos fornecidos como vídeos, exercícios resolvidos, tutoriais, exemplos e realiza exercícios de codificação utilizando a plataforma.
Build	Após codificar cada tarefa, o código, é integrado com o restante (recorrendo por exemplo ao *GIT*). Este processo é realizado após revisão e aprovação manual de outro membro da equipa. São corridos testes automáticos sobre todo o código com o objetivo de encontrar problemas o mais rápido possível e avisar o programador para que possa resolvê-los.	O aluno: Após realizar os exercícios obtêm algum feedback automático (ex: testes de resultado, análise estática de código) Pede ajuda ao professor caso não consiga resolver alguma etapa do exercício.
Test	O novo código é colocado num ambiente de testes. São aplicados princípios como *infrastructure as Code* (IaC) para instalar os serviços. São corridos testes de aceitação, unitários, de segurança, exploratórios e/ou outros.	O aluno: Verifica se o resultado obtido no exercício está de acordo com o esperado e cumpre o objetivo do exercício. Obtém ajuda do professor para as suas dúvidas caso existam.
Release	O código está pronto para ser colocado no ambiente do cliente. Este processo pode ser manual ou automático mediante o controlo que se pretenda ter.	O aluno: Submete como concluído o seu trabalho e resoluções de exercícios na plataforma. Pode voltar à fase *Code*, até terminar todas as tarefas definidas pelo professor.
Deploy	Nesta fase o código chega ao ambiente do cliente. Poderá ainda passar vários ambientes do cliente até chegaram ao ambiente de "Produção". São feitos, caso necessário, novos testes exploratórios e afinações antes do *deploy* final para tentar que não exista qualquer *downtime* ou problema.	O professor: Analisa o progresso dos alunos através das resoluções de exercícios submetidas e outras informações como tempos de estudo e resolução presentes na plataforma. Analisa também manualmente o código dos exercícios submetidos.
Operate	Nesta fase é garantida a operacionalidade do ambiente através de ferramentas de monitorização. São também recolhidas métricas do sistema e *feedback* do cliente e de todas as partes interessadas no produto.	O professor: Análise progresso dos alunos. Dá feedback personalizado a cada aluno, ajustando os próximos objetivos de aprendizagem se necessário;
Monitor	Nesta fase, é dado suporte e monitorizado ao ambiente com base em parâmetros definidos no feedback recolhido anteriormente. É também mantida a pipeline e ferramentas usadas nas fases anteriores.	O professor: Com base no percurso anterior dos alunos, define novos objetivos de aprendizagem, por semana, por tema ou pelas necessidades especificas de cada aluno. Identifica os elementos de estudo e define tarefas a cumprir.

Aplicando as fases do ciclo anteriormente descritas ao processo de ensino de programação, poderiam ser percorridos iterações deste ciclo de fases até todas as tarefas necessárias à aprendizagem de uma determinada linguagem de programação, sintaxe e métodos estarem concluídas por parte do aluno. A Figura 3, ilustra a aplicação das fases descritas na Tabela 3.

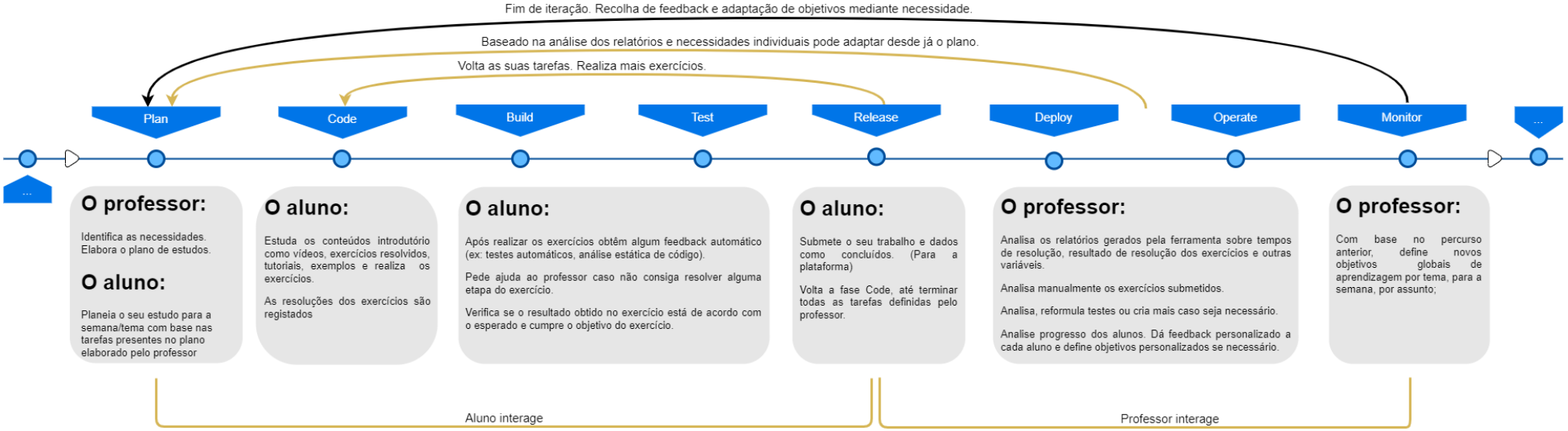


Figura 3 - Adaptação *DevOps* para o ensino de programação

3 Estudo do problema

Neste capítulo são analisadas algumas unidades curriculares de programação lecionadas no ISEC e os possíveis pontos chave da aprendizagem de programação.

3.1 Ensino e aprendizagem de programação no ISEC

O Instituto Superior de Engenharia de Coimbra, unidade orgânica de ensino do Instituto Politécnico de Coimbra, foi fundado no ano de 1974. O curso de bacharelato em Engenharia Informática e de Sistemas foi criado em 1989. Com a alteração da Lei de Bases do Sistema Educativo passou a ser uma licenciatura bietápica, constituídas por um 1º ciclo, com a duração de três anos - que confere o grau de bacharel - e um 2º ciclo, com duração de dois anos que confere o grau de Licenciatura. Tendo sido começado a ser lecionados no ano letivo de 1998/99.

No ano letivo de 2006/2007 as licenciaturas bietápicas foram adequadas ao modelo de Bolonha. Deste trabalho de adequação resultaram também propostas de vários mestrados. [23]

No ano letivo de 2018/2019 houve 240 alunos colocados no total das fases de candidatura às variantes existentes do curso de engenharia informática e 1024 alunos inscritos na totalidade (814 de Regime Diurno, 147 de Pós-laboral e 63 de Curso Europeu). Os documentos de avaliação disponíveis na plataforma Sistema Interno de Garantia de Qualidade (SIGQ) do curso de Engenharia Informática mostram taxas de insucesso de 43% nas unidades curriculares de programação, especialmente nas introdutórias, como Introdução à Programação e Programação [10] [24] [25].

No ano letivo de 2019/2020 houve 184 alunos colocados no total das fases de candidatura aos cursos existentes de Engenharia Informática e 1058 alunos inscritos na totalidade (838 de Regime Diurno, 147 de Pós-laboral e 71 de Curso Europeu). Os documentos de avaliação disponíveis na plataforma SIGQ do curso de Engenharia Informática, relativos a este ano letivo, realçam novamente taxas de insucesso algo elevadas em disciplinas de programação como Introdução à Programação e Programação Orientada a Objetos [26] [27] [28].

Para analisar os resultados obtidos e metodologias usadas, foram escolhidas as disciplinas de Introdução à Programação, Programação, Programação Orientada a Objetos.

3.1.1 Introdução à Programação

Esta unidade curricular aplicou genericamente as metodologias *language-oriented* em aulas teóricas e nas aulas práticas seguiu uma metodologia *applying as tool* ou *sample task-based.* Esta informação corresponde a uma análise externa para entender o funcionamento das aulas teóricas e práticas, não confirmada por parte do docente da unidade curricular.

3.1.1.1 Ano letivo 2018/2019

A unidade curricular de Introdução à Programação (IP) contou, no ano letivo de 2018/2019, com 243 alunos inscritos, dos quais 12 alunos anularam a matrícula e 99 alunos obtiveram aprovação no final da época normal e de recurso.

Na Figura 4, é possível verificar o número de alunos aprovados, reprovados, que faltaram, que desistiram ou que anularam a matrícula por cada época de avaliação [29].

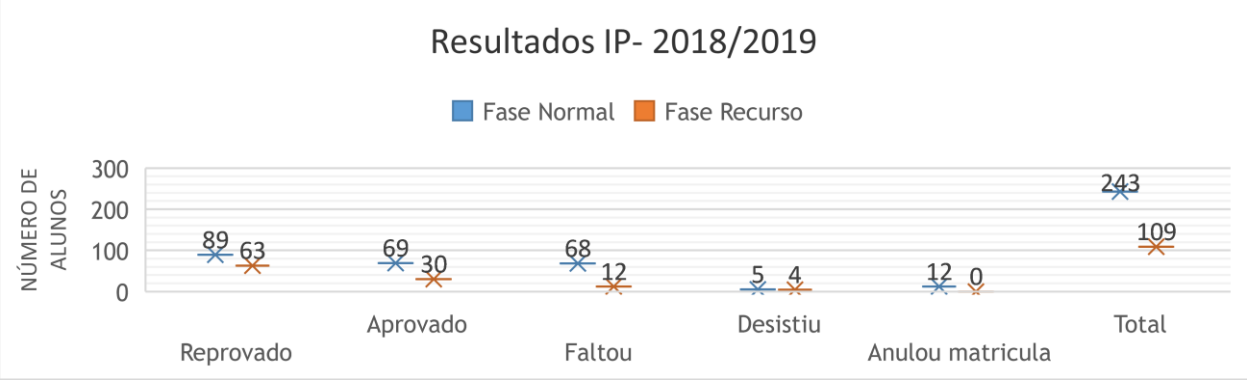


Figura 4 – Introdução a Programação em 2018/2019

No documento SIGQ [10] é mostrado que a unidade curricular de Introdução à Programação possui uma taxa de insucesso de 42%, fazendo a seguinte análise:

- *Os alunos deveriam ser mais assíduos às aulas (com especial destaque para as aulas teóricas).*
- *Alguma disparidade na formação base na área de programação dos alunos que frequentam esta UC.*
- *Alguns alunos poderiam ter melhor aproveitamento se tivessem mais interesse na programação (que é uma área fulcral do curso).*

São também apresentadas as seguintes medidas para melhoraria do aproveitamento dos estudantes:

1. *Maior apoio aos alunos para os ajudar a analisar e iniciar a resolução dos exercícios;*
2. *Deve motivar-se os alunos a serem assíduos especialmente às aulas Teóricas (dado que muitos alunos faltam às aulas Teóricas);*
3. *Promoção, a nível institucional, do trabalho/estudo e assiduidade com Workshop, palestras ou concurso de programação (ex. Para novos alunos);*

É destacado também que existem muitos alunos inscritos que não se submetem à avaliação, sendo importante tentar motivar estes alunos.

3.1.1.2 Ano letivo 2019/2020

No ano letivo 2019/2020, o documento SIGQ [26] mostra que a unidade curricular de Introdução à Programação possui uma taxa de insucesso de 38%, com as seguintes fundamentações:

- *"A baixa assiduidade dos alunos às aulas, a disparidade na formação de base dos alunos na área de informática, e a falta de apetência/habilidade de alguns alunos para a área de programação limita um melhor desempenho global. Além disso, há alunos que ingressam no Curso da LEI por uma via que não é o Concurso Nacional de Acesso (1ª e 2ª fase do Concurso dos Maiores de 23, 1ª e 2ª fase dos Concursos Especiais, Concurso de Mudança de Par Instituição/Curso, 1ª e 2ª fase do Concurso Especial para Titulares de Diploma das Vias Profissionalizantes, etc.). Muitos destes alunos só se matriculam/inscrevem em outubro, novembro, dezembro… e vão aparecendo nas aulas ao longo do semestre. Não é espectável que um aluno que começa a frequentar as aulas "a meio" vá ter o mesmo aproveitamento que teria se começasse no início do semestre."*

São também apresentadas as seguintes medidas para melhoraria do aproveitamento dos estudantes:

1. *A presença dos alunos nas aulas é fundamental para o processo de aprendizagem. Deve motivar-se os alunos a serem assíduos quer às aulas Teóricas quer às Práticas/Laboratoriais (embora muitos alunos faltem maioritariamente às aulas Teóricas).*
2. *Tornar mais exigentes os critérios de acesso ao Curso, nomeadamente através da diminuição do número de vagas no concurso nacional de acesso.*
3. *Diminuir o número de vagas (ou considerar mesmo vagas zero) para os diversos regimes de acesso a este Curso de licenciatura que não seja pela via do Concurso Nacional de Acesso.*
4. *Além disso, o Calendário/Prazos de candidatura destes "outros" Concursos deveriam permitir que estes alunos se matriculassem/inscrevessem antes do início das aulas das licenciaturas correspondentes (não se justificando a existência de concursos a decorrer já em período letivo).*
5. *As candidaturas ao regime de Reingresso deveriam permitir que os alunos se matriculassem/inscrevessem antes do início das aulas de cada um dos semestres das licenciaturas correspondentes (e não a meio de um semestre, o que leva os alunos a ainda querer "aproveitar" as aulas finais desse semestre e a fazer os respetivos exames).*
6. *A Unidade Curricular de "Introdução à Programação" pertence ao 1º ano/ 1º semestre do plano de estudos da Licenciatura em Engenharia Informática. Sendo uma UC basilar da área de programação deste curso, a não aquisição dos conhecimentos necessá-*

rios poderá repercutir-se nos resultados obtidos pelos alunos noutras UCs da licenciatura, nomeadamente nas UCs da Programação, Programação Orientada a Objetos e Sistemas Operativos. Nesse sentido, seria importante considerar a possibilidade de funcionamento em regime deslizante da UC de IP, de modo a permitir que alunos que não conseguiram "acompanhar" a UC no 1º semestre o possam fazer no 2º semestre.

Não foi possível recolher mais dados sobre a disciplina neste ano letivo.

3.1.2 Programação

Esta unidade curricular aplicou genericamente a metodologia *language-oriented* em aulas teóricas e *task type-oriented* ou sample *task-based* nas aulas práticas. Esta informação corresponde a uma análise externa para entender o funcionamento das aulas teóricas e práticas, não confirmada por parte do docente da unidade curricular.

3.1.2.1 Ano letivo 2018/2019

A unidade curricular de Programação (P) contou, no ano letivo de 2018/2019. com 484 alunos inscritos, dos quais 15 alunos anularam a matrícula e 77 alunos obtiveram aprovação na fase normal e de recurso.

Na Figura 5, é possível verificar o número de alunos aprovados, reprovados, que faltaram, que desistiram ou que anularam a matrícula por cada época de avaliação [29].

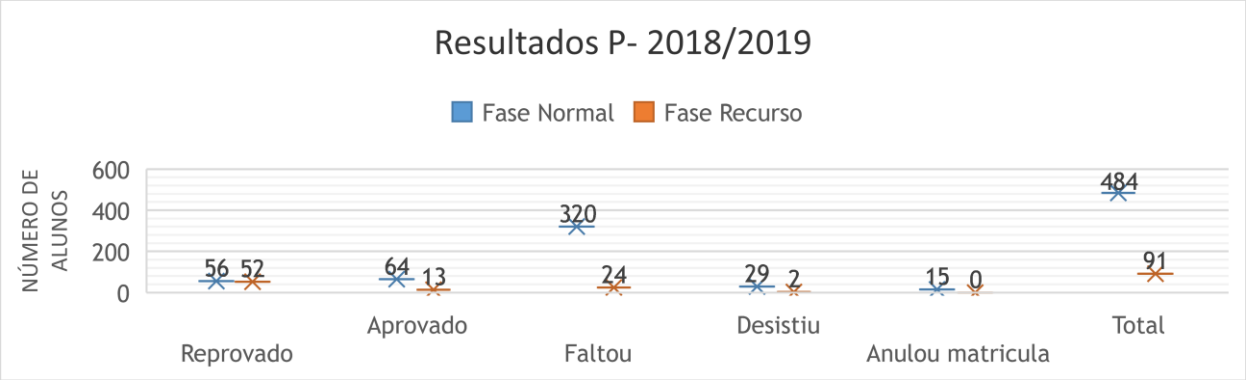


Figura 5 – Programação em 2018/2019

No documento do SIGQ [10] é mostrado que a unidade curricular de Programação possui uma taxa de insucesso de 47% com as seguintes fundamentações:

- *O número de alunos inscritos na UC é completamente inadequado para os recursos existentes, tanto docentes, como de instalações;*
- *Dadas as circunstâncias atuais de funcionamento, não é possível delinear e implementar estratégias pedagógicas que permitam melhorar o sucesso escolar;*

- *Falta de preparação dos alunos.*

São também apresentadas as seguintes medidas para melhoraria do aproveitamento dos estudantes:

1. *Aumentar o número de horas laboratoriais;*
2. *Criar condições para o funcionamento regular das aulas, através do reforço significativo do corpo docente afeto à unidade curricular e/ou diminuição do número de alunos inscritos;*
3. *Implementar precedências no curso de forma a garantir que os alunos chegam com a preparação mínima necessária para as matérias abordadas;*
4. *Tentar incluir metas no trabalho de programação.*

3.1.2.2 Ano letivo 2019/2020

A unidade curricular não está referenciada no documento SIGQ [26] do ano letivo 2019/2020 como uma das unidades curriculares com reprovação superior a 30%, não tendo sido possível recolher mais dados sobre a disciplina neste ano letivo.

3.1.3 Programação Orientada a Objectos

Esta unidade curricular aplicou genericamente as metodologias *language-oriented* em aulas teóricas e *task type-oriented* ou *sample task-based* nas aulas práticas. Esta informação corresponde a uma análise externa para entender o funcionamento das aulas teóricas e práticas, não confirmada por parte do docente da unidade curricular.

3.1.3.1 Ano letivo 2018/2019

A unidade curricular de Programação Orientada a Objetos (POO) contou, no ano letivo de 2018/2019, com 364 alunos inscritos, dos quais 13 anularam a matrícula e 84 obtiveram aprovação na fase normal e de recurso.

Na Figura 6, é possível verificar o número de alunos aprovados, reprovados, que faltaram, que desistiram ou que anularam a matrícula, segmentados por época de avaliação [29].

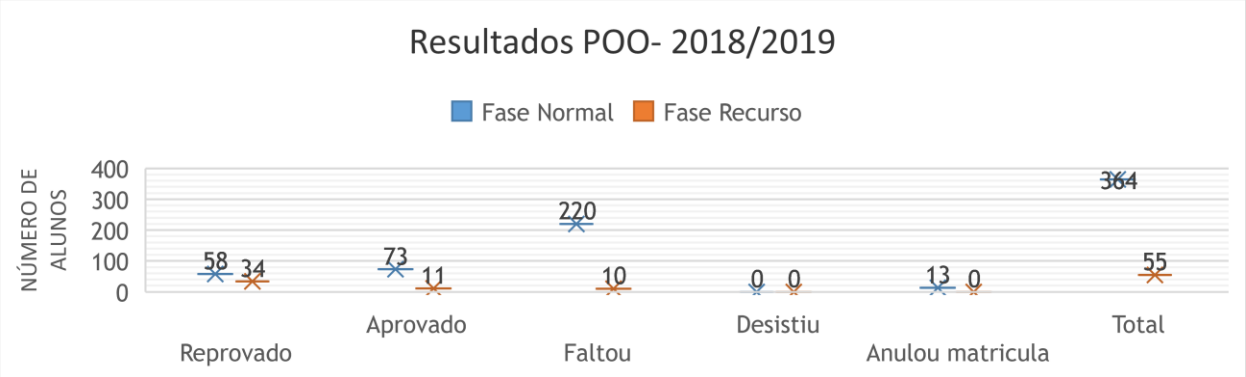


Figura 6 – Programação Orientada a Objetos em 2018/2019

No documento do SIGQ [10] é mostrado que a unidade curricular de Programação Orientada a Objetos possui uma taxa de insucesso de 41% com as seguintes fundamentações:

- *A não existência de precedências leva à frequência de alunos que não têm os conhecimentos pressupostos para o sucesso à disciplina.*

São também apresentadas as seguintes medidas para melhoraria do aproveitamento dos estudantes:

1. *A existência de precedências obrigatórias ou aconselhadas levaria a um melhor aproveitamento;*
2. *Aumentar número de laboratórios.*

3.1.3.2 Ano letivo 2019/2020

No ano letivo 2019/2020 o documento SIGQ [26] mostra que a disciplina de Programação Orientada a Objetos possui uma taxa de insucesso de 34% com as seguintes fundamentações:

- *Muitos estudantes frequentam esta unidade curricular sem ter obtido aprovação em unidades curriculares (Introdução à Programação e Programação) que lhes proporcionariam as competências necessárias para um bom aproveitamento.*

São também apresentadas as seguintes medidas para melhoraria do aproveitamento dos estudantes:

1. *Muitos dos alunos que frequentam Programação Orientada a Objetos não obtiveram aprovação às disciplinas de Programação do 1º ano. Desta maneira é difícil acompanharem esta disciplina que pressupõe adquiridos os conhecimentos das disciplinas de Programação do 1º ano. Estes alunos dispersam a sua atenção entre as disciplinas que precisariam de sedimentar (do 1º ano) e aquelas que ainda não estão em condições de acompanhar bem (do 2º ano), fazendo uma má gestão do seu tempo e esforço. Para que o esforço destes alunos fosse focado e tivesse rendimento, uma proposta de plano*

de atuação seria a existência de precedências obrigatórias ou fortemente aconselhadas. De forma espontânea os alunos não têm a visão em perspetiva nem a maturidade suficiente para a melhor gestão do seu esforço.

Ambas as unidades curriculares possuem taxas de insucesso superiores a 30%, sendo realçado que os alunos nem sempre demonstram o interesse necessário para conseguirem obter os conhecimentos necessários e serem aprovados. Assim, a participação dos alunos nas aulas começa também a diminuir, levando a que os mesmos deixem essa unidade curricular para trás.

Outro aspeto que pode ser realçado é que se uma das unidades curriculares não for realizada com sucesso, vai impactar o aproveitamento das restantes (É necessário fazer IP para um bom aproveitamento a P, sendo necessário fazer IP e P para um bom aproveitamento a POO)

Medidas como reduzir o número de alunos a frequentar as unidades curriculares, diminuir o número de alunos por cada turma ou aumentar o número de turmas são pontos comuns.

3.2 Análise de pontos chave na aprendizagem de programação

Com base nos artigos [1] [2] [3] [13] [14] [17] [18] [19] [20] citados anteriormente, capítulo 3.1 e outos aspetos relativos ao insucesso e sucesso nas unidade curriculares de Introdução à Programação, Programação e Programação Orientada a Objetos, foi possível identificar e reunir um conjunto de aspetos que aparentam ser os mais relevantes para a aprendizagem da programação, ilustrados na Figura 7.

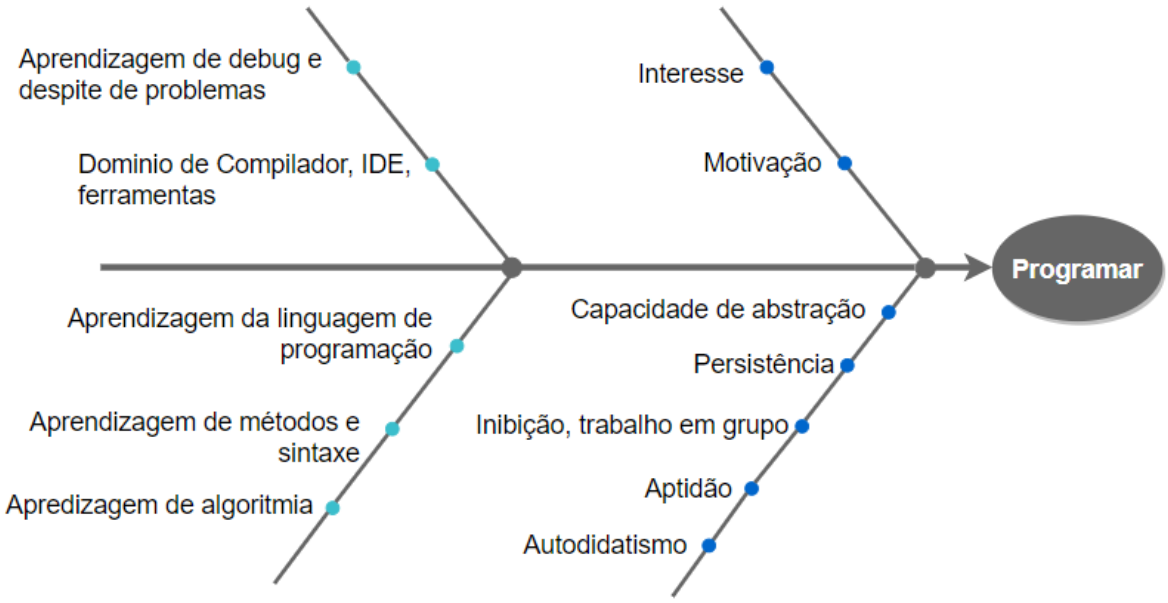


Figura 7 – Diagrama de fatores “relevantes” para programar

Após a elaboração destes pontos foi realizada uma reunião para obtenção de feedback com os professores das disciplinas observadas, com o objetivo de validar se os pontos e aspetos da Figura 7 podem ser os mais relevantes, atendendo ao seu ponto de vista.

É realçado pelos professores que todos os pontos presentes na Figura 7 são importantes e têm um papel relevante no percurso de qualquer aluno. No entanto, o interesse e motivação na aprendizagem de programação, o domínio de algoritmia e a capacidade de abstração na escrita de código são os aspetos que unanimemente, segundo os professores, permitem aos alunos obter melhores resultados.

Os alunos que melhor dominam a algoritmia conseguem também mais facilmente atingir um nível de abstração para encontrarem soluções para os exercícios e problemas propostos.

Outros aspetos como a aprendizagem de métodos e sintaxe de uma linguagem de programação, domínio de compilador/*IDE* ou *debug* de código são essenciais para programar, mas são possíveis de ser alcançados e melhorados através da resolução de exercícios práticos.

No campo das habilidades, estes aspetos também são adquiridos com a prática e sobretudo com a persistência na resolução de exercícios.

4 Plataforma de ajuda ao ensino-aprendizagem de programação

Para além de ser necessário aplicar os métodos mais adequados no processo do ensino de programação, também as ferramentas e tecnologias podem e devem ser tidas em conta [15]. O leccionamento de aulas pode usar estas tecnologias, como as plataformas online de disponibilização de conteúdos ou o ensino à distância, para melhorar o processo de aprendizagem [15], permitindo um ensino mais específico às necessidades de cada aluno e também ao seu gosto.

É essencial criar e usar ferramentas de apoio ao ensino-aprendizagem de programação. Assim, o objetivo deste capítulo é o estudo e criação de uma plataforma que será apelidada de *LMSOps.*

4.1 Ferramentas open-source de e-learning para programação

Para além dos processos e métodos mais adequados para o ensino de programação, também as ferramentas e tecnologias são cada vez mais usadas. Tradicionalmente, o processo de educação conta com três elementos - estudantes, professores e a sala de aula - mas com as crescentes ferramentas e conteúdos que podem ser disponibilizados na *internet,* o ensino à distância está a ser cada vez mais usado, nomeadamente na programação.

Alunos e aulas podem ou mesmo devem conseguir usar estas tecnologias para melhorar a aprendizagem e tentar colmatar problemas e dificuldades, permitindo um ensino mais específico e adaptado às necessidades de cada aluno e também ao seu gosto.

Existem diversas ferramentas online que já pretendem cumprir o objetivo de ajudar no ensino-aprendizagem de programação à distância, seja por vídeos-tutoriais, exercícios explicados, slides ou ambiente *IDE*/compilador online, que podem ser combinadas com as aulas tradicionais.

O conteúdo disponibilizado é normalmente organizado em cursos e pretende dar conhecimento sobre uma determinada matéria, linguagem de programação, metodologia ou ferramenta. O número de *Learning Management Systems (LMS)* tem crescido ao longo dos últimos anos e apresenta inúmeras oportunidades a estudantes e programadores profissionais que pretendam adquirir ou desenvolver as suas aptidões [15].

4.1.1 Udemy

Udemy, https://www.udemy.com/pt/, é uma plataforma que disponibiliza cursos online de diversas áreas construindo uma comunidade de instrutores e utilizadores. Cada utilizador da plataforma, para além de poder subscrever cursos, pode também disponibilizar cursos na plataforma sobre um tópico no qual tenha conhecimento. Cada curso está dividido em diversos conteúdos e pode apresentar vídeos, conteúdo teórico ou exercícios de escolha múltipla e/submissão. A Figura 8, apresenta um exemplo da interface de um curso.

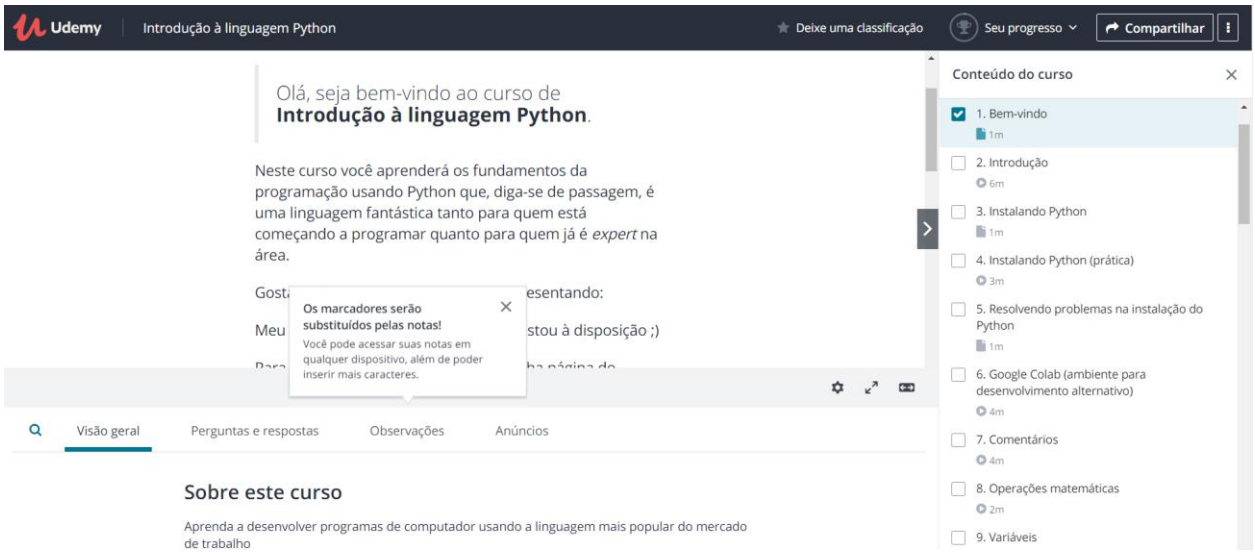


Figura 8 - Interface *Udemy*

4.1.2 CodeCademy

CodeCademy, https://www.codecademy.com/, é uma plataforma que disponibiliza um catálogo de cursos orientados a diversas áreas de conhecimento, disponibilizando a construção de *career paths* adaptados às necessidades de cada utilizador. Possui cursos de acesso livre e de acesso premium, limitado aos utilizadores com subscrição. Os cursos estão divididos por capítulos teóricos com vídeos e guias teóricos e capítulos práticos orientados à realização de exercícios práticos via editor *inline* e interagindo com o utilizador através de dicas e feedback automático. A Figura 9, apresenta um exemplo da interface desta plataforma.

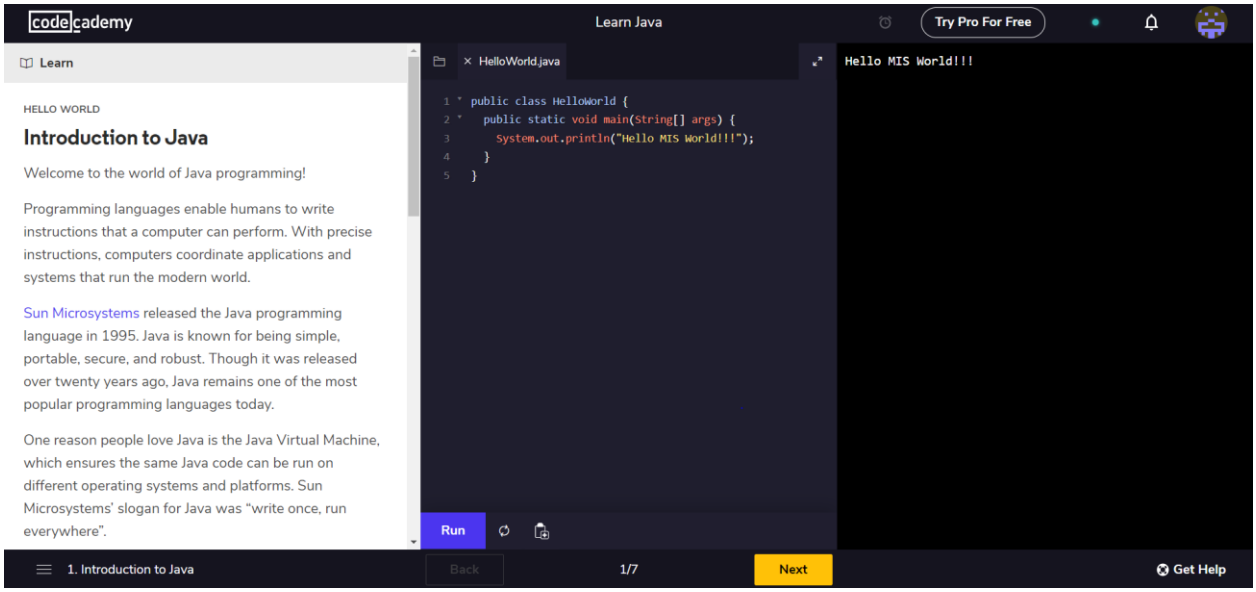


Figura 9 - Interface *CodeCademy*

4.1.3 edX

edX, https://www.edx.org/, é uma plataforma que disponibiliza um catálogo de cursos, graduações e *bootcamps* online de diversas áreas de conhecimento, recorrendo a protocolos com diversas instituições conceituadas. Os cursos estão divididos por capítulos teóricos com dicas e vídeos e são orientados à realização de exercício de escolha múltipla com feedback automático. A Figura 10, apresenta um exemplo da interface dum curso desta plataforma.

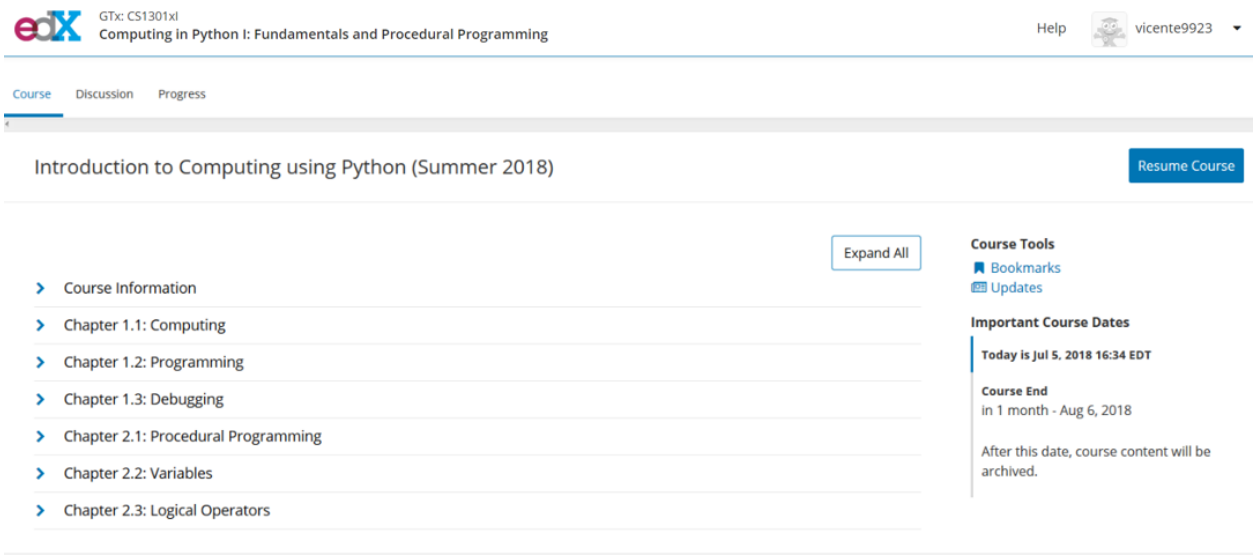


Figura 10 - Interface *edX*

4.1.4 DataCamp

DataCamp, https://www.datacamp.com/, é uma plataforma com cursos de *data science* e análise de dados, *Python* ou *SQL*. Os cursos estão divididos em capítulos teóricos e capítulos práticos recorrendo a um editor *inline* com exercícios codificados ou exercícios de escolha múltipla que dão feedback automático ao utilizador. A Figura 11, apresenta um exemplo da interface de um curso desta plataforma.

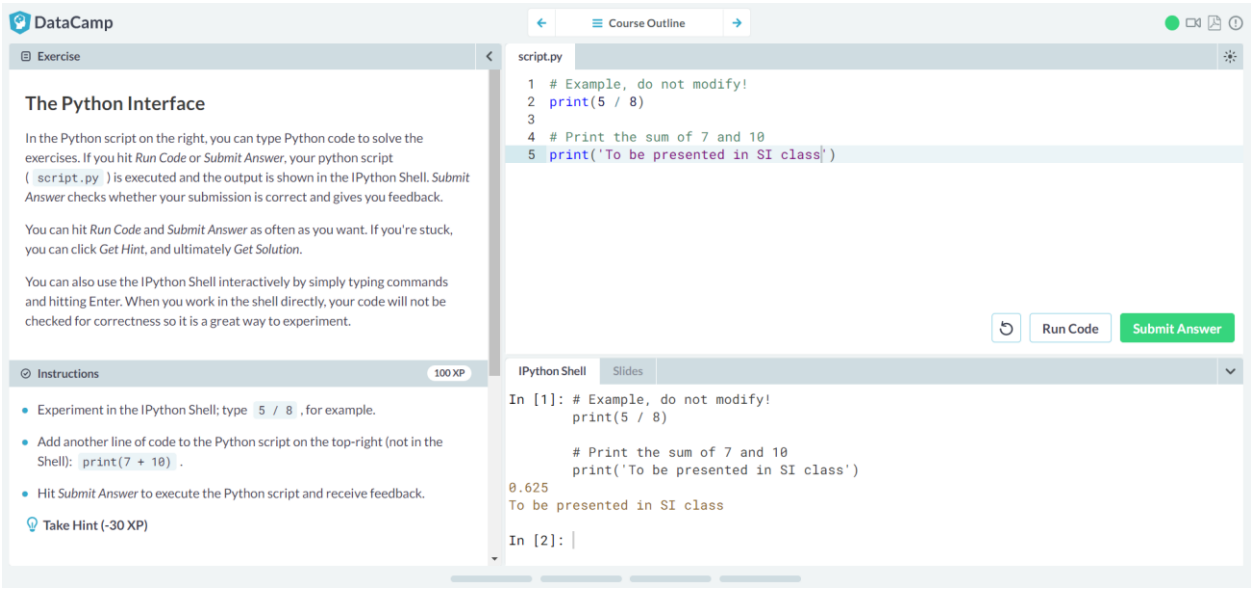


Figura 11- Interface DataCamp

A Tabela 4, realiza uma comparação genérica entre as funcionalidades das plataformas *Udemy, CodeCademy, edX e DataCamp.*

Tabela 4 - Comparação entre plataformas *LMS*

	Metodologia	***Funcionalidades***	***Motivação***	***Feedback/Comunicação***
Udemy	*Statement-oriented* ou *Sample task-based* (Dependendo do curso)	Vídeos com explicação passo-a-passo dos exercícios. Exercícios de escolha múltipla.	Explicações e demonstrações detalhadas através dos vídeos;	Correções automáticas dos exercícios. *Feedback* ao curso e/ou instrutor do curso;
codecademy	*Task type-oriented* ou *Sample task-oriented*	Plataforma com *IDE* integrado; Testes automáticos do código com dicas e explicações; Testes de resultado final e objetivo do código;	Dicas automáticas; Possibilidade de mostrar a solução; Organizados em lições pequenas ou exercícios;	Correções automáticas dos exercícios. Fórum de comunicação entre utilizadores;
edX	*Statement-oriented* ou *Sample task-based* (Dependendo do curso)	Vídeos; Exercícios online;	Exercícios e fundamentos teóricos; Organizados em lições e capítulos; Possibilidade de obter certificação;	Correções automáticas dos exercícios. *Feedback* ao curso;
DataCamp	*Task type-oriented* ou *Sample task-oriented*	Plataforma com *IDE* integrado; Exercícios com instruções e dicas; Slides com explicação teórica;	Exercícios e fundamentos teóricos; Possibilidade de mostrar a solução; Organizados em lições e capítulos;	Correções automáticas dos exercícios. *Feedback* ao curso e as ajudas que podem ser dadas nos exercícios.

Atendendo às fases do *DevOps* aplicado ao ensino-aprendizagem de programação, apresentadas no capítulo 3.2, a Tabela 5 apresenta sucintamente a forma como cada uma destas plataformas concretiza cada fase e os mecanismos que usa.

Tabela 5- Funcionalidades das plataformas em cada fase do ciclo DevOps

	E-learning de programação
Plan	*Udemy*- Cursos por tópico ou linguagem de programação com diversas tarefas, vídeo-tutoriais e exemplos de resolução de exercícios. *CodeCademy*- Cursos por tópico ou linguagem de programação divididos em tarefas, conteúdo teórico e introdutório. *EDx*- Cursos por tópico ou linguagem de programação divididos em tarefas, conteúdo teórico e introdutório. *DataCamp*- Cursos por tópico ou linguagem de programação divididos em tarefas, conteúdo teórico e introdutório.
Code	*Udemy*- Vídeo-tutorias explicativos com todos os passos necessários. *CodeCademy*- IDE integrado no browser. *EDx*- IDE integrado, mas em ferramenta à parte/adicional. *DataCamp*- IDE integrado no browser.
Build	*Udemy*- É usado o IDE local e verificado o resultado manualmente comparando com o resultado do tutorial. *CodeCademy*- O IDE possui testes automáticos para verificar o resultado do código e dar dicas. *EDx*- O IDE apenas corre o código, realizando alguns *auto-completes* nas funções se necessário. *DataCamp*- O IDE possui testes automáticos para verificar o resultado do código, e dar dicas.
Test	*Udemy*- Sem existência de testes ou testes automáticos contra o resultado do programa. *CodeCademy*- Testes automáticos contra o resultado do programa. *EDx*- Faz questões sobre os exercícios realizados. *DataCamp*- Testes automáticos contra o resultado do programa.
Release	*Udemy*- É na generalidade marcado como concluído pelo utilizador ou automaticamente. *CodeCademy*- Exercício fico como realizada na plataforma/curso após passar os testes. *EDx*- Capítulo do exercício e exercício fica como realizado na plataforma/curso após passar o questionário final. *DataCamp*- Exercício fico como realizada na plataforma/curso após passar os testes.
Deploy	*Udemy*- Exercício/capítulo fica como "resolvido", podendo ser revisto. *CodeCademy*- Após resolver cada exercício ficam disponíveis os seguintes. *EDx*- Capítulo do exercício é dado como resolvido apos todas as tarefas estarem realizadas. *DataCamp*- Após resolver cada exercício ficam disponíveis os seguintes.
Operate	*Udemy*- São sugeridos cursos idênticos ou no mesmo leque de tecnologias/aptidões. *CodeCademy*- São sugeridos cursos idênticos ou no mesmo leque de tecnologias/aptidões. *EDx*- São sugeridos cursos idênticos ou no mesmo leque de tecnologias/aptidões. *DataCamp*- São sugeridos cursos idênticos ou no mesmo leque de tecnologias/aptidões.
Monitor	*Udemy*- É sugerido deixar feedback à plataforma, exercícios e curso. *CodeCademy*- É sugerido deixar feedback à plataforma, exercícios e curso. *EDx*- É sugerido realizar upgrade para plano pago (com direito a certificado). *DataCamp*- É sugerido deixar feedback à plataforma, exercícios e curso.

4.2 Análise entre ferramentas *DevOps* e ferramentas para o ensino-aprendizagem de programação

Para conseguir implementar as fases definidas capítulo 2.3 é necessário apoiar o projeto na *toolchain DevOps*. É necessário explorar e apropriar as ferramentas aos problemas recolhidos anteriormente do ensino e aprendizagem de programação. Assim, cada fase poderá usar mecanismos e ferramentas diferentes que se interliguem entre si.

A Tabela 6, possui uma comparação entre algumas das ferramentas usadas na indústria pelo *DevOps* - que podem ser adaptadas para o ensino de programação - e os objetivos das ferramentas para cada fase do ciclo no ensino de programação.

Tabela 6 - Análise de ferramentas DevOps e ferramentas para o ensino-aprendizagem de programação

	Indústria –Ferramentas usadas em DevOps	***Possíveis ferramentas a usar no ensino-aprendizagem de programação***
Plan	Ferramentas de gestão, planeamento de projeto e colaboração como: *Slack, Trello, Redmine, Confluence*	Ferramentas que permitam criar, gerir e acompanhar o percurso de aprendizagem do aluno na resolução de resolução de exercícios. Por exemplo o *Moodle, Matjax*.
Code	Ferramentas que permitem desenvolvimento, gestão e versionamento de código como: *GIT, SVN, Confluence, Eclipse*.	Ferramentas que permitam codificar os exercícios e versionar cada resolução. Por exemplo, o *GIT*.
Build	Ferramentas especificas de *building* que se interligam à linguagem de programação e tecnologias usadas como: *Maven, Ant, Gradle*	Ferramentas que permitam ao aluno submeter, editar ou rever o código realizado para cada exercício e obter *feedback*. Por exemplo, o *Code Runner (plugin Moodle), Virtual programming lab (plugin Moodle), JupyterLab, rstudio, theia*.
Test	Ferramentas que permitem desenvolver e codificar testes para performance, qualidade de código e configuração: *Selenium, JUnit, Cucumber*.	Ferramentas que permitam testar as resoluções dos exercícios submetidas. Por exemplo, o *CodeRunner (plugin Moodle), Virtual programming lab (plugin Moodle), JupyterLab, rstudio, theia*.
Release	Ferramentas que permitem fazer *release* do código de forma automática e controlada: *Jenkins, Bamboo, Codeship, CircleCI*.	Ferramentas que permitam ao aluno terminar o exercício submetendo o resultado. Por exemplo, o *Code Runner (plugin Moodle), Virtual programming lab (plugin Moodle), JupyterLab, rstudio, theia*.
Deploy	Ferramentas que permitem instalar os artefactos gerados anteriormente: *Ansible, Puppet, Chef, Docker, AWS*	Ferramentas que permitam gerar relatórios dos exercícios submetidos por cada aluno e aglomerar algumas estatísticas. Por exemplo, o *Moodle, Matjax*.
Operate	Ferramentas que permitem operar as aplicações instaladas anteriormente: *Kubernets, Ansible, Chef, Docker-Compose*	Ferramentas que permitam atualizar os exercícios existentes e criar novos recursos de aprendizagem. Por exemplo, o *Moodle, Matjax*.
Monitor	Ferramentas que permitem monitorizar, obter métricas de funcionamento das aplicações/software: *Nagios, Grafana, Prometheus, Kibana*	Ferramentas que permitam aglomerar as estatísticas anteriores ajudando na tomada de decisão de acompanhamento a cada aluno. Por exemplo, o *Moodle, Matjax, Grafana.*

4.3 Análise de requisitos

Atendendo aos pontos chave da aprendizagem de programação no ensino superior, levantados no capítulo 3.2 e as restantes análises feitas nos capítulos 4.1 e 4.2, foi traçado na Tabela 7 um plano de requisitos necessários para a plataforma.

Tabela 7 - Requisitos da plataforma

Requisitos funcionais	1. *O sistema deve permitir a disponibilização de conteúdo teórico;* 2. *O sistema deve permitir a disponibilização de conteúdo pratico/atividades a ser resolvidas e submetidas por alunos da plataforma;* 3. *O sistema deve permitir que o professor configure 3 tipos de atividades: questionários teóricos, exercícios práticos codificados e compilados através do browser e submissão de trabalhos escritos;* 4. *O sistema deve permitir a troca de mensagens entre professores e alunos com o intuito de pedir ajuda para os exercícios e estabelecer comunicação pedagógica de aprendizagem;* 5. *O sistema deve permitir definir pelo professor testes automáticos para cada exercício prático, para assim validar as submissões dos alunos. Poderá também permitir que os alunos criem os seus próprios testes para os exercícios que codificam;* 6. *O sistema deve permitir análise estática de código submetido pelos alunos;* 7. *O sistema deve permitir a elaboração de relatórios do percurso de cada aluno (exercícios realizados, número de submissões necessários, tempo despendido, testes passados e falhados, lógica do código);* 8. *O sistema deve permitir a calendarização de tarefas automáticas a realizar (como gerar relatórios ou alertas).* 9. *O sistema deve permitir a correção e classificação automática de questionários teóricos e atribuição de uma nota de 0 a 20;* 10. *O sistema deve permitir correr ferramentas de deteção de plágio na submissão de trabalhos escritos. Poderá também permitir a deteção de plagio em código submetido pelos utilizadores/alunos;* 11. *O sistema deve permitir integração com ferramentas exteriores como GIT, ferramentas de análise estática de código, outras ferramentas relevantes;* 12. *O sistema deve permitir pequenas interações/funcionalidades que promovam o conceito gamification e que possam despertar interesse aos alunos (barra de status, nível de utilizador, pontos por cada exercício realizado, badges);* 13. *O sistema deve permitir deixar feedback em todas as atividades;* 14. *O sistema deve possibilitar gerar relatórios de utilização com aspetos relevantes de utilização da plataforma por parte dos utilizadores/alunos como exercícios mais resolvidos, entre outros, para auxiliar a disponibilização de novo conteúdo e funcionalidades a ser inseridas na plataforma.* 15. *O sistema deve possibilitar o contexto de sessão e login para cada utilizador e também a configuração de funções para cada utilizador (professor, aluno, admin, etc);* 16. *O sistema deve disponibilizar um IDE online com a possibilidade de compilar e correr código de determinada linguagem;* 17. *O sistema deve estar alojado online de forma a estar disponível através da internet;*
Requisitos não funcionais	18. O processo de criação do curso, deve ser o mais rápido e intuitivo possível; 19. O serviço deve ter uma interface de fácil utilização, integrar ou ser parte de uma plataforma de *LMS'* e ser modelar tendo a possibilidade de acrescentar mais funcionalidades/módulos/plugins.

Atendendo aos requisitos pretendidos, presentes na Tabela 7, foi possível também elaborar os diagramas de casos de uso para esclarecer os objetivos e interações da plataforma entre o professor e o aluno. A Figura 12, mostra o diagrama de casos de uso geral da plataforma e as interações que poderão ser realizadas por alunos e professores.

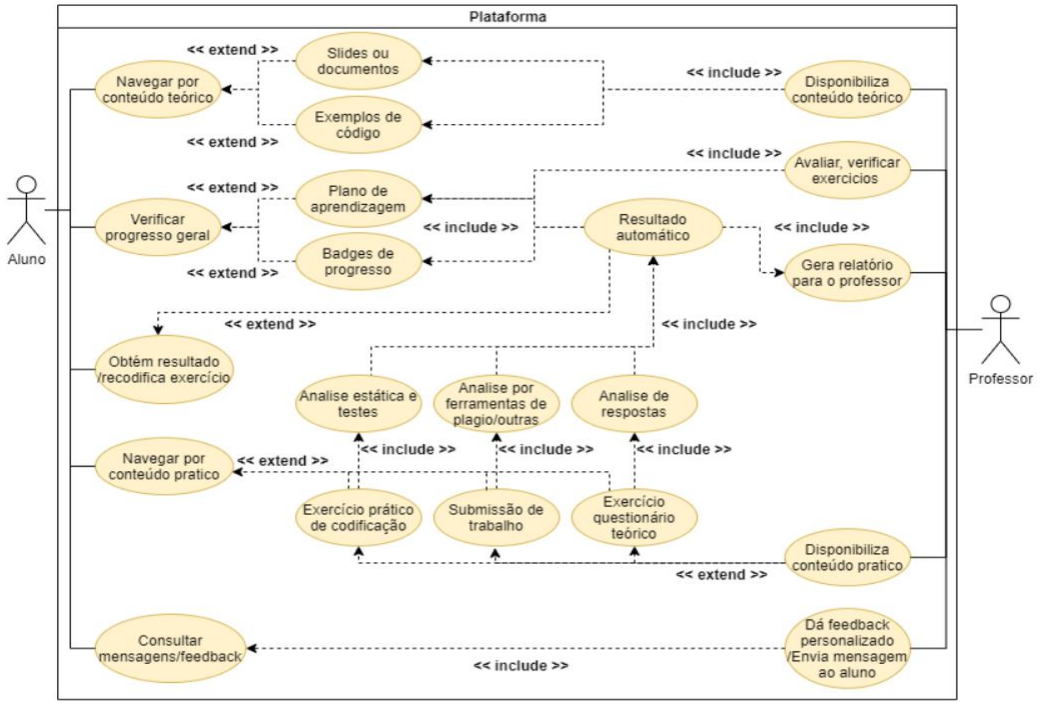


Figura 12 - Casos de uso da plataforma

As Figura 13, Figura 14 e Figura 15 mostram mais especificamente os diagramas de casos de uso das interações de resolução de um exercício prático, submissão de um trabalho escrito e resolução de um questionário, respetivamente. Estas serão as interações principais por parte dos utilizadores/alunos.

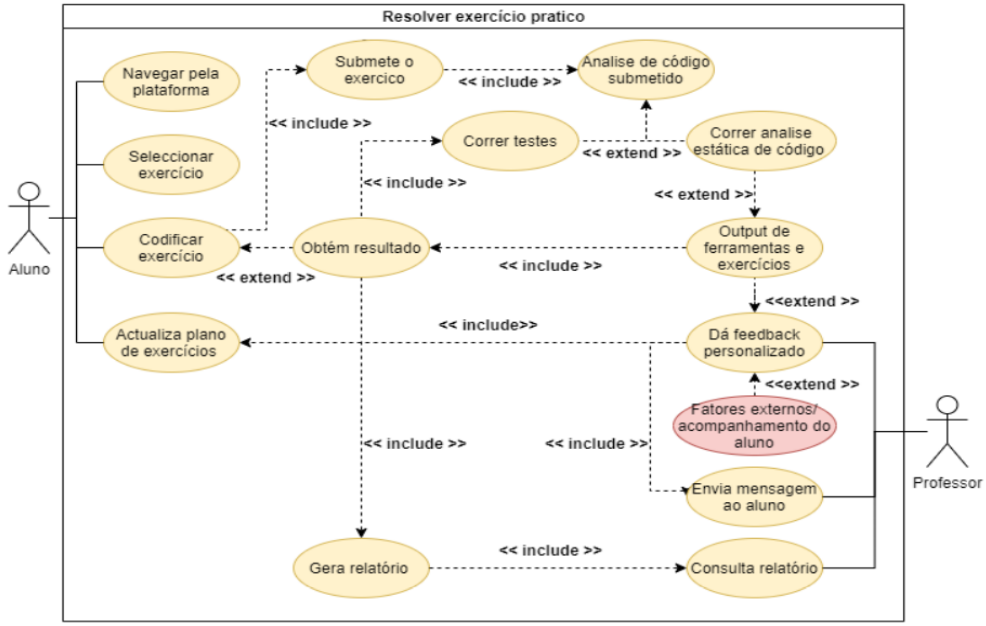


Figura 13 - Caso de uso 'Resolver exercício prático'

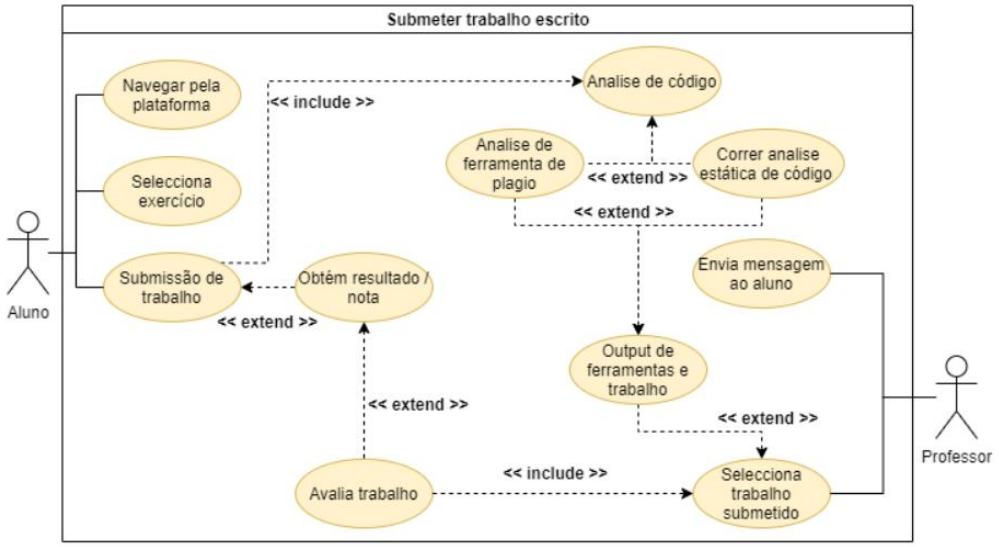


Figura 14 - Caso de uso 'Submeter trabalho escrito'

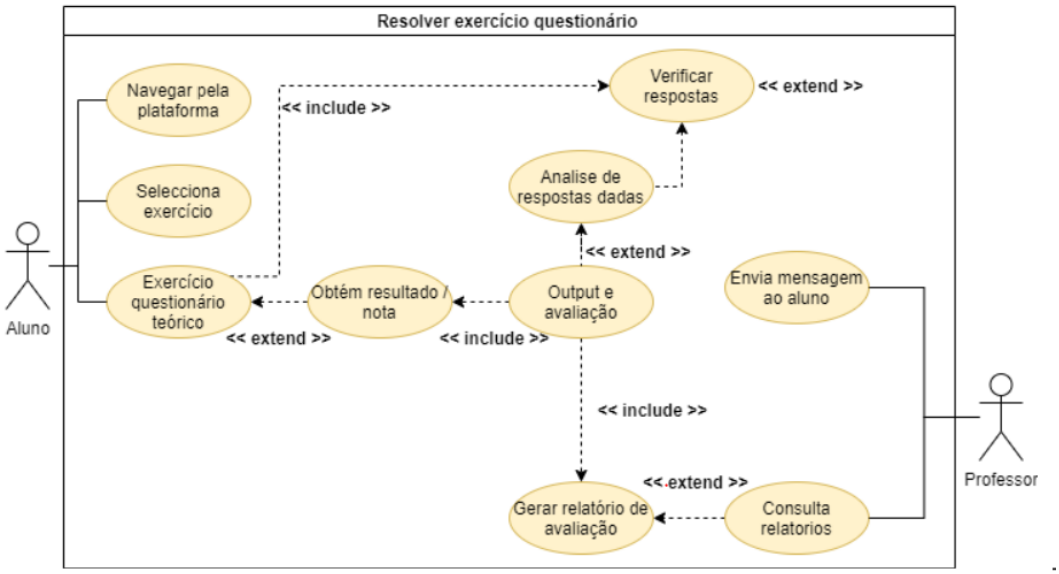


Figura 15- Caso de uso 'Resolver questionário'

4.4 Escolha da plataforma de ajuda à programação

Após todas as informações e requisitos levantados anteriormente, foram iniciadas pesquisas no sentido de identificar eventuais plataformas que possam satisfazer as necessidades elencadas no capítulo anterior.

Foram tomadas em conta plataformas e ferramentas que têm como objetivo o treino e aprendizagem de programação, que sejam *open-source* e que possam englobar as funcionalidades pretendidas.

Apesar do objetivo deste projeto não ser o desenvolvimento de funcionalidades ou componentes para a plataforma, mas a interligação de funcionalidades já existentes em plataformas modulares, que possibilitem o desenvolvimento de novos componentes será valorizada.

Foram identificadas as seguintes plataformas:

4.4.1 Codeboard

O *codeboard* é um *IDE web* para resolução de exercícios em linguagens de programação como o *C, C++, Eiffel, Haskell, Java, Python*. Permite a criação de projetos (exercícios de código) no qual podem existir alunos. A conta que cria o projeto é considerada como professor, sendo as contas convidadas consideradas como alunos, que podem resolver os exercícios e código disponibilizado. O aluno poderá realizar submissões sobre as quais são corridos testes para determinar a taxa de acerto do código submetido. As Figura 16 e Figura 17 ilustram a interface do c*odeboard*. Encontra-

se disponível mais informação sobre esta ferramenta em https://codeboard.io/ e em https://github.com/codeboardio .

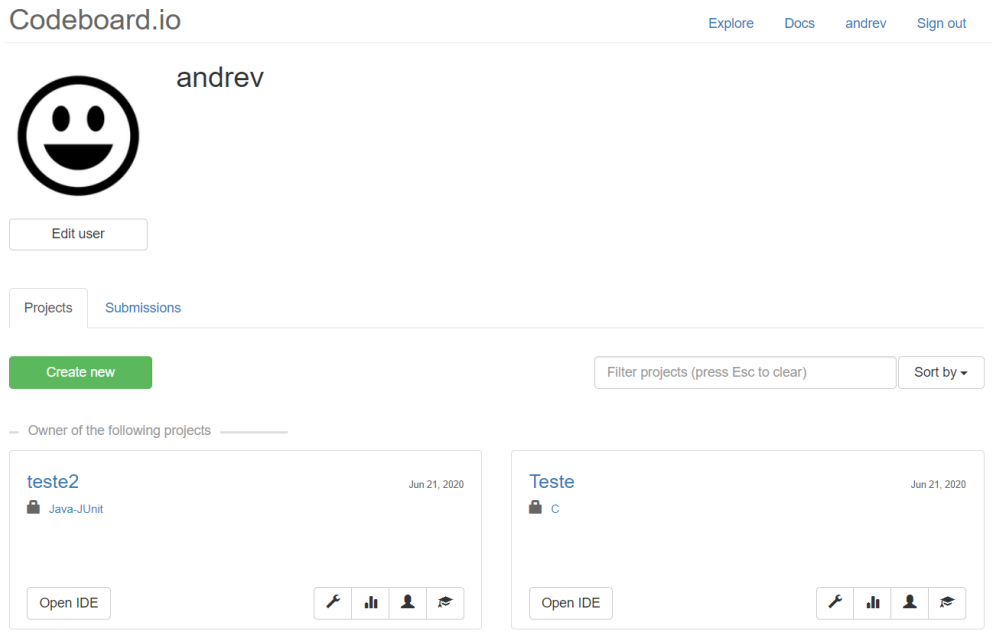


Figura 16 - Interface do Codeboard.io

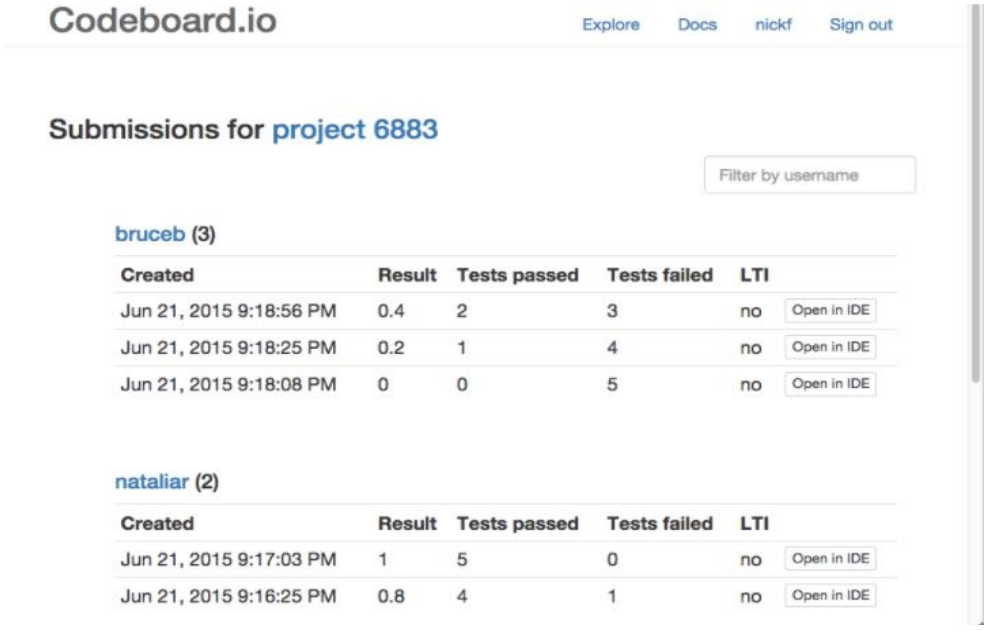


Figura 17 - Gestão de projetos e submissões no codeboard.io

4.4.2 Classroom Github

O *classroom GitHub* funciona de maneira similar ao *GitHub* na criação de código, projetos, na sua partilha e colaboração. O *classroom* permite depois atribuir *assignments* a determinados utilizadores e correr testes sobre os seus *commits* e *merge requests*, permitindo deixar feedback através dos comentários aos *commits*.
É possível integrar sistemas *LMS* com o *classroom GitHub*, no entanto apenas permite lançar a plataforma, sendo necessária uma conta adicional para o utilizar. Não é possível partilhar recursos ou analisá-los.

As Figura 18 e Figura 19 ilustram a interface do *classroom GitHub.*

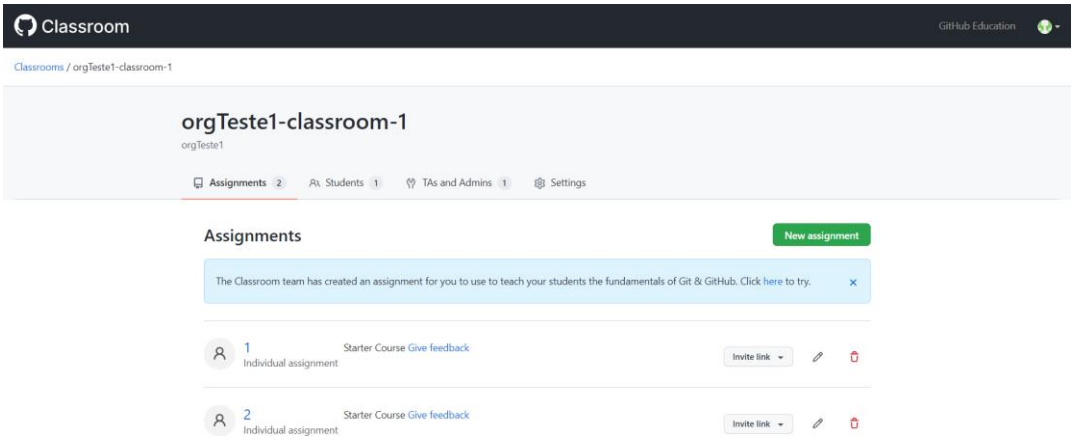


Figura 18 - Interface do *Classroom*

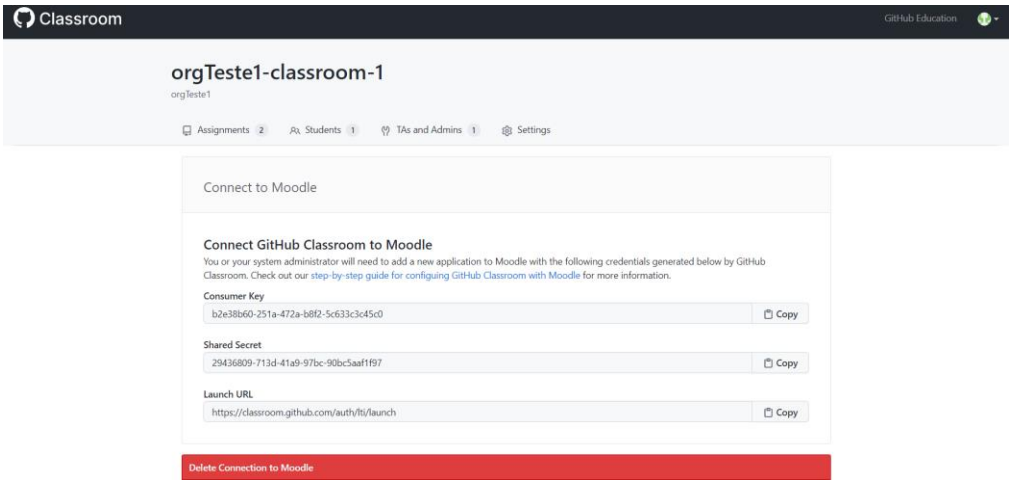


Figura 19- Interligar *Classroom* a um sistema *LMS* (ex: *Moodle*)

4.4.3 Moodle

O *Moodle* é a plataforma *LMS* usada pelo ISEC ao longo dos últimos anos, permitindo que os utilizadores, professor ou aluno, desempenhem funções como a criação de disciplinas para disponibilizar conteúdo teórico, prático ou a criação de atividades para submissão de trabalhos.
Esta plataforma desenvolvida em PHP, é modular e permite a inclusão de funcionalidades adicionais através de plugins, permitindo inclusivamente o desenvolvimento de módulos próprios.
Esta plataforma dispõe já de muitas funcionalidades como o envio de notificações, a criação de questionários, exercícios de escolha múltipla, exercícios de submissão, lições, fórum, calendário, pauta onde é possível configurar avaliação, mensagens entre utilizadores. Existe também uma comunidade ativa que desenvolve funcionalidades adicionais para o Moodle que podem ser utilizadas e configuradas.

4.4.4 Análise final das plataformas

Assim, as plataformas estudadas anteriormente, preenchem genericamente os requisitos assinalados na Tabela 8, sendo a coluna A o *Codeboard*, a B o *Classroom Github* e a C o *Moodle*.

A plataforma *Codeboard* possui alguns pontos positivos, no entanto não cobre todos requisitos definidos. Possui ainda alguns pontos negativos adicionais como:

- O projeto encontra-se em estado *deprecated*, não tendo qualquer atualização desde 2015;
- A plataforma necessita de diversos componentes e as instruções de instalação, de cada um, possuem comandos *outdated* ou instruções que já não funcionam na sua totalidade;
- Não possui suporte direto ou imagem oficial no *DockerHub* ou outro tipo container/microserviço;
- Os componentes aparentam estar num estado beta de desenvolvimento, tendo perdido a comunidade e a equipa de desenvolvimento ao longo dos últimos anos.

A plataforma *Classroom Github* também não cobre todos requisitos definidos. Possui alguns pontos negativos adicionais como:

- A conta de utilizador do sistema *LMS* e da plataforma *classroom GitHub* são distintas, mantendo assim contextos diferentes (exercícios, resultados, entre outros recursos) em cada plataforma;
- Os *assignments* e notas obtidas não são transmissíveis entre plataformas, sendo por exemplo impossível sincronizar resultados e pautas de notas;
- A plataforma possui limitações na criação/disponibilização de recursos ou exercícios de escolha múltipla, submissão e/ou outros.

Das ferramentas exploradas, o *Moodle* foi a que melhor correspondeu aos requisitos especificados. Sendo também uma plataforma mais versátil e modelável, sobre a qual alunos e professores do ISEC já usam e possuem conhecimento, ficando mais fácil sugerir alterações ou melhorar tarefas já realizadas.

Assim, foi tomada a decisão de partir do *Moodle* e construir/melhorar a plataforma criando uma disciplina que pudesse corresponder ao ciclo e processo anteriormente especificado.

Existem também diversas questões adicionais que já se encontram resolvidas como a gestão de sessões, perfis e contas de utilizador para alunos e professores, e o facto de ser uma plataforma *LMS* modelar com possibilidade de desenvolvimento e inclusão de novas funcionalidades.

Tabela 8 - Pré-requisitos preenchidos pelo *Codeboard*, *Classroom Github* e *Moodle*

		A	B	C
Requisitos funcionais	1. O sistema deve permitir a disponibilização de conteúdo teórico;			X
	2. O sistema deve permitir a disponibilização de conteúdo pratico/atividades a ser resolvidas e submetidas por alunos/utilizadores da plataforma;	X	X	X
	3. O sistema deve permitir que o professor configure 3 tipos de atividades: questionários teóricos, exercícios práticos codificados e compilados no browser, submissão de trabalhos escritos;			X
	4. O sistema deve permitir a troca de mensagens entre professores e alunos com o intuito de pedir ajuda para os exercícios e estabelecer comunicação pedagógica de aprendizagem;		X	X
	5. O sistema deve permitir definir testes automáticos para cada exercício pratico para assim validar o output submetido pelos utilizadores/alunos. Poderá também permitir que os utilizadores/alunos criem os seus próprios testes para os exercícios que codificam;	X	X	X
	6. O sistema deve permitir análise estática de código submetido pelos alunos;			
	7. O sistema deve permitir a elaboração de relatórios do percurso de cada aluno (exercícios realizados, número de submissões necessários, tempo despendido, testes passados e falhados, logica de código);			
	8. O sistema deve permitir a calendarização de tarefas automáticas a realizar (como gerar relatórios ou alertas).			
	9. O sistema deve permitir a correção e classificação automática de questionários teóricos e atribuir uma nota de 0 a 20;			X
	10. O sistema deve permitir correr ferramentas de deteção de plagio na submissão de trabalhos escritos. Poderá também permitir a deteção de plagio em código submetido pelos utilizadores/alunos;			X
	11. O sistema deve permitir integração com ferramentas exteriores como GIT, ferramentas de análise estática de código, outras ferramentas relevantes;		X	
	12. O sistema deve permitir pequenas interações/funcionalidades que promovam o conceito *gamification* e que possam despertar interesse aos alunos (barra de status, nível de utilizador, pontos por cada exercício realizado, badges);			X
	13. O sistema deve permitir deixar feedback em todas as atividades;	X	X	X
	14. O sistema deve possibilitar gerar relatórios de utilização com aspetos relevantes de utilização da plataforma por parte dos utilizadores/alunos como exercícios mais resolvidos, entre outros, para auxiliar a disponibilização de novo conteúdo e funcionalidades a ser inseridas na plataforma.			
	15. O sistema deve possibilitar o contexto de sessão e login para cada utilizador e também a configuração de funções para cada utilizador (professor, aluno, *admin*, etc);	X	X	X
	16. O sistema deve disponibilizar um IDE online com a possibilidade de compilar e correr código de determinada linguagem;	X	X	X
	17. O sistema deve estar alojado online de forma a estar disponível através da internet;	X	X	X
Requisitos não funcionais	18. O processo de criação do curso, deve ser o mais rápido e intuitivo possível;			
	19. O serviço deve ter uma interface de fácil utilização, integrar ou ser parte de uma plataforma de LMS e ser modelar tendo a possibilidade de acrescentar mais funcionalidades/módulos/plugins.	X	X	X

5 Implementação e testes da plataforma LMSOps

Neste capítulo, são apresentadas as funcionalidades planeadas para a plataforma assim como as fases de experimentação que decorreram.

5.1 Plano genérico de funcionalidades

De acordo com o ciclo de aprendizagem *DevOps,* especificado anteriormente, foi criado um conjunto de tarefas relevantes que adicionam funcionalidades à plataforma em termos de operacionalidade ou funcionalidade para professores e alunos. Estas tarefas foram divididas em 3 versões de elementos a desenvolver/configurar de maneira evolutiva durante a experimentação da plataforma.

5.1.1 Versão 1

Esta versão pretende:

- Criar um plano de estudo com competências que correspondem aos diversos tópicos de uma disciplina;
- Criar exercícios codificados que possam validar as resoluções submetidas, retornar feedback e atribuir uma nota;
- Permitir aos professores receber notificação das resoluções e rever as notas;
- Criar um sistema que possa atribuir pontos e níveis a cada aluno mediante a sua utilização da plataforma;
- Criar um relatório de resolução do exercício para o professor;
- Criar uma pauta que possa conter um resumo de todas as resoluções e notas;
- Criar um pipeline que possa automatizar ao máximo todos as atividades realizadas no moodle.

5.1.2 Versão 2

Esta versão pretende criar/aprofundar automatismos para os passos criados na versão anterior, com incidência na:

- Criação de disciplina;
- Criação de tópicos de aprendizagem, e disponibilização de conteúdos;

- Criação de pauta e atribuição de pesos;
- Geração de relatórios das resoluções de exercícios submetidas;
- Mecanismos adicionais que possam ajudar nos fluxos pretendidos;
- Melhorar a interpretação de exercícios, investigando a possibilidade de utilização de ferramentas de análise estática de código para validarem o código submetido.

5.1.3 Versão 3

Esta versão pretender investigar e melhorar pequenos aspetos como:

- Melhorar a informação disponível, relativa aos alunos e a resolução de exercícios, através de consultas a BD ou outros mecanismos;
- Individualização das abordagens, tentativa de refinamento do ciclo criado e das ferramentas e passos criados.

A metodologia de ensino-aprendizagem de programação desenhada e a plataforma escolhida para o projeto foram abordadas numa reunião introdutória com o professor de Programação da LEI-ISEC. O objetivo será usar a plataforma como auxiliar no ensino-aprendizagem dessa unidade curricular.

Foram definidos os seguintes pontos:

1. O público-alvo seria constituído pelos alunos que se encontravam a realizar a unidade curricular de Programação. Assim, a experiência visou ajudar na aquisição de conhecimentos na linguagem de programação em C;
2. Seria pretendido ter uma amostra de cerca de 30/40 alunos a participar na plataforma;
3. Seriam disponibilizados conteúdos programáticos distintos acompanhando os conteúdos programáticos a serem lecionados na unidade curricular de Programação;
4. Dividir a experiência em fases e melhorar iterativamente a plataforma com base no feedback obtido;
5. Seriam recolhidos feedback e dados de forma anónima.

A plataforma será chamada nas próximas secções de plataforma *LMSOps*, sendo apresentadas de seguida as fases de experimentação que decorreram com a mesma.

5.2 Fase 1

A primeira fase de experimentação da plataforma *LMSOps* decorreu durante 4 semanas e incidiu sobre o tópico de Estruturas de Dados da linguagem de programação C. Foram disponibilizadas 30 contas no *Moodle*, que suporta esta plataforma, aos alunos de programação que se inscreveram para a experiência. A plataforma ficara alojada nos servidores do ISEC sendo acessível por VPN.

5.2.1 Implementação e funcionalidades

O percurso de atividades que os alunos iriam percorrer contava com:

1. Dez exercícios 'Teste/Exercício' de programação com *Code Runner*, com testes para verificar a sua correta resolução e *feedback* adequado;
2. Dois inquéritos, um inicial com perguntas direcionadas ao percurso de programação e conhecimentos técnicos dos alunos e um a realizar no final da experiência sobre os conhecimentos de programação obtidos e a interação com a plataforma no geral;
3. Um exercício teórico com alíneas de escolha múltipla direcionadas à compreensão de pedaços de código e a sua compilação;
4. *O plugin 'Tarefas a realizar'*, elemento de visualização na disciplina que permite verificar o percurso de exercícios a realizar para finalizar a fase;
5. Aba *'Pauta'*, onde é possível verificar as notas obtidas nos exercícios;
6. *O plugin* '*Message my teacher'*, elemento de visualização da disciplina que permite aos utilizadores da plataforma iniciarem um contacto direto com o professor da disciplina através do sistema de mensagens direto da plataforma;
7. *O plugin* '*Level'*, elemento de visualização da disciplina que atribui um nível a cada utilizador mediante os pontos que são ganhos à medida que resolve exercícios. Este elemento enquadra-se no âmbito de utilizar *gamification* na resolução de exercícios, atribuindo pontos a cada exercício resolvido e níveis mediantes o total de pontos que cada utilizador dispõem.

A Figura 20, ilustra a plataforma para esta fase.

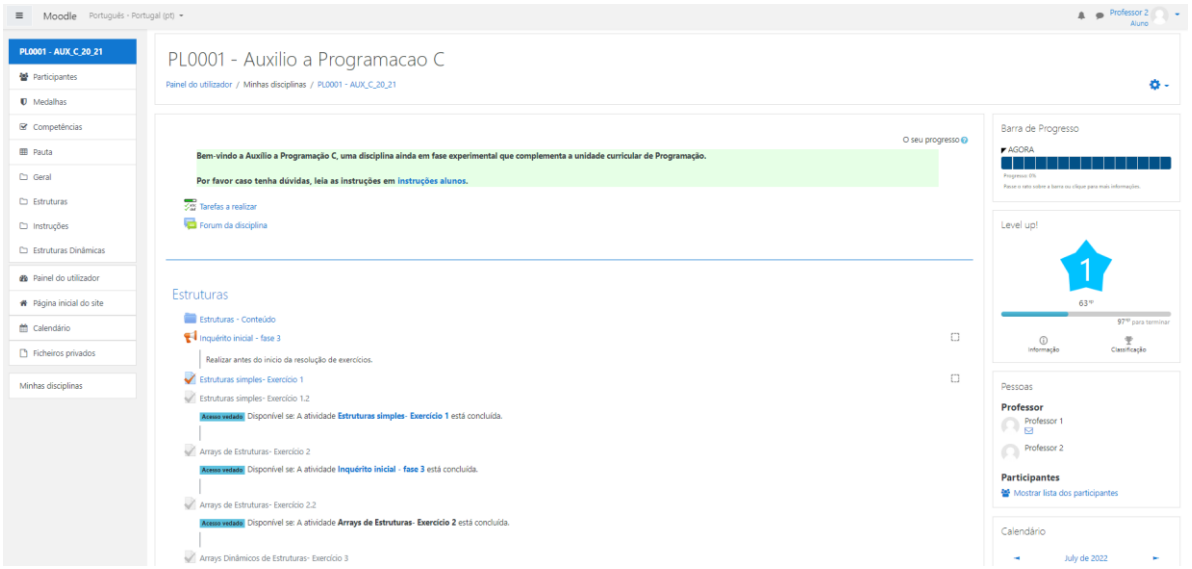


Figura 20 – *Overview* da disciplina fase 1

É possível verificar as instruções de uso e todas as funcionalidades disponibilizadas no anexo D.

De acordo com as versões definidas no capítulo 5.1.1.1, 5.1.1.2 e 5.1.1.3, foram apenas desenvolvidas funcionalidades e recursos da versão 1. Assim, atendendo a uma escala crescente de: não contemplado, parcialmente contemplado ou contemplado - o mapeamento das fases da pipeline definidas na plataforma, as suas funcionalidade e recursos para esta fase são as seguintes:

1. Fase *Plan*:
    1.1. O professor define os objetivos de aprendizagem por tema e/ou semanais – **contemplado**
        - Cria disciplina, faz upload do conteúdo por tópicos de aprendizagem, cria ou importa as perguntas codificadas para a base de dados de perguntas da disciplina.
        - Define pauta e pesos das questões e restantes parâmetros.
        - Dá destaque ao tópico/conteúdos programáticos e aos exercícios que pretender.

        A Figura 21, ilustra a definição de exercícios e tópicos dentro da disciplina.

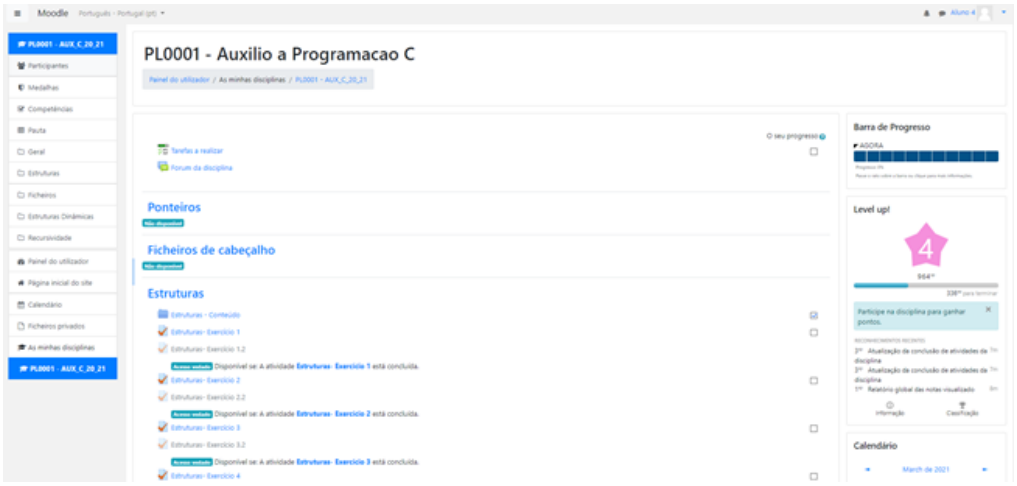


Figura 21 - Disciplina e exercícios

A Figura 22, ilustra a definição de pesos para cada exercício, utilizando a pauta da disciplina.

Figura 22 - Definição do peso de cada exercício na disciplina

1.2. O professor identifica os elementos de estudo para cada tema e/ou semana – **parcialmente contemplado**

- Pode definir questionários teóricos ou definir conteúdos a ver através do fórum, outros avisos ou mensagens do *Moodle*. O plugin 'Tarefas a realizar' cumpre necessariamente a finalidade de lista de tarefas.

A Figura 23, ilustra a edição da lista de tarefas a realizar no plugin 'Tarefas a realizar'.

Figura 23 - Edição da lista de tarefas a realizar no plugin 'Tarefas'

1.3. O professor define as tarefas a cumprir por tema e/ou semana – **parcialmente contemplado**

- É definido globalmente uma barra de progresso com os exercícios que são pretendidos que os alunos realizem. Essa barra pode ir sendo alargada conforme os exercícios são disponibilizados. A barra é geral para todos os alunos inscritos na disciplina.
- Abordagens mais individuais não são ainda suportadas. No entanto é possível através de grupos disponibilizar mais exercícios aos alunos que tenham dificuldades em cada tópico.
- Data de término nos exercícios.

A Figura 24, ilustra a visão dos exercícios resolvidos pelos alunos assim como o percurso de atividades de cada um deles.

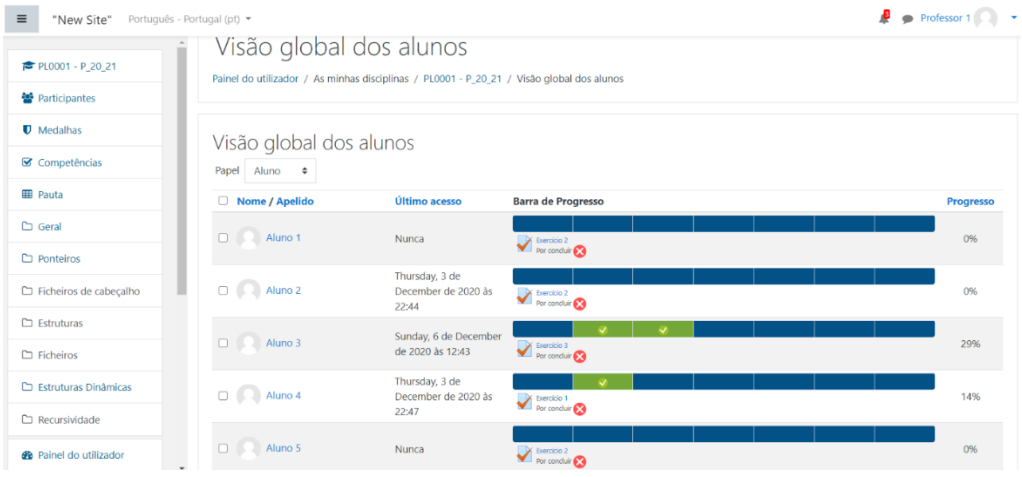


Figura 24 - Lista de tarefas a cumprir no plugin 'Barra de Progresso'

2. Fase *Code*:

2.1. O aluno planeia o seu estudo para a semana ou para o tema – **contemplado**

- Usa o *plugin* 'Tarefas a realizar' para verificar as novas tarefas a realizar. Normalmente terá os seus tópicos alinhados pelo professor à semana, pelo conteúdo disponibilizado ou pelas datas de término dos exercícios.
- Pode consultar a barra de tarefas e a data de término das tarefas a realizar. Desbloqueia novos exercícios pela resolução de exercícios anteriores.

A Figura 25, ilustra a visão do aluno da 'Barra de Progresso', mostrando as tarefas que tem para realizar.

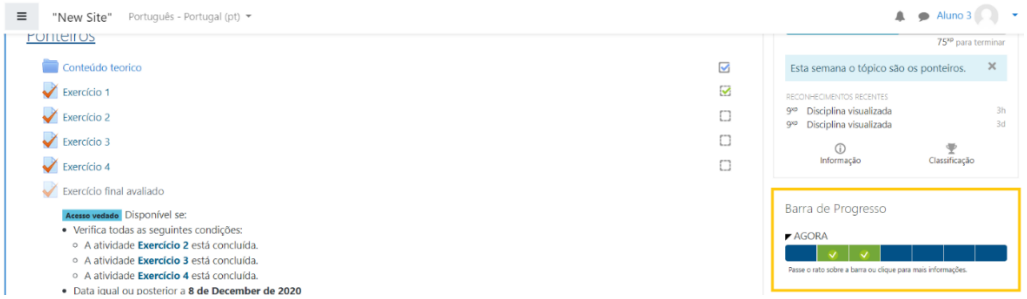


Figura 25 - Plugin 'Barra de Progresso'

A Figura 26, ilustra a visão do aluno do 'Tarefas a realizar', mostrando as tarefas que tem para realizar.

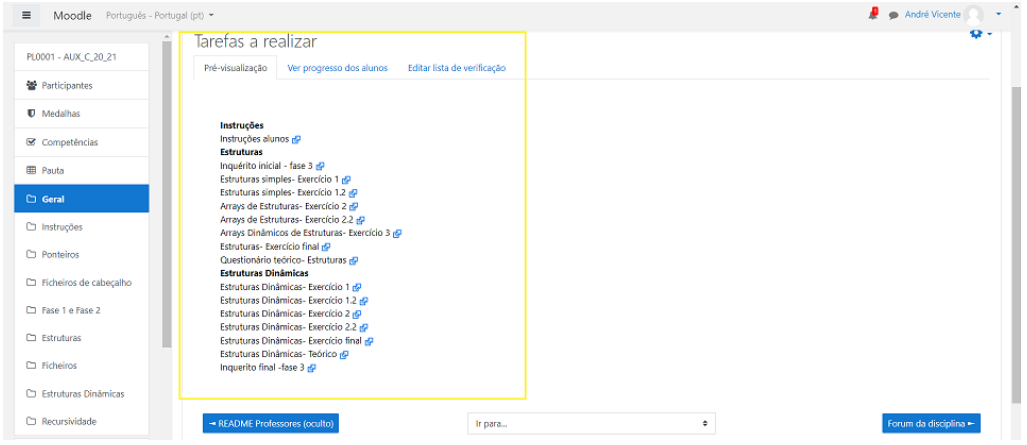


Figura 26 - Plugin 'Tarefas a realizar'

3. Fase *Build and Test* (as fases são executadas em simultâneo em tarefas comuns):
   3.1. O aluno estuda elementos – **contemplado**
   - Visualiza e estuda os elementos teóricos disponibilizados na disciplina e vê e realiza exercícios codificados, questionários ou outros recursos.

   3.2. O aluno realiza exercícios – **contemplado**
   - As questões implementadas são do tipo 'Teste/Exercício' com *Code Runner* e questionários.

   A Figura 27, ilustra a interface de um exercício para o aluno.

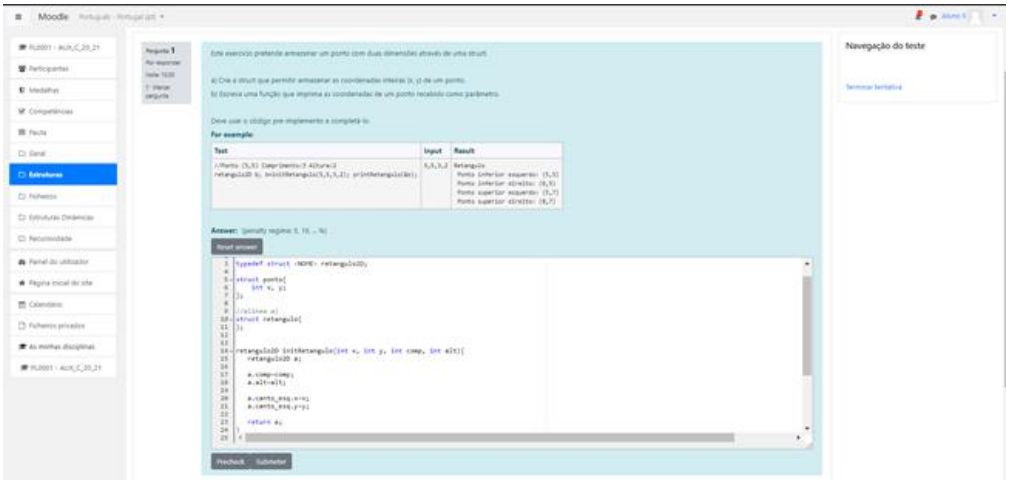


Figura 27 - Visão exercício

3.3. O aluno visualiza os elementos de estudo (conteúdo teórico ou prático) e resolve os exercícios – **contemplado**

- Conteúdo teórico.
- Conteúdo prático, os exercícios são criados através do recurso 'Teste/Exercício', que faz registo do progresso e submissões feitas em cada exercício;

3.4. O aluno tem feedback automático (ex: testes, análise estática de código) – **parcialmente contemplado**

- Após realizar os exercícios é dado feedback sobre a resolução submetida através da indicação de exercícios ou conteúdo teórico semelhante presente nos slides teóricos e do objetivo do exercício. Mediante a nota alcançada no exercício o feedback também aconselha alguns passos como repetir o exercício ou visualizar novamente o conteúdo teórico;
- É apenas validado o input e output do programa, não existindo análise estática de código. No entanto, o aluno pode ser forçado a usar certar estruturas ou variáveis através das pré-implementações dos exercícios codificados.

A Figura 28 e Figura 29 ilustram o *feedback* automático que é retornado após submissão de um exercício ao aluno e o *feedback* retornado pelos testes do exercício (que valida o *output* que é submetido pelo aluno) respetivamente.

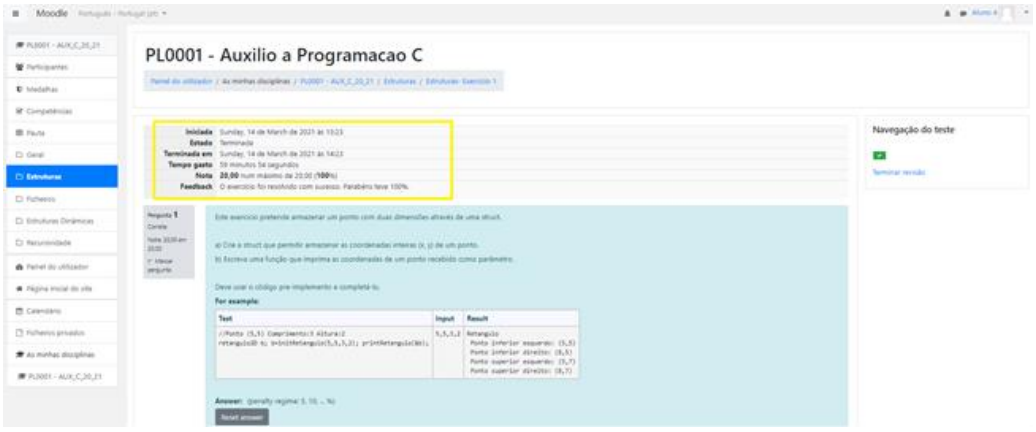


Figura 28 - Output do exercício (nota atribuída e feedback)

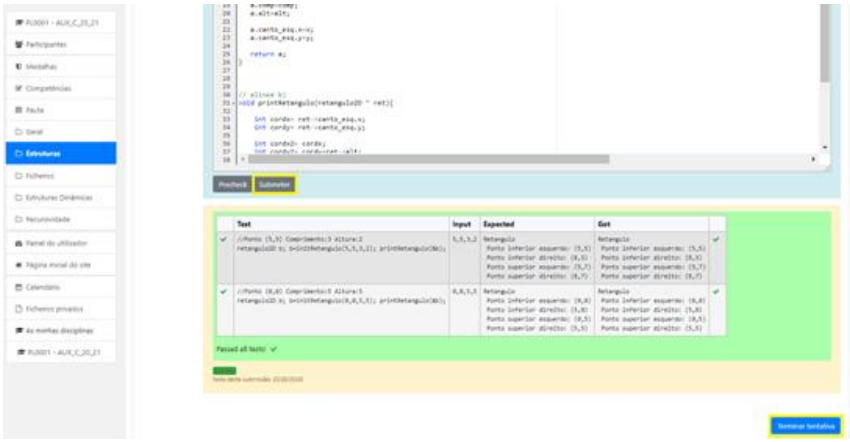


Figura 29 - Testes de um exercício

3.5. O aluno pode pedir ajuda ao professor - **parcialmente contemplado**

- Pode contactar o professor através do sistema de mensagens integrado do Moodle;
- O plugin '*Message my Teacher*' presente na página da disciplina apresenta um *shortcut* para o chat com o(s) professor(es) da disciplina;
- Poderá também enviar email ou participar no fórum da disciplina.

A Figura 30, ilustra a funcionalidade '*Message my Teacher*'.

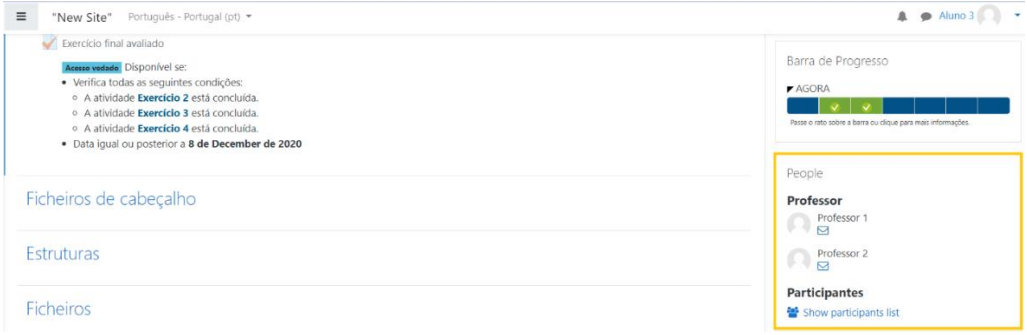


Figura 30 - Plugin *'Message my Theacher'*

4. Fase *Release*:
   4.1. O aluno submete os trabalhos que são dados como concluídos na plataforma - **contemplado**
   - São disponibilizadas questões 'Teste/Exercício' com *Code Runner* e questões 'Teste/Exercício' do tipo questionário. Aos exercícios é atribuída uma nota, automaticamente após a submissão de resolução. e é enviada uma notificação ao professor.

   A Figura 31, ilustra a edição das tentativas e submissão de um exercício.

Figura 31 - Submissão de exercício

   4.2. O aluno pode voltar ao ponto '3. Fase Build e Test/Estudo', até terminar todas as tarefas definidas pelo docente - **contemplado**
   - Vários exercícios disponíveis em cada tópico/unidade programática.
   - O tempo despendido e número de tentativas do aluno em cada exercício é registado na submissão do teste/exercício.

5. Fase *Deploy and Operate* (as fases são executadas em simultâneo em tarefas comuns):
   5.1. O professor analisa os relatórios disponíveis na plataforma sobre a sua utilização e os resultados dos exercícios - **contemplado**
   - É recebida notificação após cada aluno finalizar a resolução de um exercício.
   - Pode visualizar relatórios de submissão/resolução no teste/exercício.

   A Figura 32, ilustra as notificações.

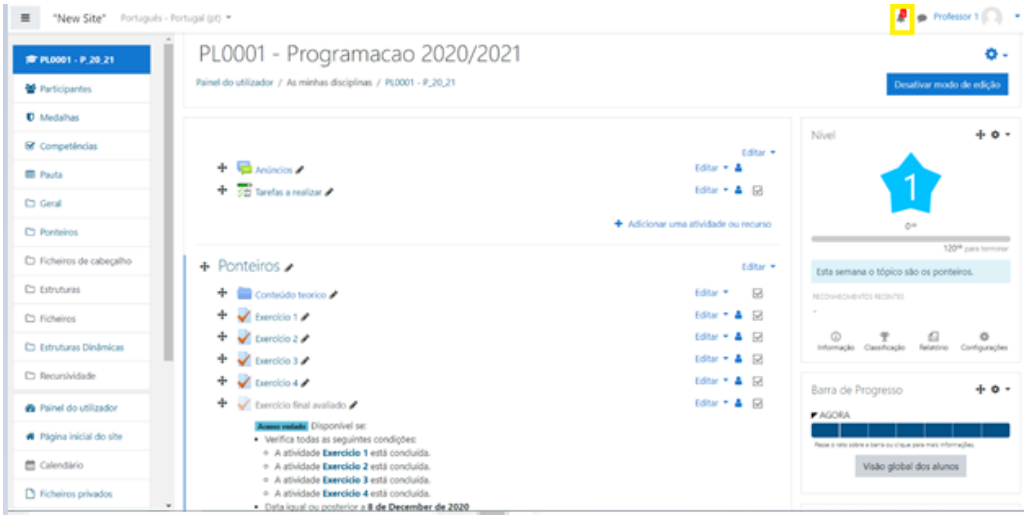


Figura 32 – Notificações

5.2. O professor analisa manualmente os exercícios submetidos - **contemplado**

- Através do link recebido na notificação é possível ir diretamente para a resolução do exercício, submetida pelo aluno, e rever a tentativa.

A Figura 33, ilustra a visualização das estatísticas e resoluções submetidas pelos alunos num determinado exercício.

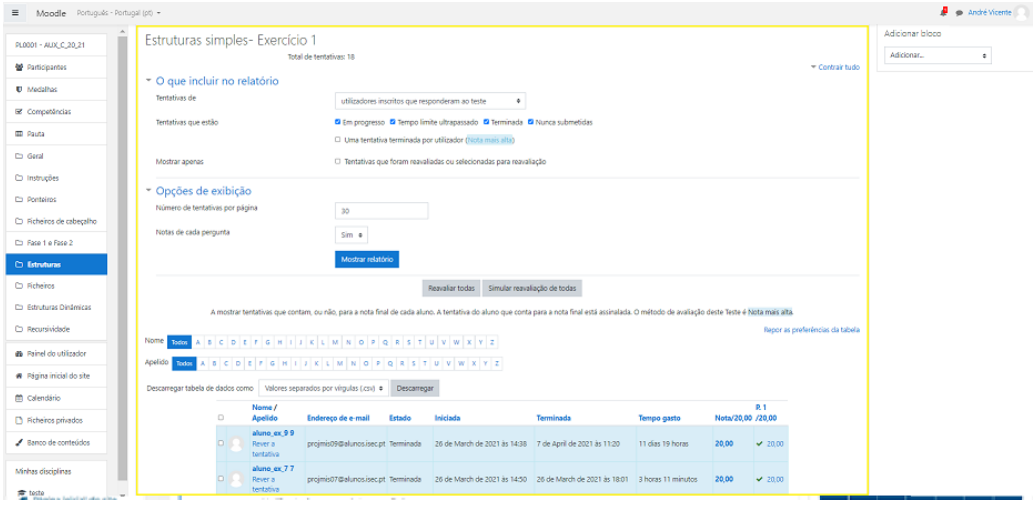


Figura 33 - Visualização das submissões de um exercício

6. Fase *Monitor*:

6.1. O professor dá feedback semiautomático ao aluno, com base nos relatórios da ferra–enta - **parcialmente contemplado**

- É possível colocar comentários nas resoluções submetidas, no entanto, o aluno não recebe notificação desse feedback. Pode também ser utilizado outro recurso para este efeito como o sistema de mensagens.

A Figura 34, ilustra o relatório de avaliador que professor pode obter na pauta.

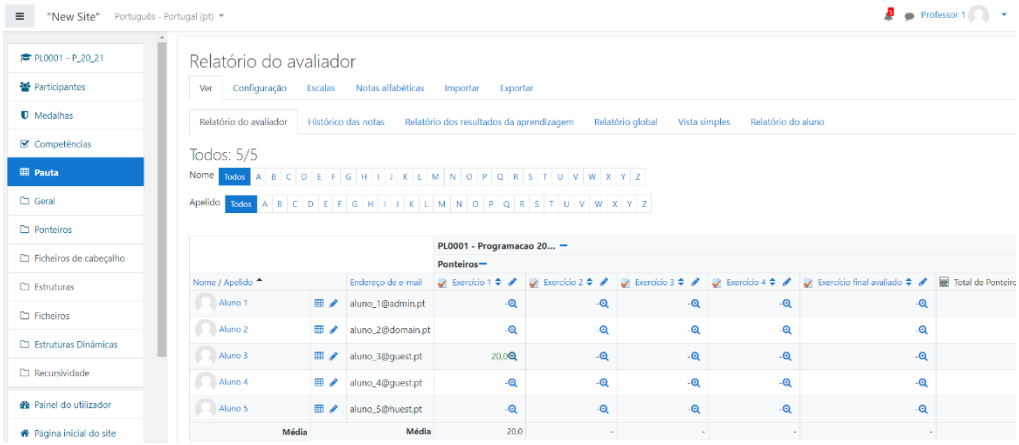


Figura 34 - Relatório avaliador

6.2. O professor dá feedback personalizado ao aluno, ajustando os próximos objetivos de aprendizagem - **parcialmente contemplado**

- Não é possível definir os próximos passos individualmente, no entanto, pode ser individualizado a grupos com base nas dificuldades sentidas pelos alunos. Esses grupos poderão ter acesso a mais exercícios daquele tópico/unidade programática.

5.2.2 Resultados

5.2.2.1 Inquéritos

Através do inquérito inicial, realizado antes de iniciar o percurso de resolução de exercícios, foram obtidos os seguintes resultados:

1. Foram submetidas 17 respostas;
2. 71% destes alunos aprenderam a programar há menos de 4 anos, entre os quais 41% afirmam gostar de programar e 18% não gostam, enquanto os restantes 41% são indiferentes à programação;
3. 88% destes alunos já realizaram a unidade curricular de Introdução à Programação e 70% frequentam e/ou submeteram-se pela primeira vez a avaliação à unidade curricular de Programação;
4. Os alunos consideram que as suas principais dificuldades estão em compreender os problemas/exercícios (23% das repostas), desenvolver o algoritmo necessário (19% das repostas), na sintaxe da linguagem (16% das repostas) e em codificar (16% das repostas). Alguns alunos realçam que já têm hábitos de programação em outras linguagens e que, por vezes, é difícil trocar o paradigma para a linguagem de programação C;
5. Os alunos consideram que o que mais os ajuda a estudar programação é a resolução de exercícios (28% das repostas) e a compreensão das bases teóricas (17% das repostas), a utilização de uma ferramenta ou plataforma que, através de feedback e testes, os ajude

na resolução de exercícios práticos (13% das repostas) e a consulta de fóruns de discussão ou tutoriais (13% das repostas). Fatores como o apoio dos professores de programação (12% das repostas) e a explicações dadas nas aulas (11% das repostas) ficam imediatamente a seguir nos fatores que foram considerados como mais relevantes;

6. Os alunos que vão participar no estudo dedicam normalmente entre 30 ou mais minutos (82% das repostas) no estudo de programação e na realização de exercícios de programação (94% das repostas). Sendo que no estudo de programação, normalmente, são maioritariamente despendidos entre 30min-1h (29% das repostas) e 2-3h (29%), e na realização de exercícios de programação 30min-1h (35% das repostas) e 2-3h (29% das repostas);
7. Os alunos frequentam, na sua maioria, mais de 75% das aulas teóricas (70% das repostas) e das aulas laboratoriais (75% das repostas).

É possível verificar as questões e a totalidade dos resultados obtidos no inquérito inicial consultando o anexo A.

O inquérito final foi dividido em duas partes, uma primeira parte realizada no *Moodle* com perguntas direcionadas à plataforma e as suas funcionalidades, e uma segunda parte realizada através de um inquérito no *google forms*, direcionado à aprendizagem do tópico de programação que foi coberto nesta fase.

É possível verificar a totalidade dos resultados obtidos na primeira parte do inquérito final, no anexo A. Através dos resultados da primeira parte do inquérito final foram obtidas as seguintes conclusões sobre a plataforma e as suas funcionalidades:

- Foram submetidas 9 respostas ao inquérito;
- Das 9 respostas, 7 estudantes realçaram não ter chegado ao fim (78% das repostas) tendo os restantes (22% das repostas) saltados alguns passos;
- Numa escala de 0-10, oito (89%) das repostas consideram que as explicações e passos apresentados na plataforma têm uma classificação igual ou superior a 6, enquanto que nove (100%) das repostas classifica como 7 ou superior a plataforma, relativamente a ser intuitiva e fácil de seguir, com três (33%) dessas respostas a classificaram com 10/avaliação máxima;
- Das respostas obtidas, 66,66% considera que teve necessidade de pedir ajuda por 1 ou 2 vezes ou que teve necessidade de pedir ajuda mas não chegou a pedir, 3 das respostas (33,33%) não sentiu necessidade de pedir ajuda no que diz respeito a questões técnicas de programação;
- Relativamente às maiores dificuldades, é realçado que as pré-implementações, assim como os enunciados dos exercícios de codificação, podem por vezes ser difíceis de seguir ou confusas. E que as pré-implementações de código podem ainda limitar/atrapalhar a implementação do restante código por terem algumas decisões ou maneiras de programar menos familiares ao utilizador;
- Numa escala de 0-10, oito (89%) das respostas classificou a clareza das explicações com 5 ou mais;

- Relativamente à utilização do *Code Runner* nos exercícios de programação e numa escala de 0-10, nove (100%) respostas classificaram com 7 ou mais a experiência de utilização do *Code Runner*, oito (89%) respostas classificaram com 5 ou mais o feedback obtido nos exercícios;
- Numa escala de 0-10, o feedback obtido nos exercícios teóricos ou diretamente do professor é classificado positivamente com 7 ou mais por nove (100%) das respostas e com oito (89%) respostas a classificarem com 7 ou mais a experiência de utilização da plataforma no que diz respeito à aprendizagem do tópico de Estruturas de Dados.
- A experiência, no geral, é classificada como positiva, apesar dos alunos não terem conseguido realizar todos os passos ou terminar todas as tarefas da plataforma. As interações e atividades são realçadas pelos alunos como interessantes sendo reconhecido que praticar e resolver exercícios é um dos pontos fundamentais para a aprendizagem de programação. São referidos como pontos menos positivos desta fase, compreender as explicações e feedback dos exercícios (29%), compreender os testes e/ou as alíneas dos exercícios (21%) e o uso de VPN para aceder à plataforma (21%).

É possível verificar as questões e a totalidade dos resultados obtidos da segunda parte inquérito final no anexo A. Através destes resultados foram obtidas as seguintes conclusões sobre a aprendizagem do tópico de Estruturas de Dados, que foi coberto nesta fase:

- Foram submetidas 6 respostas ao inquérito;
- A amostra de dados disponível é pequena;
- A generalidade das respostas indica que a experiência foi benéfica para adquirir ou melhorar os conhecimentos do tópico Estruturas de Dados na linguagem de programação C. Há, no entanto, uma opinião que realça que a qualidade das atividades (exercícios, explicações, testes e enunciados) poderia ser mais refinada e, que por vezes, contêm bugs ou implementações menos corretas.

Foi também levantado o ponto de vista do professor de Programação, que acompanhou esta fase, relativamente à sua interação com a plataforma e às perguntas/dúvidas colocadas pelos alunos. A plataforma contou com o acesso do professor durante um ou dois dias por semana. Este professor realçou, após o término desta fase, os seguintes aspetos:

- Os alunos que pediram ajuda normalmente não conseguiam identificar qual o exercício em que estavam a ter dificuldades;
- Existiram algumas dificuldades ou dúvidas relacionadas com o *Code Runner* nomeadamente nos exercícios com testes que contêm *prints*, uma vez que um espaço, *Enter* ou um simples caracter errado poderia classificar o teste como errado;

- É difícil visualizar as estatísticas e classificações dos exercícios como um todo, apenas é possível verificar os exercícios individualmente o que também dificulta o acompanhamento do progresso dos alunos;
- É difícil verificar a participação dos alunos e distinguir quem participa e não participa;
- A sintaxe da linguagem de programação e familiarização com a mesma é um aspeto que é melhorado através do treino de resolução de exercícios;
- Os hábitos de programação em outras linguagens de programação podem não ajudar inicialmente nas implementações e familiarização com a linguagem de programação C, no entanto, a aprendizagem de uma linguagem de programação é aprofundada através da resolução de exercícios.

5.2.2.2 Estatísticas de participação e desempenho dos alunos

5.2.2.2.1 Participação

Relativamente à participação dos alunos, e atendendo às 30 contas distribuídas pelos participantes, é possível verificar que:

- Doze alunos nunca acederam à disciplina. Esses utilizadores são apresentados na Figura 35;
- Os restantes 18 alunos acederam pelo menos 1 vez à disciplina, podendo, no entanto, ter ou não realizado exercícios.

		Alunos em risco
Email	Username	Ultimo Acesso à disciplina
aluno_ex_11@example.pt	aluno_ex_11	Nunca acedeu
aluno_ex_12@example.pt	aluno_ex_12	Nunca acedeu
aluno_ex_13@example.pt	aluno_ex_13	Nunca acedeu
aluno_ex_18@example.pt	aluno_ex_18	Nunca acedeu
aluno_ex_2@example.pt	aluno_ex_2	Nunca acedeu
aluno_ex_22@example.pt	aluno_ex_22	Nunca acedeu
aluno_ex_23@example.pt	aluno_ex_23	Nunca acedeu
aluno_ex_24@example.pt	aluno_ex_24	Nunca acedeu
aluno_ex_26@example.pt	aluno_ex_26	Nunca acedeu
aluno_ex_27@example.pt	aluno_ex_27	Nunca acedeu
aluno_ex_28@example.pt	aluno_ex_28	Nunca acedeu
aluno_ex_30@example.pt	aluno_ex_30	Nunca acedeu

Figura 35 - Alunos que nunca acederam à disciplina

Dos dados relativos à participação dos utilizadores nos exercícios (10 exercícios *Code Runner* e um exercício teórico):

- Apenas 9 alunos realizaram 1 ou mais exercícios;

- Destes 9 alunos, 6 (66,67%) realizaram mais exercícios dentro do prazo que tinha sido definido do que após o prazo. Os restantes 33,33% dos alunos realizou mais exercícios fora da data espectável de resolução do que até à data prevista;
- Apenas 1 utilizador desses nove chegou a 100% de resolução dos exercícios previstos, contando, com as tentativas que foram submetidas e as que estão em progresso, outros dois utilizadores realizaram mais de 50% dos exercícios.

A Figura 36, apresenta um resumo das conclusões acima.

Detalhes de exercícios feitos

email	Exercícios feitos a tempo	Exercícios em que passou a data	Total de tentativas	Numero de Quiz
aluno_ex_10@example.pt	1	0	1	11
aluno_ex_14@example.pt	3	4	7	11
aluno_ex_15@example.pt	10	1	11	11
aluno_ex_16@example.pt	2	0	2	11
aluno_ex_1@example.pt	2	0	2	11
aluno_ex_3@example.pt	5	3	8	11
aluno_ex_7@example.pt	7	1	8	11
aluno_ex_8@example.pt	0	1	1	11
aluno_ex_9@example.pt	1	1	2	11

Figura 36 - Estatísticas de resolução de exercícios

5.2.2.2.2 Desempenho

Relativamente à resolução dos exercícios, notas obtidas e submissões realizadas nos exercícios pelos alunos, observou-se que:

- A participação nos exercícios foi diminuindo ao longo do tempo. Os alunos realizaram mais tentativas (submetidas ou em progresso) nos primeiros exercícios que nos últimos exercícios. No exercício teórico houve uma ligeira subida na participação dos alunos;
- Todos os exercícios têm avaliação positiva (as submissões estão 50% corretas ou mais);
- O exercício final não possui nenhuma submissão com nota atribuída.

A Figura 37, ilustra os resultados identificados acima. Para cada exercício são mostrados o número total de tentativas e o seu estado, a média e a nota máxima e mínima obtidas pelos alunos.

Estatísticas dos exercícios

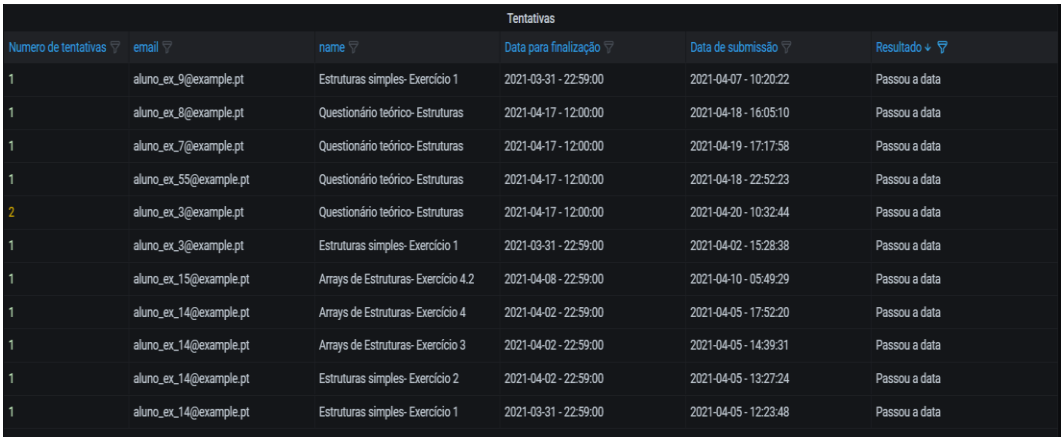
name	Tentativas	Tentativas finalizadas	Tentativas em progresso	Média	Nota máxima	Nota mínima
Estruturas simples- Exercício 1	16	9	7	19.9	20	19
Estruturas simples- Exercício 2	10	5	5	20	20	20
Arrays de Estruturas- Exercício 3	8	5	3	19.8	20	19
Arrays de Estruturas- Exercício 4	5	3	2	18.7	19	18
Estruturas- Exercício final	1	0	1	Sem nota	Sem nota	Sem nota
Questionário teórico- Estruturas	6	6	0	16.3	20	13.3
Arrays Dinâmicos de Estruturas- Exercício 5	2	1	1	18	18	18
Estruturas simples- Exercício 1.2	9	6	3	17.7	20	14
Estruturas simples- Exercício 2.2	4	4	0	18.3	19	17
Arrays de Estruturas- Exercício 3.2	5	3	2	12.3	20	0
Arrays de Estruturas- Exercício 4.2	2	1	1	19	19	19

Figura 37 - Exercícios da fase 1

Relativamente aos dados das submissões dos alunos em todos os exercícios:

- Foram submetidas cerca de quarenta e duas tentativas de resolução (juntando todas submissões);
- Doze dessas tentativas de resolução dos exercícios foram submetidas fora da data prevista de resolução do exercício;
- Trinta dessas tentativas de resolução foram submetidas dentro da data prevista do exercício.

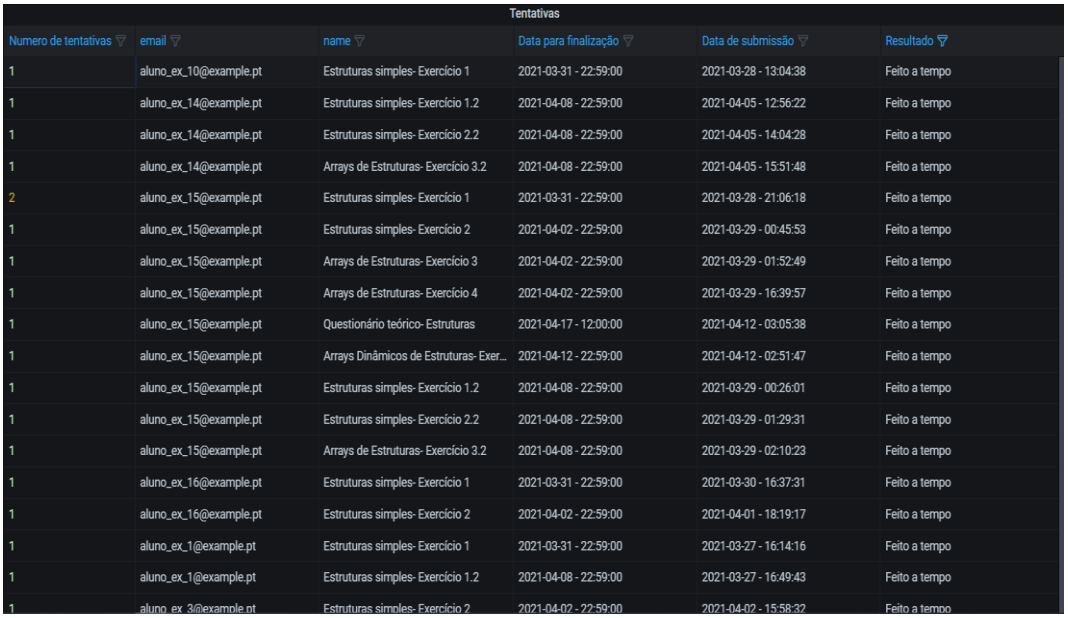
As Figura 38 e Figura 39, ilustram os resultados identificados acima.



Tentativas

Numero de tentativas	email	name	Data para finalização	Data de submissão	Resultado
1	aluno_ex_9@example.pt	Estruturas simples- Exercício 1	2021-03-31 - 22:59:00	2021-04-07 - 10:20:22	Passou a data
1	aluno_ex_8@example.pt	Questionário teórico- Estruturas	2021-04-17 - 12:00:00	2021-04-18 - 16:05:10	Passou a data
1	aluno_ex_7@example.pt	Questionário teórico- Estruturas	2021-04-17 - 12:00:00	2021-04-19 - 17:17:58	Passou a data
1	aluno_ex_55@example.pt	Questionário teórico- Estruturas	2021-04-17 - 12:00:00	2021-04-18 - 22:52:23	Passou a data
2	aluno_ex_3@example.pt	Questionário teórico- Estruturas	2021-04-17 - 12:00:00	2021-04-20 - 10:32:44	Passou a data
1	aluno_ex_3@example.pt	Estruturas simples- Exercício 1	2021-03-31 - 22:59:00	2021-04-02 - 15:28:38	Passou a data
1	aluno_ex_15@example.pt	Arrays de Estruturas- Exercício 4.2	2021-04-08 - 22:59:00	2021-04-10 - 05:49:29	Passou a data
1	aluno_ex_14@example.pt	Arrays de Estruturas- Exercício 4	2021-04-02 - 22:59:00	2021-04-05 - 17:52:20	Passou a data
1	aluno_ex_14@example.pt	Arrays de Estruturas- Exercício 3	2021-04-02 - 22:59:00	2021-04-05 - 14:39:31	Passou a data
1	aluno_ex_14@example.pt	Estruturas simples- Exercício 2	2021-04-02 - 22:59:00	2021-04-05 - 13:27:24	Passou a data
1	aluno_ex_14@example.pt	Estruturas simples- Exercício 1	2021-03-31 - 22:59:00	2021-04-05 - 12:23:48	Passou a data

Figura 38 - Resoluções submetidas dentro do prazo



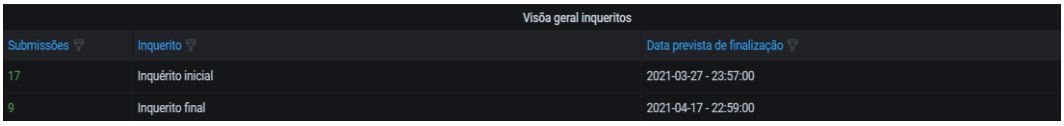
Tentativas

Numero de tentativas	email	name	Data para finalização	Data de submissão	Resultado
1	aluno_ex_10@example.pt	Estruturas simples- Exercício 1	2021-03-31 - 22:59:00	2021-03-28 - 13:04:38	Feito a tempo
1	aluno_ex_14@example.pt	Estruturas simples- Exercício 1.2	2021-04-08 - 22:59:00	2021-04-05 - 12:56:22	Feito a tempo
1	aluno_ex_14@example.pt	Estruturas simples- Exercício 2.2	2021-04-08 - 22:59:00	2021-04-05 - 14:04:28	Feito a tempo
1	aluno_ex_14@example.pt	Arrays de Estruturas- Exercício 3.2	2021-04-08 - 22:59:00	2021-04-05 - 15:51:48	Feito a tempo
2	aluno_ex_15@example.pt	Estruturas simples- Exercício 1	2021-03-31 - 22:59:00	2021-03-28 - 21:06:18	Feito a tempo
1	aluno_ex_15@example.pt	Estruturas simples- Exercício 2	2021-04-02 - 22:59:00	2021-03-29 - 00:45:53	Feito a tempo
1	aluno_ex_15@example.pt	Arrays de Estruturas- Exercício 3	2021-04-02 - 22:59:00	2021-03-29 - 01:52:49	Feito a tempo
1	aluno_ex_15@example.pt	Arrays de Estruturas- Exercício 4	2021-04-02 - 22:59:00	2021-03-29 - 16:39:57	Feito a tempo
1	aluno_ex_15@example.pt	Questionário teórico- Estruturas	2021-04-17 - 12:00:00	2021-04-12 - 03:05:38	Feito a tempo
1	aluno_ex_15@example.pt	Arrays Dinâmicos de Estruturas- Exer...	2021-04-12 - 22:59:00	2021-04-12 - 02:51:47	Feito a tempo
1	aluno_ex_15@example.pt	Estruturas simples- Exercício 1.2	2021-04-08 - 22:59:00	2021-03-29 - 00:26:01	Feito a tempo
1	aluno_ex_15@example.pt	Estruturas simples- Exercício 2.2	2021-04-08 - 22:59:00	2021-03-29 - 01:29:31	Feito a tempo
1	aluno_ex_15@example.pt	Arrays de Estruturas- Exercício 3.2	2021-04-08 - 22:59:00	2021-03-29 - 02:10:23	Feito a tempo
1	aluno_ex_16@example.pt	Estruturas simples- Exercício 1	2021-03-31 - 22:59:00	2021-03-30 - 16:37:31	Feito a tempo
1	aluno_ex_16@example.pt	Estruturas simples- Exercício 2	2021-04-02 - 22:59:00	2021-04-01 - 18:19:17	Feito a tempo
1	aluno_ex_1@example.pt	Estruturas simples- Exercício 1	2021-03-31 - 22:59:00	2021-03-27 - 16:14:16	Feito a tempo
1	aluno_ex_1@example.pt	Estruturas simples- Exercício 1.2	2021-04-08 - 22:59:00	2021-03-27 - 16:49:43	Feito a tempo
1	aluno_ex_3@example.pt	Estruturas simples- Exercício 2	2021-04-02 - 22:59:00	2021-04-02 - 15:58:32	Feito a tempo

Figura 39 - Resoluções submetidas fora do tempo expectável de resolução do exercício

Ainda relativamente à participação dos alunos nos inquéritos e conforme é ilustrado na Figura 40:

- Houve 17 respostas ao inquérito inicial;
- Houve 9 respostas ao inquérito final.



Visão geral inqueritos

Submissões	Inquerito	Data prevista de finalização
17	Inquérito inicial	2021-03-27 - 23:57:00
9	Inquerito final	2021-04-17 - 22:59:00

Figura 40 - Participação nos inquéritos

5.2.3 Conclusões

Com o auxílio dos resultados recolhidos e opiniões dos participantes, podemos concluir que a plataforma cumpriu o pressuposto de ajudar os alunos a praticar exercícios de Estruturas de Dados. Os alunos que a utilizaram destacaram que a resolução de exercícios que valida a resolução por eles submetida é de extremo valor para validar o seu conhecimento e adquirirem prática. A diversidade de exercícios apresentados aparenta também ser uma boa combinação para captar e despertar a atenção dos alunos.

No entanto, a plataforma falhou em diversos pontos, apenas um aluno completou o percurso total de exercícios desta fase tendo os restantes alunos deixado gradualmente de participar na

plataforma. Foram também realçados, pelos alunos, diversos pontos de melhoria como incoerências, falhas nos exercícios, enunciados e respetivos testes, percurso longo e com demasiados exercícios com o mesmo objetivo ou dificuldades de utilização da VPN.

Assim, foi traçado um plano para revisão/adaptação da plataforma e do processo de ajuda à aprendizagem de programação, sendo considerados os seguintes pontos de atuação/melhoria para a segunda fase:

1. Melhorar o acesso e os passos necessários para os alunos acederem à plataforma. Colocar a plataforma disponível num site normal e sem necessidade de utilizar a VPN do ISEC;
2. Melhorar a qualidade dos exercícios disponibilizados e o feedback dado, revendo mais vezes os exercícios e as suas alíneas, especificando o mais possível o feedback dado ao utilizador;
3. Adaptar os testes dos exercícios de forma a serem mais claros e com comentários claros para ajudar na sua interpretação. Remover ao máximo os *prints* que têm de ser realizados pelos alunos nas alíneas dos exercícios. E, no caso de existirem, facultar o seu formato na respetiva alínea do exercício para eliminar erros advindos de formatação desse *print*;
4. Reduzir/tornar mais sucinto o número de tarefas a realizar pelos alunos;
5. Acrescentar e permitir notificações via email, para exercícios em atraso, acompanhamento das atividades, mensagens na plataforma;
6. Criar uma plataforma adicional modelável de administração/visualização de dados da plataforma que possa ajudar na visualização e acompanhamento dos alunos, exercícios e participação;
7. Prever e implementar, se possível, algumas melhorias sugeridas pelos alunos como o *dark mode*.

5.3 Fase 2

A segunda fase de experimentação da plataforma *LMSOps* decorreu durante 2 meses e incidiu sobre o tópico de Estruturas Dinâmicas simples da linguagem de programação C. Devido à fraca participação por parte dos alunos, esta fase foi prolongada até à fase de exames (normal e de recurso). Esta fase esteve disponível para os 30 alunos de programação que se inscreveram para a primeira fase da experiência.

5.3.1 Implementação e funcionalidades

O percurso de atividades desta fase contava com:

1. Cinco exercícios *Code Runner*, exercícios codificados com testes para verificar a sua correta resolução, assim como feedback adequado a cada classificação possível;

2. Um inquérito final direcionado à experiência de aprendizagem e à experiência com a plataforma nesta segunda fase em comparação à primeira fase;
3. Um exercício teórico composto por alíneas de escolha múltipla direcionadas à compreensão de pedaços de código.

Com base na experiência, resultados obtidos e conclusões anteriores, esta segunda fase foi implementada respeitando os pontos de melhoria que foram apontados pelos alunos e as conclusões de melhoria definidas durante a primeira fase. Assim, os alunos passaram a dispor nesta fase de:

- Uma plataforma que passa a estar acessível através do endereço https://projmoodle-mis.isec.pt/, deixando de ser necessário utilizar a *VPN* do ISEC;
- O percurso de exercícios a realizar é menor, facilitando a possibilidade de o aluno terminar todas as atividades;
- Os exercícios e o seu conteúdo, testes e feedback, foram revistos e refinados mais vezes que na fase anterior, com o intuito de aprimorar o mais possível o seu conteúdo;
- Todas as notificações, mensagens e confirmações de realização de exercício recebidas através da plataforma são também enviadas para o email de cada conta;
- São enviadas notificações para exercícios em atraso (de acordo com a data prevista de resolução em cada exercício);
- Passa a ser possível realizar algumas customizações na plataforma *LMSOps* como mudar o tema que é aplicado.

A Figura 41, mostra um *overview* das tarefas na plataforma para esta fase.

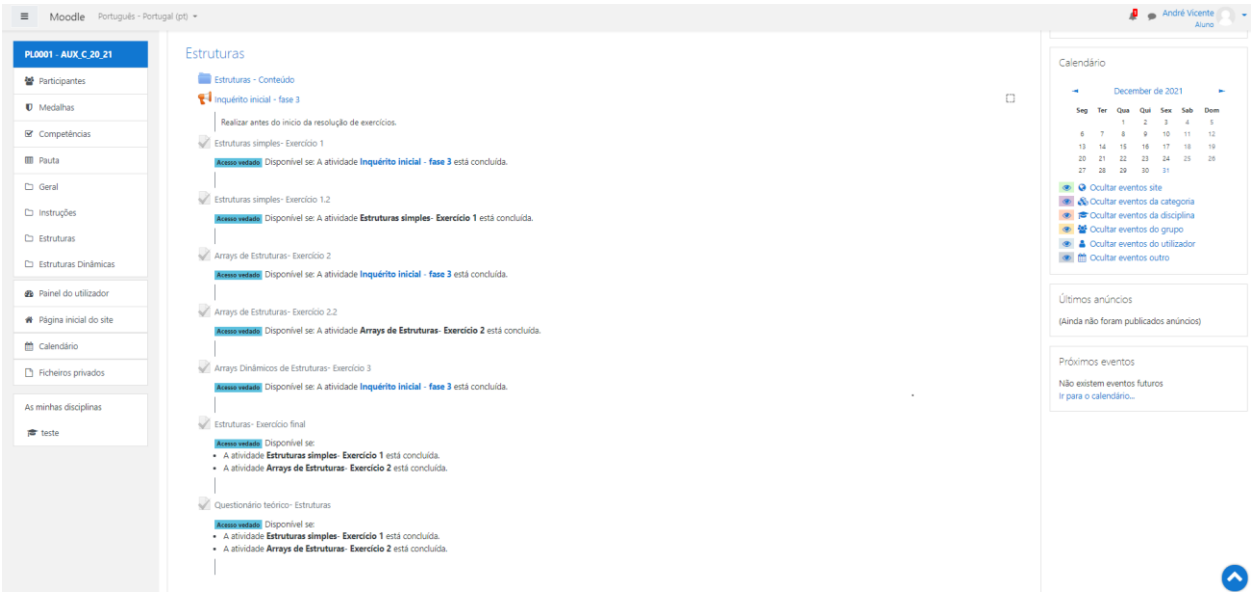


Figura 41 - *Overview* da disciplina fase 2

Foram mantidas as funcionalidades da versão 1 já existentes na primeira fase da experiência. Os automatismos referidos na versão 2 não foram desenvolvidos.

É possível verificar as instruções de uso e todas as funcionalidades no anexo E.

5.3.2 Resultados

5.3.2.1 Inquéritos

Quatro alunos iniciaram o percurso de exercícios desta segunda fase, mas apenas 1 aluno realizou esta fase por completo, tendo preenchido o respetivo inquérito final. É possível verificar a totalidade dos resultados obtidos no inquérito final desta fase no anexo B. Através destes resultados, foram obtidos os seguintes dados:

- O aluno saltou alguns passos, no entanto classificou a plataforma com 8 em relação à sua clareza e às instruções dadas numa escala de 0 a 10;
- O aluno não sentiu necessidade de pedir ajuda relacionada com os exercícios codificados, teve, no entanto, dúvidas relacionadas com matéria de Estruturas Dinâmicas. mas não pediu ajuda;
- O aluno classifica a facilidade de utilização dos exercícios, testes e a sua clareza com 8, enquanto o feedback obtido na resolução de exercícios é classificado com 9 numa escala de 0 a 10;
- O aluno classifica a utilização desta plataforma para aprendizagem de Estruturas Dinâmicas com 9, numa escala de 0 a 10, realçando que o seu ponto mais essencial/positivo é a programação/realização de exercícios;
- O aluno considerou a utilização da plataforma, nesta segunda fase, idêntica à primeira e que incentiva os alunos a programar e praticar exercícios algo que, na sua opinião, é essencial para a aprendizagem de programação.

Devido à baixa participação, não existiu a necessidade de capturar o ponto de vista do professor de programação que acompanhou a plataforma nesta fase.

5.3.2.2 Estatísticas de participação e desempenho dos alunos

5.3.2.3 Alunos

Dos dados relativos à participação dos utilizadores nos exercícios (cinco exercícios *Code Runner* e um exercício teórico), apenas um aluno realizou um ou mais exercícios da disciplina;

5.3.2.3.1 Exercícios

Relativamente à resolução dos exercícios, notas obtidas e tentativas realizadas:

- Houve 3 tentativas não submetidas por parte de três alunos no primeiro exercício;
- O aluno que completou os exercícios desta fase submeteu apenas uma tentativa em todos os exercícios, com exceção do exercício final, que não submeteu a sua tentativa;
- O exercício final não possui nenhuma submissão com nota atribuída.

A Figura 42, ilustra as observações anteriores. Para cada exercício são mostrados o número total de tentativas, o seu estado, a média, a nota máxima e mínima obtidas pelos alunos.

Estatísticas dos exercícios

name	Tentativas	Tentativas finalizadas	Tentativas em progresso	Média	Nota máxima	Nota mínima
Estruturas Dinâmicas- Exercício 1	4	1	3	18	19	17
Estruturas Dinâmicas- Exercício 1.2	1	1	0	19.5	20	19
Estruturas Dinâmicas- Exercício 2	1	1	0	16	20	12
Estruturas Dinâmicas- Exercício 2.2	1	1	0	20	20	20
Estruturas Dinâmicas- Teórico	1	1	0	12.7	14.7	10.7
Estruturas Dinâmicas- Exercício final	1	0	1	0	0	0

Figura 42 - Estatísticas dos exercícios da fase 2

5.3.3 Conclusões

Os resultados recolhidos nesta segunda fase e opiniões dos participantes são em pouco número. O aluno destacou a plataforma como intuitiva e o seu propósito de resolução de exercícios como ponto fulcral para aprender a programar. No entanto, achou a plataforma idêntica à primeira fase, não notando nenhuma das melhorias feitas nos conteúdos ou estrutura da mesma. Para alargar os resultados e feedback obtidos será importante realizar uma nova fase para revalidar novamente a plataforma.

Após esta segunda fase, foi traçado um plano para revisão da plataforma e do processo de ajuda à aprendizagem de programação para uma terceira e última fase. Assim, foram considerados os seguintes pontos de atuação/melhoria:

1. Alargar o grupo de alunos que poderá utilizar/participar na plataforma na terceira fase para assim tentar potenciar a possibilidade de participação;
2. Disponibilizar os conteúdos programáticos das primeira e segunda fases, para assim alargar as unidades programáticas da disciplina abrangidas pela plataforma;
3. Rever e refinar os conteúdos da primeira fase de acordo com o feedback recebido;
4. Tornar mais sucinto o percurso da primeira fase, aproximado do percurso da segunda fase;
5. Rever e refinar as instruções e passos presentes na plataforma para utilização da mesma.

5.4 Fase 3

A terceira fase de experimentação da plataforma *LMSOps* decorreu por 2 semanas durante a época especial de exames, que disponibilizou os conteúdos das primeira e segunda fases (tópico Estruturas e tópico Estruturas Dinâmicas). Foram disponibilizadas 100 novas contas de utilizador direcionadas aos alunos inscritos em época especial que tivessem interesse em treinar estes conteúdos programáticos. A divulgação da plataforma e a distribuição de contas por alunos inscritos na fase especial foi realizada pelo mesmo professor que acompanhou as fases anteriores.

5.4.1 Implementação e funcionalidades

O percurso contou com:

- Onze exercícios de programação, seis exercícios sobre o tópico Estruturas e cinco exercícios sobre o tópico Estruturas Dinâmicas. Exercícios codificados com testes para verificar a sua correta resolução e feedback apropriado;
- Dois inquéritos, um inicial com perguntas direcionadas ao percurso de programação e outro a realizar no final, direcionado à experiência de aprendizagem e com a plataforma nesta terceira fase;
- Dois exercícios teóricos, um do tópico de Estruturas outro do tópico de Estruturas Dinâmicas. Compostos por alíneas escolha múltipla direcionadas à compreensão de pedaços de código e à sua compilação.

Com base na experiência, resultados obtidos e conclusões das primeira e segunda fases, esta terceira fase foi implementada respeitando todos os pontos de melhoria anteriores, tendo como principal objetivo tentar obter uma maior participação por parte dos alunos a realizar a unidade curricular de Programação.

Foram melhorados e revistos todos os conteúdos disponibilizados na plataforma de acordo com todo o feedback recebido nas fases anteriores. O professor da disciplina passa a ter disponível uma plataforma de monitorização para acompanhar diversas estatísticas como os acessos, resolução de exercícios/notas e outros indicadores de desempenho, funcionalidade que estava prevista para a versão 3 da plataforma.

Além das funcionalidades e melhorias referidas anteriormente, foram mantidas as funcionalidades da versão 1 da plataforma já existentes na segunda fase da experiência.

A Figura 43, mostra um *overview* das tarefas na plataforma para esta fase.

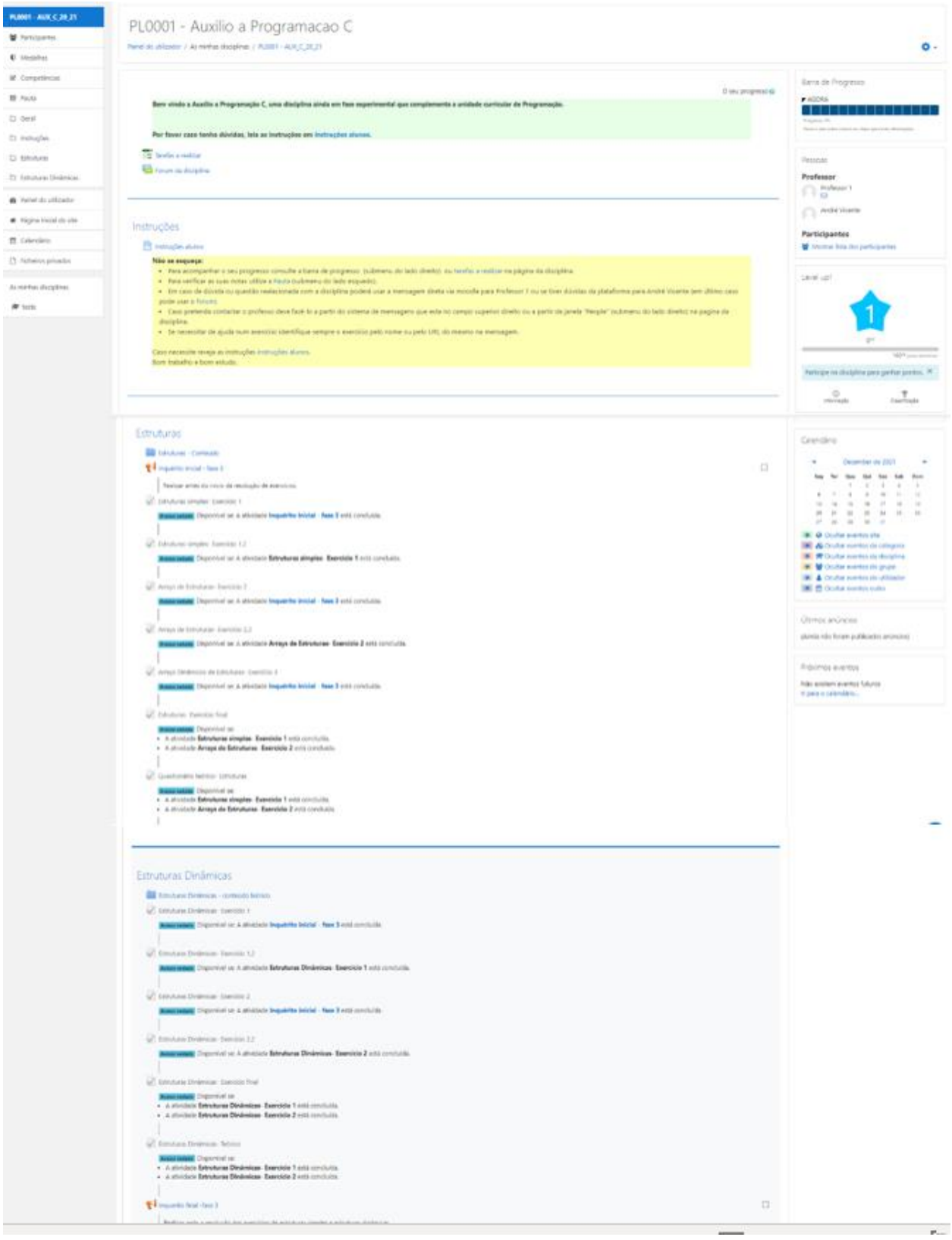


Figura 43 - *Overview* da disciplina fase 3

É possível verificar as funcionalidades adicionadas no anexo F.

5.4.2 Resultados

5.4.2.1 Inquéritos

Apenas um aluno preencheu os inquéritos.

É possível verificar a totalidade dos resultados obtidos no inquérito inicial no anexo C. Através do inquérito inicial foram obtidos os seguintes dados sobre o aluno que participou nesta fase:

- Aprendeu a programar há cerca de 1-2 anos, adora programar, já realizou Introdução à Programação e não está na primeira matrícula de programação, no entanto não foi ainda avaliado;
- Considera que as suas maiores dificuldades são desenvolver os algoritmos e a sintaxe da linguagem, dedicando semanalmente 30min-1h no estudo de programação e na realização de exercícios, frequentando (50%-75%) maioritariamente as aulas práticas e teóricas;
- Valoriza o método presencial por ser a forma mais fácil de interagir e esclarecer dúvidas com o professor.

O inquérito final teve perguntas direcionadas à plataforma e às suas funcionalidades. É possível verificar a constituição do inquérito final e a totalidade das respostas obtidas no anexo C. Através deste inquérito, foram obtidos os seguintes dados sobre a plataforma e as suas funcionalidades:

- O aluno saltou alguns passos, instruções e tarefas considerando a plataforma fácil e intuitiva de utilizar com classificação 10 numa escala de 0-10;
- O aluno foi-se adaptando às questões tendo, no entanto, necessidade de pedir ajuda várias vezes com questões técnicas de programação e relacionadas com a matéria;
- O aluno classificou a explicação dos exercícios, a facilidade de utilização do *Code Runner*, o feedback automático obtido e a experiência de utilização da plataforma no geral com 10 numa escalar de 0-10;
- O aluno considerou como pontos menos bons sair e entrar na plataforma e compreender algumas explicações/feedback dos exercícios;
- O aluno indica que o tempo de submissão dos exercícios na plataforma é elevado.

Foi também levantado o ponto de vista do professor que acompanhou esta última fase, relativamente à sua interação com a plataforma e a perguntas/dúvidas sobre a matéria de estruturas ou estruturas dinâmicas que tenham sido colocadas por partes dos alunos. Após o término desta fase foram realçados pelo professor os seguintes aspetos:

- A fase especial de exames é a menos regular da época de exames, na qual os alunos podem inscrever-se até 1 dia antes do exame. É por isso extremamente difícil identificar ou divulgar a plataforma aos alunos que realmente possam ter interesse e que vão realizar exame;

- As inscrições foram baixas nesta época assim poucas contas foram disponibilizadas aos alunos;

5.4.2.2 Estatísticas e desempenho dos alunos nos exercícios/percurso da plataforma

5.4.2.2.1 Alunos

Dos dados relativos à participação dos alunos nos exercícios (10 exercícios *Code Runner* e 2 exercício teórico), apenas 1 aluno submeteu resoluções a um ou mais exercícios da plataforma;

5.4.2.2.2 Exercícios

Relativamente à resolução dos exercícios, notas obtidas e tentativas realizadas:

- O aluno que completou os exercícios desta fase submeteu duas ou mais tentativa em todos os exercícios codificados, com exceção do exercício final de Estruturas, no qual, não submeteu a sua tentativa;
- O exercício final de Estruturas Dinâmicas possui apenas uma submissão, feita pelo aluno que realizou esta fase com 0 valores de nota.

A Figura 44, ilustra as observações anteriores. Para cada exercício são mostrados o número total de tentativas, o seu estado, a média, a nota máxima e mínima obtidas pelos alunos.

Estatísticas dos exercícios

name	Tentativas	Tentativas finalizadas	Tentativas em progresso	Média	Nota máxima		Nota mínima	
Estruturas simples- Exercício 1	18	11	7	19.9	20		19	
Arrays de Estruturas- Exercício 2	9	6	3	15.8	20		0	
Estruturas simples- Exercício 1.2	11	8	3	17.9	20		14	
Arrays de Estruturas- Exercício 2.2	6	4	2	9.25	20		0	
Estruturas- Exercício final	1	0	1	Sem nota	Sem nota		Sem nota	
Arrays Dinâmicos de Estruturas- Exercício 3	2	1	1	18	18		18	
Questionário teórico- Estruturas	6	6	0	16.3	20		13.3	
Estruturas Dinâmicas- Exercício 1	6	3	3	18	19		17	
Estruturas Dinâmicas- Teórico	2	2	0	12.7	14.7		10.7	
Estruturas Dinâmicas- Exercício 1.2	3	3	0	19.5	20		19	
Estruturas Dinâmicas- Exercício 2	4	4	0	16	20		12	
Estruturas Dinâmicas- Exercício 2.2	3	3	0	20	20		20	
Estruturas Dinâmicas- Exercício final	2	1	1	0	0		0	

Figura 44 - Estatísticas dos exercícios da fase 3

5.4.3 Conclusões

Esta terceira fase constituiu a última fase de experimentação desta plataforma. Os resultados recolhidos foram em pouco número. A plataforma foi novamente destacada como intuitiva,

assim como importante o seu propósito de resolução de exercícios pelo aluno que realizou esta fase.

A tentativa de alargar a amostra de alunos que poderiam participar nesta fase não teve sucesso, algo que se poderá dever em grande parte ao facto de a fase especial da época de exames ser irregular. Os alunos podem realizar a inscrição para o exame até à véspera do dia em que o mesmo se realiza, havendo assim alguma incerteza sobre quais os alunos que pretendem realizar a exame e propor-se à avaliação, podendo assim ter interesse na plataforma.

Após o término desta última fase de experimentação da plataforma foi realizada uma reunião final com o professor que acompanhou a plataforma, no sentido de expor a sua opinião sobre a plataforma e experiência realizada dando também o seu parecer geral.

Como pontos fortes foram realçados os seguintes pontos:

1. *"Ferramenta interativa que permite/incentiva a prática autónoma de programação. Disponibiliza percursos com exercícios, com dificuldade crescente, e possui material de apoio para ajudar a compreensão da matéria. Poderia ter igualmente alguns links para materiais externos";*
2. *"Compilação online, permitindo imediatamente a validação e teste do código produzido. Valoriza e acelera a recompensa recebida pelo aluno durante o estudo autónomo, uma vez que percebe imediatamente o que está a fazer e de que forma influencia o resultado. Pode ajudar a perceber, a um nível básico, qual o impacto nas alterações de código. Uma nova instrução altera o comportamento do programa e o resultado será diferente";*
3. *"Comunicação direta entre professor e aluno. O professor percebe o estado em que se encontra a realização dos exercícios, para cada um dos alunos. Os alunos podem, em qualquer momento, solicitar ajuda e esclarecer dúvidas";*
4. *"Do ponto de vista dos alunos, a ferramenta está bem organizada e clara. Os alunos facilmente têm acesso à informação e aos exercícios disponíveis, sendo clara a organização e encadeamento dos tópicos";*
5. *"Existe bastante material adicional disponível, complementando o que é visto em contexto de aula".*

Como pontos fracos/possibilidades de melhoria foram realçados os seguintes pontos:

1. *"A adesão foi fraca, mas este não é um problema da ferramenta ou da estratégia de aprendizagem que está a ser testada. Existe algum alheamento dos alunos, o que pode ser parcialmente justificado pelo facto do teste ter sido feito numa altura em que aulas decorreram exclusivamente de forma remota. Faltou igualmente garantir uma maior ligação entre a unidade curricular e a ferramenta";*
2. *"Melhorar/Aprofundar a perspetiva do docente em relação à realização das tarefas. Seria interessante ter uma visão global do estado da turma (existe uma visão global dos alunos e são produzidos alguns relatórios de aprendizagem, mas sinto que gostaria*

de ter alguma informação adicional), com dados mais detalhados e contextualizados. O ideal seria ter uma ferramenta de BI[8] que analisasse a informação, embora eu saiba que isto seria um outro projeto. O que foi feito, qual o tempo necessário, qual o percurso seguido, que tipo de ferramentas de auxílio permitiram desbloquear dificuldades, que tipo de grupos de alunos existem (alunos diferentes poderão ter dificuldades diferentes e poderão existir padrões interessantes). Estes dados poderiam surgir agregados e individualmente. A plataforma adicional desenvolvida e disponibilizada na fase 2 e 3 constitui um passo neste sentido, e poderá ser trabalhada e aprimorada neste sentido";

3. *"Criar percursos alternativos de forma dinâmica. O percurso ideal depende da maturidade/autonomia/grau de conhecimento do aluno. Alguns alunos poderão desmotivar com tarefas pouco desafiantes, enquanto que outros poderão achar que têm uma montanha demasiado alta para escalar";*
4. *"O Moodle tem a vantagem de ser um recurso simples e conhecido dos alunos, mas não terá alguma falta de flexibilidade que impede a implementação de outro tipo de estratégias? Os alunos poderiam sentir-se mais motivados se encontrassem um ambiente diferente do habitual?".*

[8] Business intelligence (BI) são tipos de software que recolhem e processam grandes quantidades de dados de outros sistemas internos e externos, possibilitando a configuração de indicadores e outras análises.

6 Conclusões

6.1 Enquadramento, investigação e fundamentação teórica

O presente projeto visou aliar a metodologia *DevOps* ao processo de ensino-aprendizagem de programação e foi desenvolvido ao longo de cerca de ano e meio durante os anos letivos 2019/2020 e 2020/2021.

Apesar deste projeto ter sido, inicialmente, bastante abrangente e com possibilidade de usar diferentes ferramentas, mecanismos e/ou abordagens, este acabou por se basear principalmente no processo que é aplicado no ensino-aprendizagem de programação no ensino superior e na sua adaptação e otimização.

A procura por artigos e fundamentação sobre o estado da aprendizagem de programação noutras instituições foi algo difícil, pois existem poucos artigos a falar da taxa de reprovação ou desistência nas unidades curriculares de programação de cursos superiores.

No levantamento inicial de ideias foi pensado desenvolver/integrar uma plataforma que pudesse, além de ajudar os alunos na aprendizagem de programação, ajudá-los também na sua integração no mundo empresarial, possibilitando o uso de ferramentas como o *GIT* ou a utilização de testes unitários.

No seguimento da investigação inicial foram observadas diversas plataformas que já tentavam cumprir este objetivo e que disponham de abordagens distinta. Algumas funcionavam numa base por subscrição paga, outras com o auxílio da comunidade na criação de cursos que podiam depois ser disponibilizados de forma gratuita ou por subscrição.

De seguida foram estudadas as plataformas *open-source* que podiam responder/servir o processo que se pretendia implementar. Com o objetivo de disponibilizar diversos recursos, síncronos e assíncronos que suportassem o processo de aprendizagem, as plataformas escolhidas necessitavam de ser *LMS* e permitir a criação e gestão de um "ecossistema" de aprendizagem para alunos.

6.2 Plataforma e fases de experimentação

A escolha recaiu sobre a plataforma *Moodle*. Apesar das restantes plataformas apresentarem algumas vantagens no que toca à resolução de exercícios, correção de código - através de testes - integração com *GIT* - através de *commits* e *merge requests* para submeter o código – as plataformas revelaram falta de capacidade em ter outros requisitos, como a disponibilização de conteúdos teóricos, exercícios teóricos ou de submissão. A distinção entre aluno e professor era também mais difícil, assim como a atribuição de notas. Mesmo lançadas através de outras plataformas, como por exemplo unindo o *Moodle* ao *Codeboard* ou *Moodle* ao *GitHub*, as mesmas não funcionavam corretamente ou então funcionavam de forma isolada levando à necessidade de utilização de contas e contextos distintos.

Apesar desta integração poder ser interessante, não foi alvo de exploração por ter sido pretendido focar ao máximo o projeto na ensino-aprendizagem de programação e não na utilização de ainda mais ferramentas adicionais ou outros conceitos acessórios.

Adicionalmente, existiram ainda outros fatores que levaram à afirmação do *Moodle* como plataforma a trabalhar, como o facto de esta já ser a plataforma usada no ISEC, nomeadamente na licenciatura em Engenharia Informática, facilitando a integração e aceitação por parte de alunos e professores. O facto de esta também ser modelável onde seria possível desenvolver módulos de raiz e utilizá-los neste âmbito é também uma mais-valia, apesar deste não ser o objetivo deste projeto.

Após decidir usar o *Moodle* como base, foram realizadas algumas provas de conceito e investigação de diversos plugins com intuito de construir a integração que pudesse melhor satisfazer o ciclo/processo desenhado.

Posteriormente à elaboração da primeira versão funcional foi também desenhado um plano de melhorias para as funcionalidades da disciplina experimental do Moodle, de maneira a puder ser desenvolvido para cada fase de experimentação.

No plano inicial foram previstas 2 fases de experimentação, sendo que posteriormente, foram alargadas para 3 fases na tentativa de aumentar a participação. Cada fase contou com conteúdos de uma unidade programática da disciplina, sendo que a terceira fase contou com os dois conteúdos programáticos da fase 1 e 2.

No decorrer de cada uma das fases foram levantados os pontos de vista dos participantes e também da sua experiência de maneira a melhorar a plataforma com base em todos os pontos identificados como menos positivos, quer pelos alunos que participaram, como pelo professor que acompanhou esta experimentação.

Após o término de todas as fases da experiência com a plataforma é possível identificar as seguintes conclusões:

- Foi extremamente difícil interessar os alunos que estavam a realizar a unidade curricular de programação da licenciatura em engenharia informática do ISEC a participar nesta disciplina.
- A divulgação da plataforma, assim com a angariação de alunos para participação na plataforma falhou, principalmente nas 2ª e 3ª fases. Tornou-se difícil divulgar e levar alunos a participar na plataforma, uma vez que não conseguimos prender o seu interesse com nenhuma "recompensa" ou gratificação, tendo a divulgação também decorrido à distância via email e fórum da disciplina de programação. Dos 30 alunos que mostraram interesse em participar na primeira fase houve 40% que nunca acedeu à plataforma ou que acedeu e não realizou qualquer exercício ou apenas o primeiro. Com uma manifestação ainda maior na segunda fase, na qual apenas um aluno chegou ao final do percurso de atividades. A terceira fase, mesmo tendo alargado a amostra de alunos que poderiam participar, a participação manteve-se em linha com a da segunda fase.
- Os alunos do primeiro ano da licenciatura são um dos públicos-alvo mais difícil para esta experiência. Normalmente o primeiro ano no ensino superior é um ano difícil para os alunos, no qual por norma acabam por não estar ainda comprometidos totalmente com as unidades curriculares e/ou curso, acabando alguns alunos por mudar de curso, instituição de ensino ou por deixar diversas unidades curriculares por fazer.
- A participação e frequentação dos alunos na disciplina de Programação também diminuiu ao longo do tempo, coincidindo com a participação dos alunos na plataforma ao longo das 3 fases. Tendo a fase de exames da disciplina de Programação da licenciatura em Engenharia Informática contado com cerca de 220 alunos inscritos na fase normal, 80 na fase de recurso e 30 na fase especial.
- O módulo adicional de gestão e acompanhamento dos resultados dos participantes da plataforma poderá ser essencial para utilizar o moodle como auxílio no processo de ensino de programação, uma vez que facilita a análise de resultados dos utilizadores da plataforma.
- O facto da experiência ter decorrido durante a altura pandémica dificultou bastante quer a promoção da plataforma, como incentivo/cativação de alunos para experimentar a mesma;
- Os alunos que participaram na experiência, ao contrário do que inicialmente seria espectável, foram alunos com dificuldades ou alunos que gostavam de programar e, por sua vez, que já tinham capacidades consideráveis e não os alunos mais "comuns" que não adoram programar e com dificuldades "mínimas". Este facto leva a que possa ser considerado que esta plataforma é de extrema importância para os alunos com mais dificuldades e um desafio que interessa os alunos que adoram programar, podendo ser potenciado pela *gamification* e concursos de resolução de exercícios entre participantes.

Ao longo das fases o conteúdo e o processo aplicado foram refinados e adaptados às necessidades dos alunos que estavam a frequentar as fases. Foi realçado nos comentários dos utilizadores que um dos fatores mais importantes, se não o mais importante, para aprender a programar é praticar e resolver exercícios, algo que foi o foco principal da plataforma.

6.3 Assunções finais e trabalho futuro

Auxiliar o processo de ensino-aprendizagem de programação recorrendo aos princípios de metodologia da indústria - como o *DevOps* - é possível. Existem diversas abordagens que podem ser tomadas não só nos conceitos aplicados - como foi o caso deste projeto - mas também nas ferramentas usadas. A abordagem seguida ao longo deste projeto baseou-se essencialmente no processo utilizado, na sua otimização e automação de tarefas para os seus intervenientes (professores e alunos). Apesar da participação na experiência ter ficado aquém do espectável, existem pontos positivos e assunções que poderemos retirar.

A dificuldade principal passou por interessar o "comum" aluno da unidade curricular de Programação pela plataforma. Não existiu nenhum incentivo adicional para utilizar a plataforma e muitos alunos podem apenas ter-se limitado a fazer as tarefas obrigatórias da unidade curricular, algo usual nos alunos de cadeiras introdutórias. No entanto, existiu algum entusiasmo por parte dos alunos que adoram programar que participaram na experiência, no sentido de se sentirem desafiados com os exercícios codificados. Será certamente um dos públicos mais aderentes a este tipo de plataformas.

A diversificação de exercícios é também um ponto importante, os exercícios teóricos recuperaram a participação na plataforma e desafiaram os alunos com situações menos usais de código ajudando os alunos a ver formas diferentes de programação. Assim como, a mostrar situações peculiares de codificação ajudando-os a entender melhor as unidades programáticas e as suas instruções. Realizar esta experiência com alunos e cadeiras que não sejam de primeiro ano poderá ajudar a aumentar a participação na plataforma e a sua aceitação obtendo resultados melhores.

Assim, podemos afirmar que esta aproximação entre o *DevOps* e processo de ensino-aprendizagem de programação é possível e poderá ser explorada para beneficiar alunos e professores.

Existe um enorme espaço de melhoria quer no processo, como nas funcionalidades que a plataforma pode ter. Sendo assim uma forte mais-valia para os alunos conseguirem praticar programação e os professores conseguirem ver o progresso dos alunos em qualquer disciplina de programação e com a possibilidade de uma melhoria continua no processo de educação de ensino-aprendizagem de programação.

Com o intuito de contribuir para a continuação de estudos idênticos foi também publicado o artigo - *Applying the DevOps methodology for a more efficient process of teaching-learning computer programming*[9] - sobre este trabalho, na *31st Annual Conference of the European Association for Education in Electrical and Information Engineering (EAEEIE)* 2022 com o resumo de todo este estudo e as suas conclusões.

[9] Ver anexo G

Referências

[1] P. Szlávi e L. Zsakó, "Methods of teaching programming," 2003.

[2] J. Bennedsen e M. E. Caspersen, "Failure Rates in Introductory Programming — 12 Years Later," vol. 10, nº 2, pp. 30-36, Junho 2019.

[3] S. A. Luxton-Reilly, V. V. Ajanovski, E. Fouh, C. Gonsalvez, J. Leinonen, J. Parkinson, M. Poole e N. Thota, "Pass Rates in Introductory Programming and in other STEM," pp. 53-71, Dezembro 2019.

[4] A. Gomes, W. Ke, S. K. Lm, A. Siu, A. J. Mendes e M. J. Marcelino, "A teacher's view about introductory programming teaching and learning — Portuguese and Macanese perspectives," Outubro 2017.

[5] T. Pham, "Analyzing The Software Engineer Shortage," Forbes, 13 Abril 2021. [Online]. Available: https://www.forbes.com/sites/forbestechcouncil/2021/04/13/analyzing-the-software-engineer-shortage/?sh=770e6a23321c. [Acedido em 2021].

[6] P. Muncaster, "Infosecurity Magazine," 7 Novembro 2019. [Online]. Available: https://www.infosecurity-magazine.com/news/cybersecurity-skills-shortage-tops/. [Acedido em 2021].

[7] N. Kosenko, "The Software Developer Shortage in the US and the Global Tech Talent Shortage in 2022," Daxx, 5 Janeiro 2022. [Online]. Available: https://www.daxx.com/blog/development-trends/software-developer-shortage-us. [Acedido em 2022].

[8] H. C. Peralta, "Falta de profissionais nas TI faz disparar salários em Portugal," *Diário de Notícias,* nº 55494, 28 Março 2021.

[9] "ACEF/1819/0209152 — Decisão do CA," [Online]. Available: https://a3es.pt/sites/default/files/ACEF_1819_0209152_acef_2018_2019_dec_fin_ca.pdf. [Acedido em 2021].

[10] ISEC e IPC, "Relatório de Avaliação do Curso," Politécnico de Coimbra, 2018/2019.

[11] T. Jenkins, "On the difficulty of learn to progr[am]," *3rd Annual LTSN-ICS Conference, Loughborough University,* pp. 53-58, 2002.

[12] S.-S. Team, "Shadow-Soft," 28 Junho 2017. [Online]. Available: https://shadow-soft.com/why-devops-important/. [Acedido em 17 04 2020].

[13] R. E. Mayer, "Annual Review of Psychology," *Teaching of Subject Matter,* 22 Setembro 2003.

[14] N. C. C. Brown e G. Wilson, "Ten quick tips for teaching programming," *PLoS Comput Biol 14(4): e1006023.,* p. https://doi.org/10.1371/journal.pcbi.1006023, 2018.

[15] S. Mohorovicic e E. Tijan, *New technologies in teaching university level programming,* Janeiro 2010.

[16] J. Lestal, "History of programming languages," 5 Agosto 2020. [Online]. Available: https://devskiller.com/history-of-programming-languages/. [Acedido em 2021].

[17] C. a. L. F. W. B. Watson, "Failure rates in introductory programming revisited," *Proceedings of the 2014 conference on Innovation technology in computer science education (ITiCSE '14).,* pp. 39-44, Junho 2014.

[18] M. Giannakos, I. Pappas, L. Jaccheri e D. G. Sampson, "Understanding student retention in computer science education: The role of environment, gains, barriers and usefulness," *Springer Science+Business Media New York 2016,* 2016.

[19] T. A. Rashid e R. F. E. Farra, "Assessing Teaching Methods in Programming A Case Study: Salahaddin University," em *The 4th International Scientific Conference of Salahaddin University-Erbil,* 2011.

[20] Z. Papp-Varga, P. Szlávi e L. Zsakó, "Annales Mathematicae et Informaticae," *ICT teaching methods–Programming languages,* p. 163–172, Janeiro 2008.

[21] C. J. J. P. a. N. R. M. Betsy Beyer, Site Reliability Engineering, O'Reilly Media, Inc, 2017.

[22] J. Korsun, "What is DevOps and Why You Should Have It," djangostars, 25 Maio 2017. [Online]. Available: https://djangostars.com/blog/what-is-devops/. [Acedido em 18 04 2020].

[23] ISEC, "História," [Online]. Available: isec.pt/pt/instituto/#lnkHistoria. [Acedido em 2020].

[24] S. IPC, "Relatório de Avaliação do Curso 9885: Licenciatura - Engenharia Informática - Regime Pós-Laboral," Politécnico de Coimbra, 2019/2020.

[25] S. IPC, “Relatório de Avaliação do Curso 9770: Licenciatura - Engenharia Informática (Curso Europeu),” Politécnico de Coimbra, 2018/2019.

[26] ISEC, “Relatório de Avaliação do Curso 6091190: Licenciatura em Engenharia Informática 2019/2020,” Politécnico de Coimbra, 2019/2020.

[27] ISEC, “Relatório de Avaliação do Curso 6098850: Licenciatura em Engenharia Informática - Regime Pós-Laboral 2019/2020,” Politécnico de Coimbra, 2019/2020.

[28] ISEC, “Relatório de Avaliação do Curso 6097700: Licenciatura em Engenharia Informática (Curso Europeu) 2019/2020,” Politécnico de Coimbra, 2019/2020.

[29] NETPA e ISEC, “Consulta publica de notas,” [Online]. Available: https://netp.isec.pt/netpa/page?stage=ConsultaPublicaNotasAluno&plano=&ramo=&disciplina=&cdLectivo=&codeCurso=&periodo=&epocaAvaliacao=&turma=&codeAnoSemestreCurricular=. [Acedido em 2019].

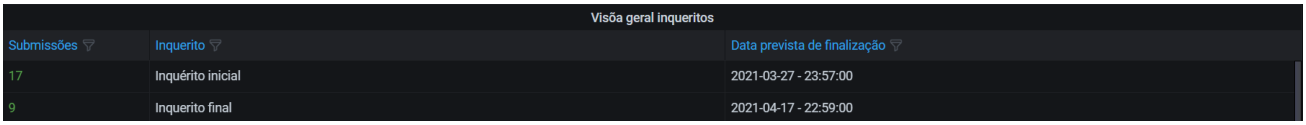
[30] A. J. Mendes, L. Paquete, A. Cardoso e A. Gomes, “Increasing student commitment in introductory programming learning,” *42nd ASEE/IEEE Frontiers in Education Conference*, pp. 82-87, 2012.

Anexo A: Plataforma LMSOps – fase 1

Nesta secção encontram-se descritas as questões e a totalidade dos resultados relativos aos inquéritos da primeira fase de experimentação da plataforma.

1. Inquérito Inicial

Relativamente ao inquérito inicial realizado aos alunos, foram submetidas 17 respostas conforme mostra a Figura 1.



Visõa geral inqueritos

Submissões	Inquerito	Data prevista de finalização
17	Inquérito inicial	2021-03-27 - 23:57:00
9	Inquerito final	2021-04-17 - 22:59:00

Figura 1- Participação inquéritos

1.1. Questões

1. Há quanto tempo aprendeu a programar?
Respostas possíveis: Este ano; 1-2 anos; 3-4 anos; >4 anos

2. Numa escala de 0-10 quanto gosta de programar?
Respostas possíveis: Não consigo saber se gosto ou não; 0-4 - Detesto programar / Não gosto particularmente de programar; 5-7 - Não gosto nem desgosto; 8-10 - Adoro programar.

3. Já realizou a cadeira de introdução à programação? Se sim, em que escala se enquadra a sua nota?
Respostas possíveis: Não fiz avaliação; <8; 8 – 9; 10 – 12; 13 – 15; 16 – 18; 19 – 20.

4. Se estiver a repetir Programação, que nota teve antes?
Respostas possíveis: Esta é a minha primeira matrícula, Não é a minha primeira matrícula, mas não frequentei/não fui avaliado a Programação, <8, 8 – 9, 10 – 12, 13 – 15, 16 – 18, 19 – 20.

5. Quais são as maiores dificuldades que tem sentido em Programação (pode assinalar várias)?
Respostas possíveis: Sintaxe da linguagem; Compreender os Problemas; Desenvolver os Algoritmos; Turma muito lotada ou dificuldade em manter a concentração ou muito barulho; Dificuldade em acompanhar o ritmo da aula ou pouco acompanhamento por parte do professor; Codificar; Testar; Debug; Outras.

6. Se anteriormente respondeu "Outras", pode indicar quais?
Respostas possíveis: Resposta curta.

7. Quais os aspetos que considera mais o ajudarem a aprender programação (pode assinalar várias)?
Respostas possíveis: Estudar cuidadosamente e compreender as bases teóricas; Realizar muitos exercícios; As explicações dadas pelos professores das aulas teóricas; O apoio dos professores das aulas laboratoriais; Trabalhar em grupo e o apoio dos colegas; Os fóruns de discussão ou outras plataformas (ex: stackoverflow.com, youtube.com, udemy.com); Utilização de uma ferramenta ou plataforma (ex. CodeRunner) com testes pré-preparados e feedback imediato; Outros.

8. Se anteriormente respondeu "Outros", pode indicar quais?
Respostas possíveis: Resposta curta de desenvolvimento.

9. Quanto tempo costuma dedicar, semanalmente, ao estudo de programação?
Respostas possíveis: <30min; 30min-1h; 1-2h; 2-3h; >3h.

10. Quanto tempo costuma dedicar, semanalmente, na realização de exercícios de programação? Respostas possíveis: <30min; 30min-1h; 1-2h; 2-3h; >3h.

11. Costuma frequentar as aulas teóricas?
Respostas possíveis: Não costumo frequentar; <25%; 25-50%; 50-75%; >75%.

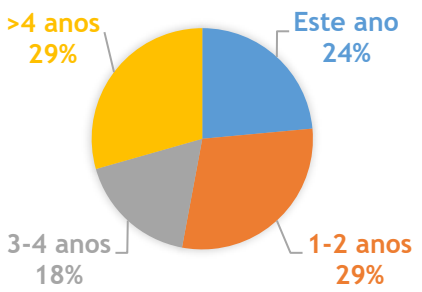
12. Costuma frequentar as aulas laboratoriais?
Respostas possíveis: Não costumo frequentar; <25%; 25-50%; 50-75%; >75%.

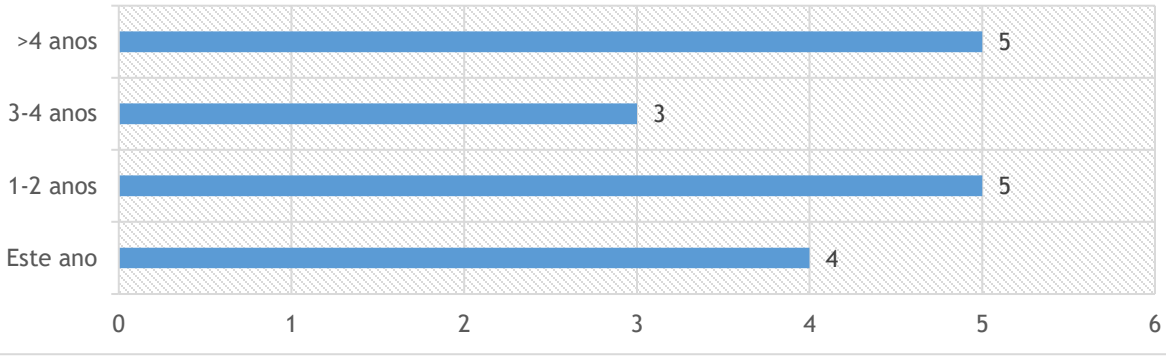
13. Quais são os aspetos mais positivos no ensino-aprendizagem de programação no ISEC?
Respostas possíveis: Resposta curta de desenvolvimento.

14. O que considera que deveria mudar no ensino da programação no ISEC de modo a melhorar a sua aprendizagem?
Respostas possíveis: Resposta curta.

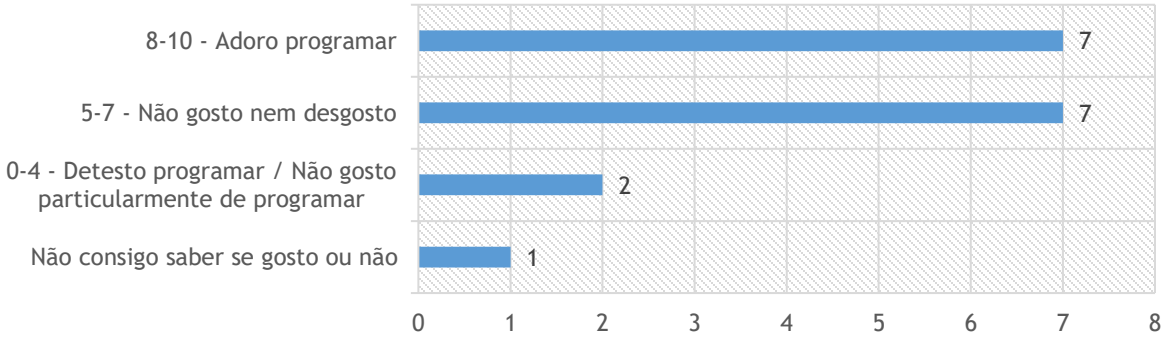
1.2. Resultados

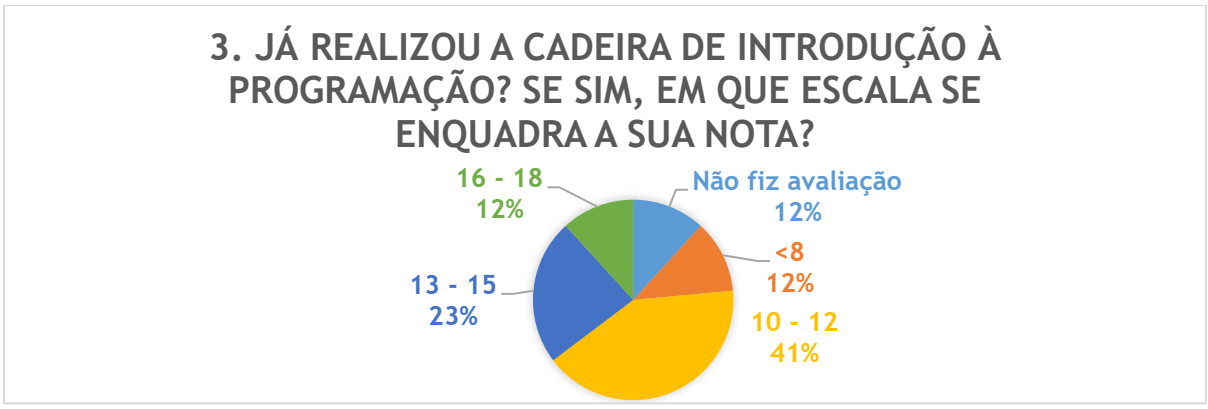
1. HÁ QUANTO TEMPO APRENDEU A PROGRAMAR?

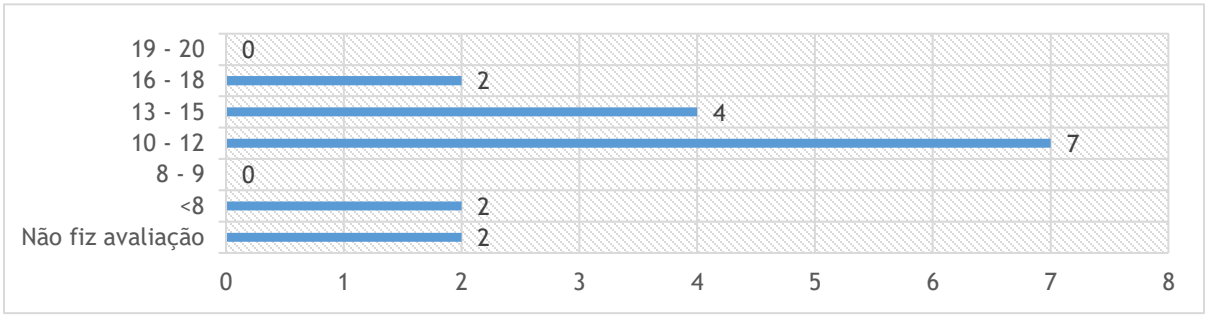




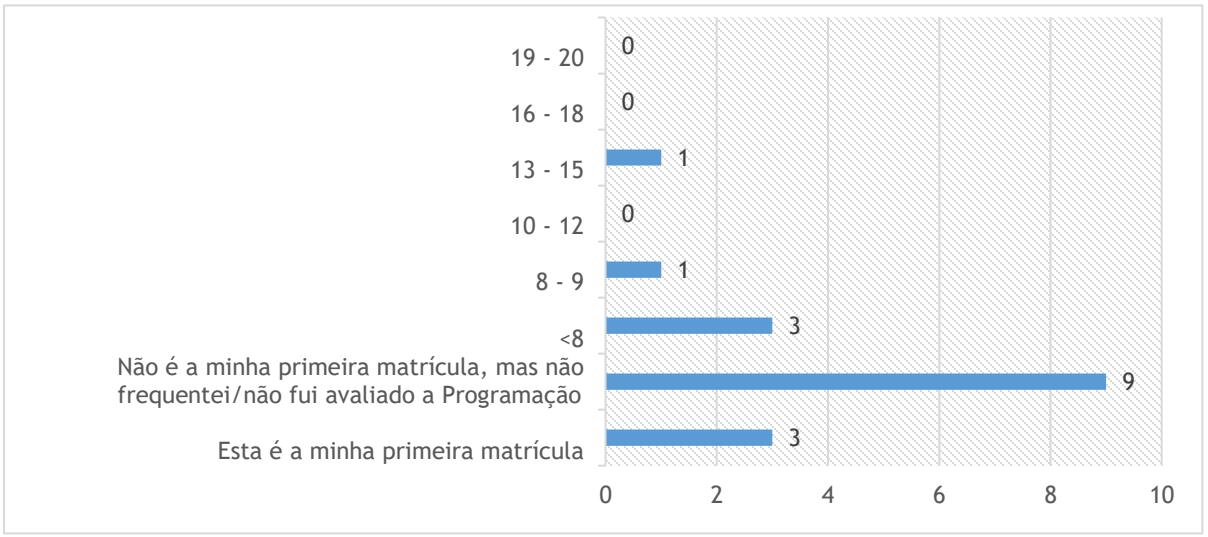
2. NUMA ESCALA DE 0-10 QUANTO GOSTA DE PROGRAMAR?




3. JÁ REALIZOU A CADEIRA DE INTRODUÇÃO À PROGRAMAÇÃO? SE SIM, EM QUE ESCALA SE ENQUADRA A SUA NOTA?
16 - 18
12%
Não fiz avaliação
12%
<8
12%
13 - 15
23%
10 - 12
41%

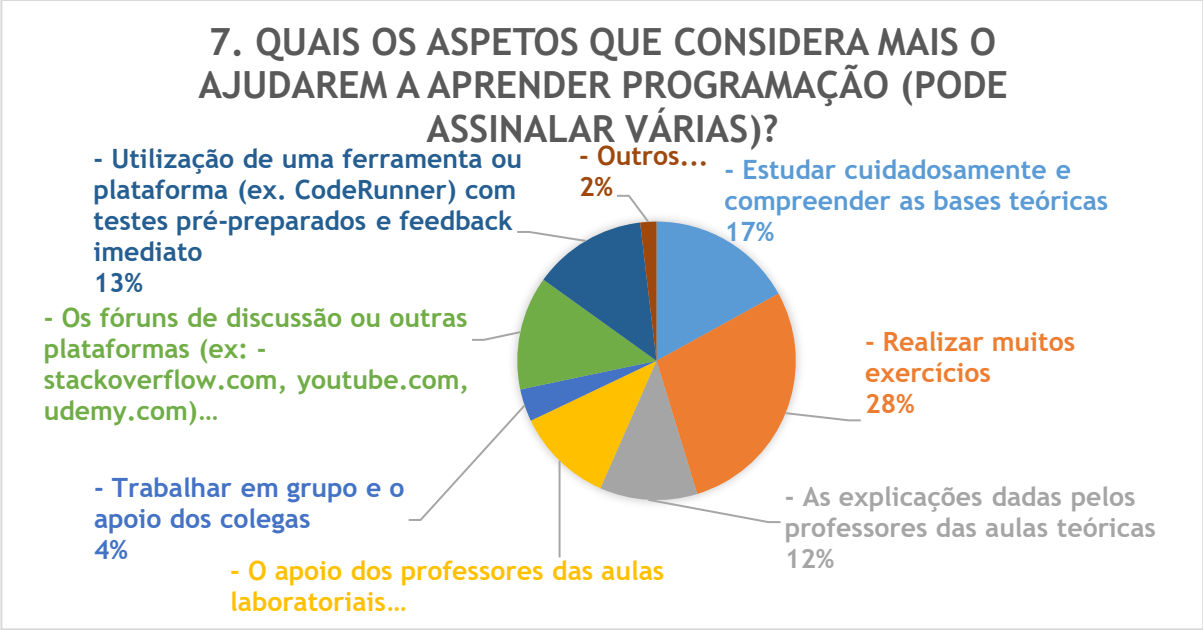

19 - 20
16 - 18
13 - 15
10 - 12
8 - 9
<8
Não fiz avaliação
0
2
4
7
0
2
2
0
1
2
3
4
5
6
7
8

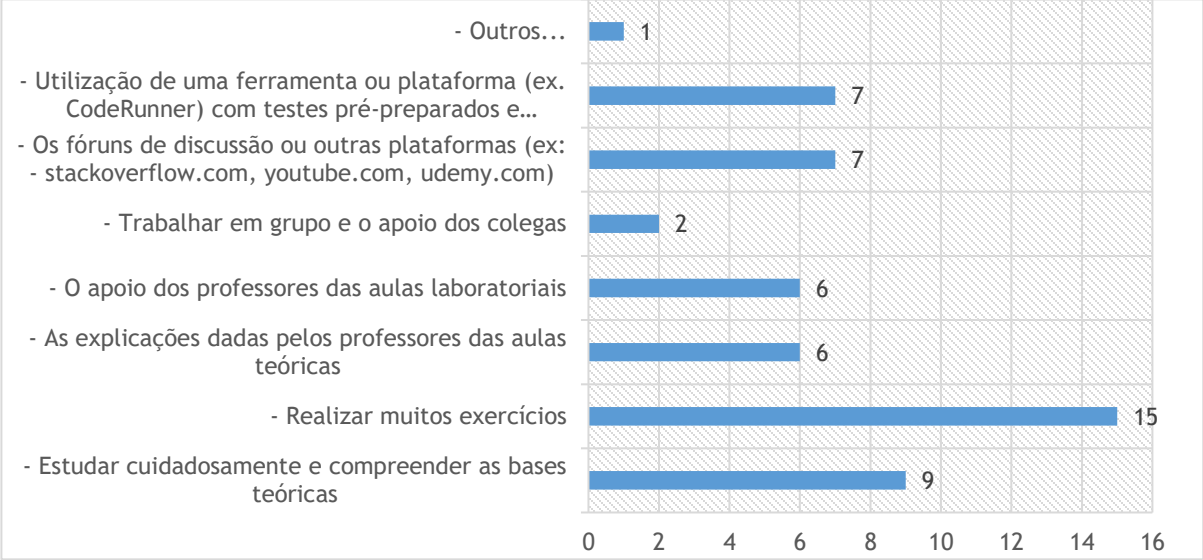

4. SE ESTIVER A REPETIR PROGRAMAÇÃO, QUE NOTA TEVE ANTES?
13 - 15
6%
8 - 9
6%
<8
18%
Esta é a minha primeira matrícula
17%
Não é a minha primeira matrícula, mas não frequentei/não fui avaliado a Programação…


19 - 20
16 - 18
13 - 15
10 - 12
8 - 9
<8
Não é a minha primeira matrícula, mas não frequentei/não fui avaliado a Programação
Esta é a minha primeira matrícula
0
0
1
0
1
3
9
3
0
2
4
6
8
10

6. Se anteriormente respondeu "Outras", pode indicar quais?

- Ainda tenho alguns hábitos de C# que às vezes dificultam o pensamento em C;
- Até percebo bem a matéria e consigo acompanhar os exercícios, mas sinto que podia melhorar o algoritmo, e fico sempre com muitas dúvidas se é a melhor maneira de os resolver;
- "Desenvolver uma API simples, robusta, simples e escalável. Usar o paradigma adequado à natureza do projeto.".

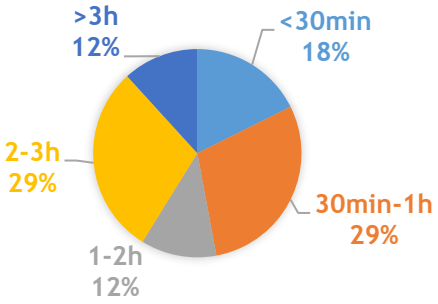


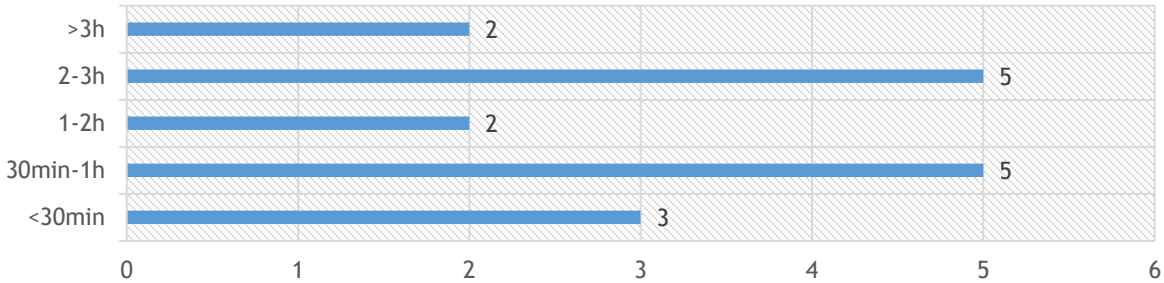


8 - Se anteriormente respondeu "Outros", pode indicar quais?

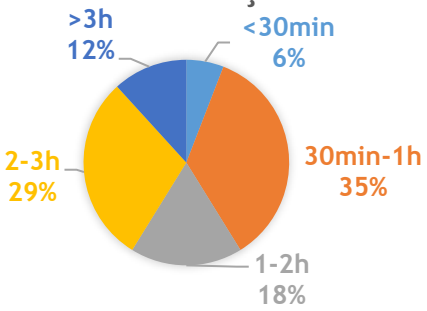
Compreender decisões de design de outros *source codes*. Compilar com *flags* relevantes. E.g.: -std=c99 -Wall -Wextra -Wpedantic -Wconversion -Wshadow. Ter à mão um C standard para rápida consulta."

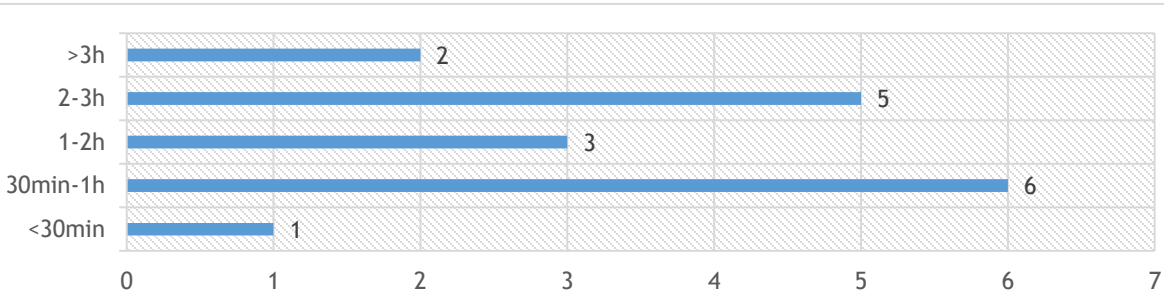
9. QUANTO TEMPO COSTUMA DEDICAR, SEMANALMENTE, AO ESTUDO DE PROGRAMAÇÃO?

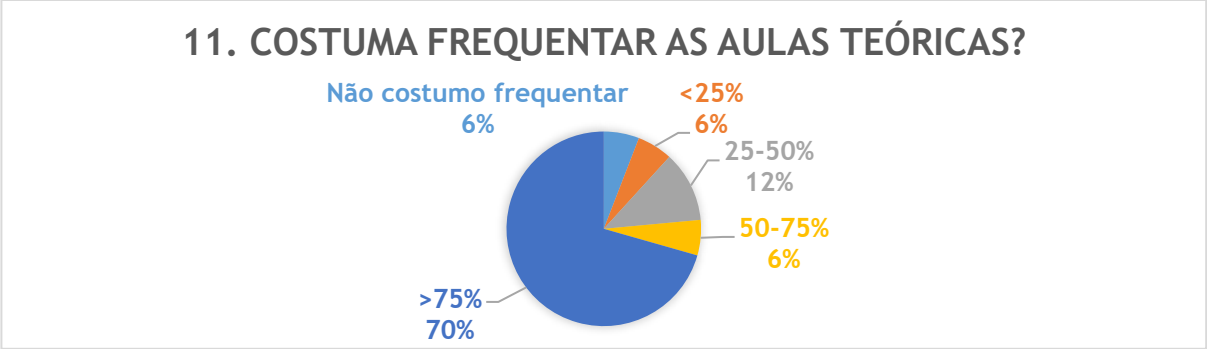


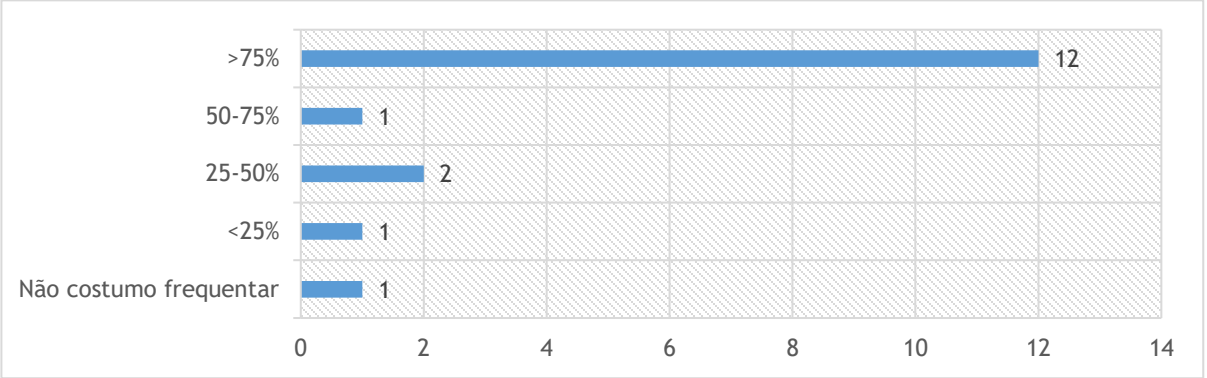


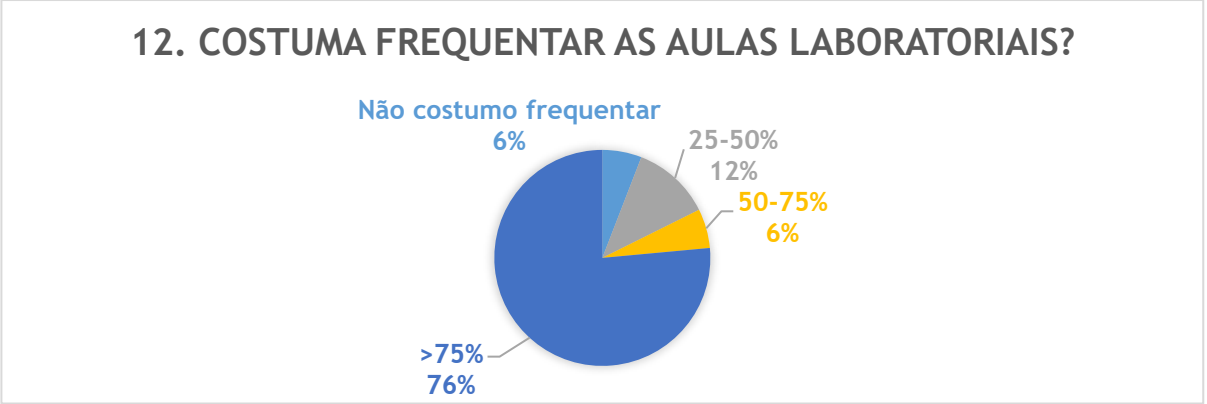
10. QUANTO TEMPO COSTUMA DEDICAR, SEMANALMENTE, NA REALIZAÇÃO DE EXERCÍCIOS DE PROGRAMAÇÃO?

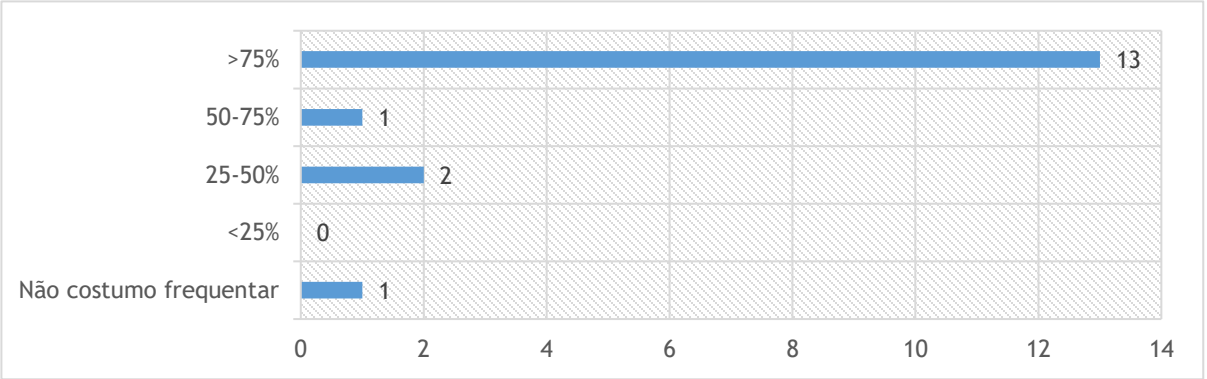





11. COSTUMA FREQUENTAR AS AULAS TEÓRICAS?
Não costumo frequentar
6%
<25%
6%
25-50%
12%
50-75%
6%
>75%
70%


>75%
12
50-75%
1
25-50%
2
<25%
1
Não costumo frequentar
1
0
2
4
6
8
10
12
14


12. COSTUMA FREQUENTAR AS AULAS LABORATORIAIS?
Não costumo frequentar
6%
25-50%
12%
50-75%
6%
>75%
76%


>75%
13
50-75%
1
25-50%
2
<25%
0
Não costumo frequentar
1
0
2
4
6
8
10
12
14

13. Quais são os aspetos mais positivos no ensino-aprendizagem de programação no ISEC?

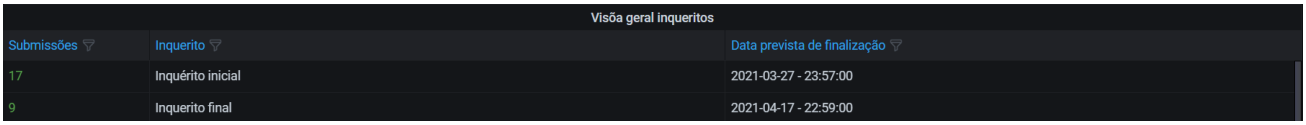
- As explicações dos professores e a forma como explicam os algoritmos;
- Os professores são impecáveis quer na explicação de conteúdos nas aulas quer na sua disponibilidade para nos esclarecer as dúvidas o quanto possível;
- A facilidade de comunicação dos professores, a sua disponibilidade para ajudar sempre que temos dúvidas;
- Os professores mostram-se sempre dispostos a ajudar os alunos com dúvidas;
- Os exemplos práticos que são dados, quer nas aulas teóricas quer nas práticas e esquemas;
- A disponibilidade de esclarecimento de dúvidas pelos professores;
- Recursos dos professores;
- Não sei;
- Acho que dá boas bases para o ensino da programação. Embora, a cadeira de IP insiste muito em pseudocódigo e fluxogramas. Para mim não faz sentido aprofundar esta matéria porque cada um à medida que aprende a programar consegue fazer o seu próprio pseudocódigo;
- As disciplinas em si terão todo o interesse. Acho fundamental a componente humana, o professor saber despertar e cativar os alunos. É o caso com os professores deste ano;
- Tanto as aulas teóricas como as aulas práticas são proveitosas e os professores fazem um acompanhamento da matéria boa;
- Os professores;
- Os guiões laboratoriais que têm muitos e bons exercícios;
- O modo que os docentes transmitem as informações, vindo de alguém do Brasil, eu considero bem explicado e detalhado, creio que isso seja o ponto mais positivo disso tudo; Provavelmente de os alunos acharem o ISEC como uma autoridade no ensino e de isso os motivar a aprender;
- Não é bem o aspeto positivo, mas também responde. As aulas do Prof. Mateus Mendes e o método de ensino dele para mim é o melhor para quem está a aprender do zero;
- Bons professores. As linguagens lecionadas têm impacto no mercado de trabalho.

14. O que considera que deveria mudar no ensino da programação no ISEC de modo a melhorar a sua aprendizagem?

- A cadeira de IP não devia insistir tanto no pseudocódigo. Penso que as aulas teóricas de IP deviam ser como as aulas de P, ou seja, usar o quadro e o IDE para explicar com exemplos;
- O resto dos professores adotarem o método do Prof. Mateus Mendes;
- Algum tipo de atividade realizada com grupos ou duplas, logicamente nada relacionado ao TP, acho interessante a troca de informações de programadores, eu particularmente acho isso uma boa ferramenta, sempre aprendo algo novo, como também transmito;
- Não sei;
- Considero que iniciativas como esta sejam uma boa direção a tomar. Programação requer muitas horas de dedicação fora da sala de aula e podendo de alguma forma criar trabalhos/exercícios interativos em que os alunos possam receber um feedback imediato em resposta às suas horas de trabalho é na minha opinião extremamente valiosa;
- No primeiro ano, é dado pouco ênfase a disciplinas de programação;
- Neste momento, não tenho ainda nenhuma sugestão que possa melhorar;
- Avaliar a capacidade de um aluno programar em teste escrito (programar com papel e caneta) não mostra a capacidade real que o aluno tem de programar;
- Fazer *quizzes* simples com o resumo da aula;
- Acho que exercícios *CodeRunner*, para podermos treinar mais exercícios do que aqueles que nos oferecem nas fichas seria proveitoso;
- Disponibilização de uma maior quantidade de exercícios no *CodeRunner* e com exemplos de resolução para facilitar o estudo autônomo dos alunos;
- Nada deveria mudar;
- Nada;
- Acho que os professores deviam, nas aulas dar espaço para os alunos resolverem por si mesmos, os exercícios, mas penso que no final da aula, os professores deviam resolver na mesma, pois não só ajuda os alunos com mais dificuldades, mas também pode dar uma perspetiva diferente daquela que os alunos possam ter pensado;
- Honestamente nada, acho que o ensino de programação é muito bom;
- Estou bastante confortável com a forma como está a decorrer. Acho também que a hipótese de aula remota devia ser para manter depois da pandemia, sem dúvida;
- Continuar com este tipo de ferramentas.

2. Inquérito Final

Relativamente ao inquérito final realizado aos utilizadores, foram submetidas 9 respostas.



Visão geral inqueritos

Submissões	Inquerito	Data prevista de finalização
17	Inquérito inicial	2021-03-27 - 23:57:00
9	Inquerito final	2021-04-17 - 22:59:00

Figura 2 - Participação nos inquéritos

O mesmo foi dividido em 2 partes, uma parte realizada na plataforma com perguntas direcionadas à interação com a plataforma e as suas funcionalidades, outra parte realizada através de *form* disponível *google forms* e com perguntas direcionadas à aprendizagem do tópico de programação que foi coberto nesta fase.

2.1. Perguntas da primeira parte do inquérito final

1. Realizou todo este programa/tarefas?
   Respostas possíveis: Realizei todos os passos cuidadosamente; Realizei todos os passos, mas a dada altura já não os fiz com atenção; Saltei alguns passos; Saltei todos ou quase todos os passos que podia até chegar ao fim; Não cheguei ao fim.

2. Acha que as instruções dadas por esta plataforma eram completas e claras?
   Respostas possíveis: 0- Nada claras e muito incompletas; 1; 2; 3; 4; 5; 6; 7; 8; 9; 10- Claríssimas e muito completas.

3. Foi fácil e intuitivo usar esta plataforma, seguindo todos os passos?
   Respostas possíveis: 0- Nada fácil nem intuitivo; 1; 2; 3; 4; 5; 6; 7; 8; 9; 10- Extremamente fácil e intuitivo.

4. Sentiu necessidade de pedir ajuda relacionada com questões técnicas de programação, para conseguir avançar nos exercícios?
   Respostas possíveis: Não; Sim, mas acabei por não pedir; Sim, 1 ou 2 vezes; Sim, várias vezes.

5. Quais foram as maiores dificuldades que teve?
   Respostas possíveis: Resposta curta.

6. Pediu ajuda relacionada com a matéria?

Respostas possíveis: Não, nunca tive necessidade; Não, mas até tinha necessidade; Sim, quanto tenho dúvidas costumo pedir ajuda; Sim, por vezes quando necessito peço ajuda; Sim, e nem é costume eu pedir ajuda.

7. Como classifica a clareza das explicações sobre os exercícios?

Respostas possíveis: 0- Nada claras; 1; 2; 3; 4; 5; 6; 7; 8; 9; 10- Claríssimas.

8. Como classifica a facilidade de utilização do *Code Runner* para a realização dos exercícios+testes?

Respostas possíveis: 0- Não foi nada fácil; 1; 2; 3; 4; 5; 6; 7; 8; 9; 10- Foi extremamente fácil.

9. Como classifica o feedback que obteve (explicações e feedback automático dos exercícios, notas, feedback do professor)?

Respostas possíveis: 0- Inútil; 1; 2; 3; 4; 5; 6; 7; 8; 9; 10- Extremamente útil.

10. Como classifica a sua experiência de utilização desta plataforma para aprendizagem de Estruturas?

Respostas possíveis: 0- Muito má; 1; 2; 3; 4; 5; 6; 7; 8; 9; 10- Excelente.

11. Quais são os pontos positivos desta experiência?

Respostas possíveis: Resposta curta.

12. Quais são os pontos menos bons?

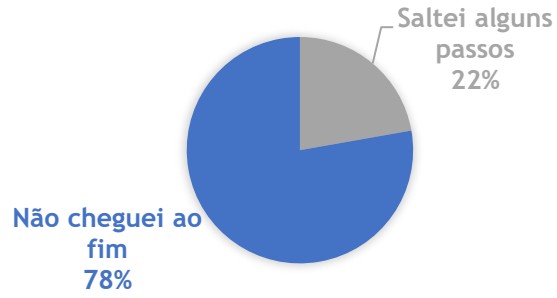
Respostas possíveis: Resposta curta.

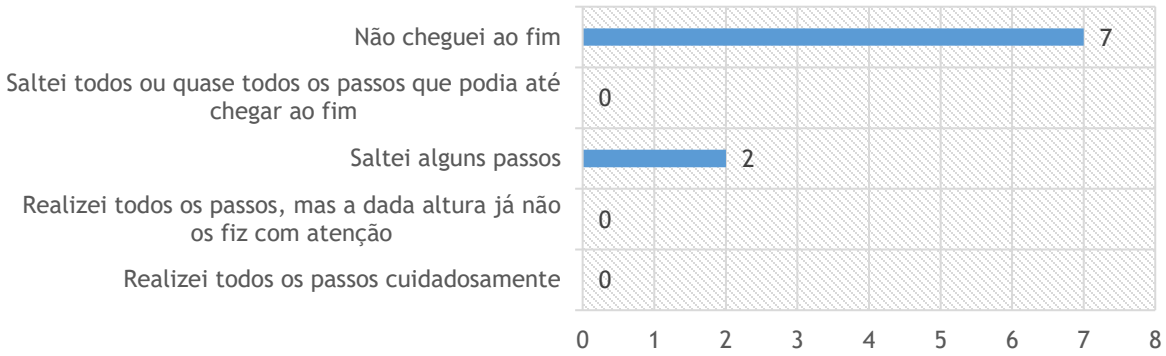
13. Tem sugestões de melhoria?

Respostas possíveis: Resposta curta.

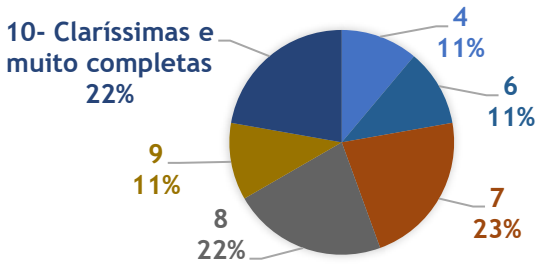
2.2. Resultados da primeira parte do inquérito final

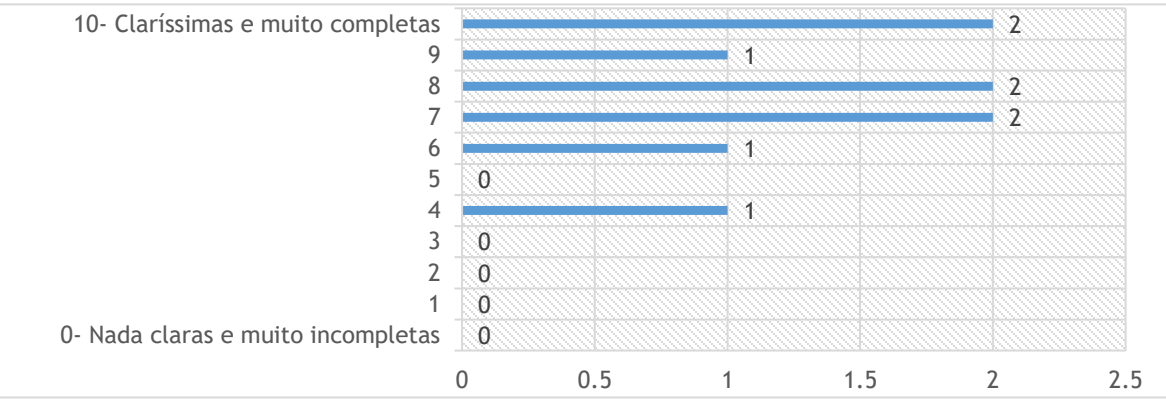
1. REALIZOU TODO ESTE PROGRAMA/TAREFAS?

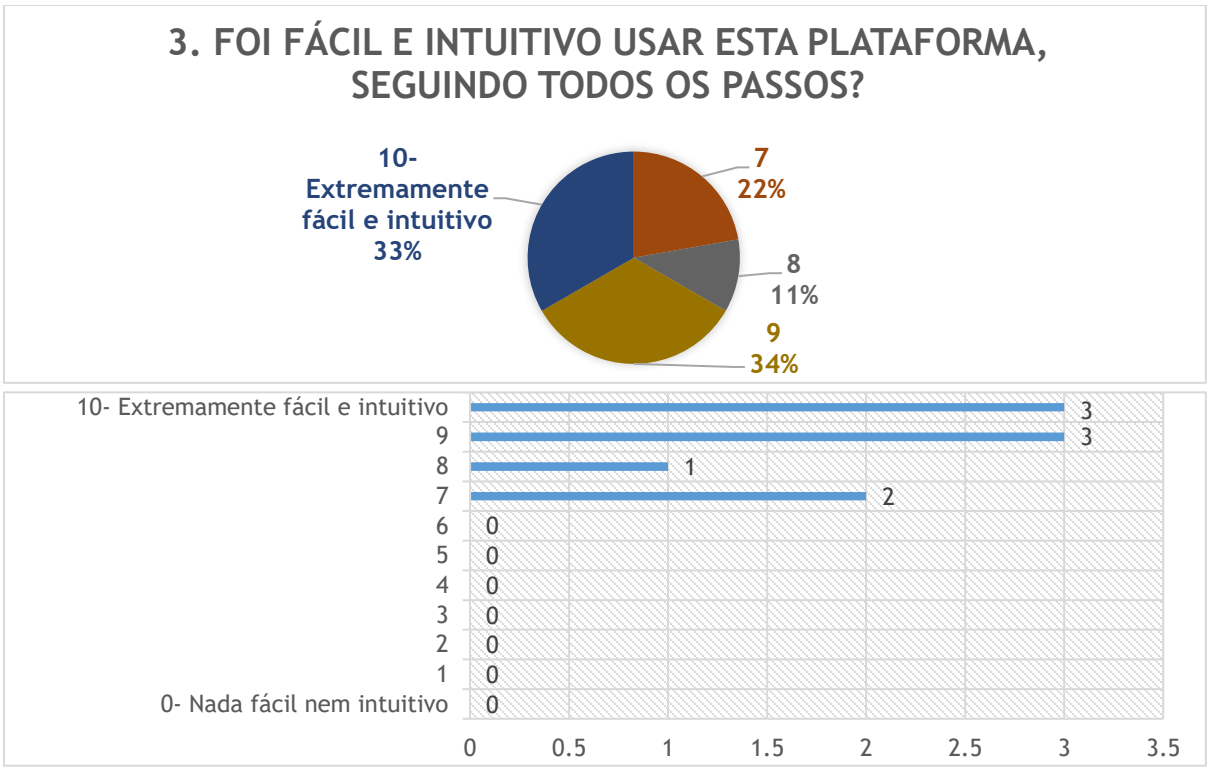


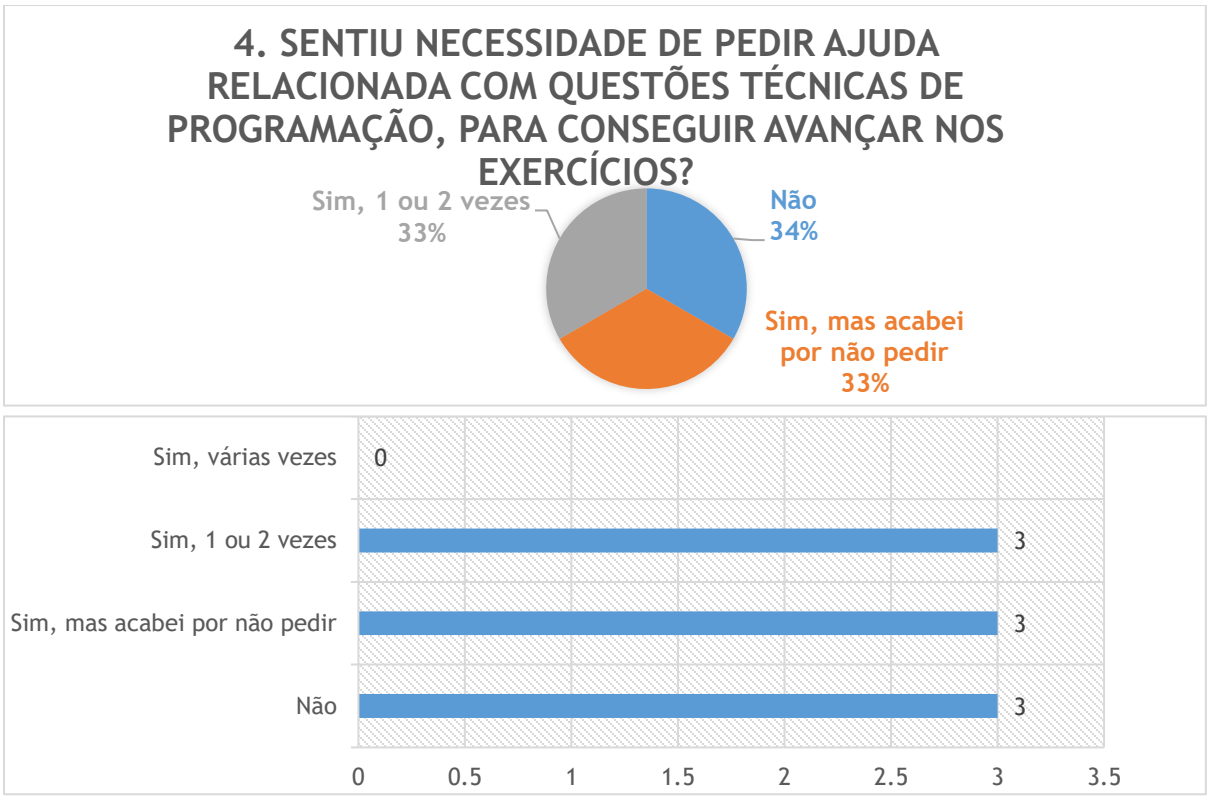


2. ACHA QUE AS INSTRUÇÕES DADAS POR ESTA PLATAFORMA ERAM COMPLETAS E CLARAS?



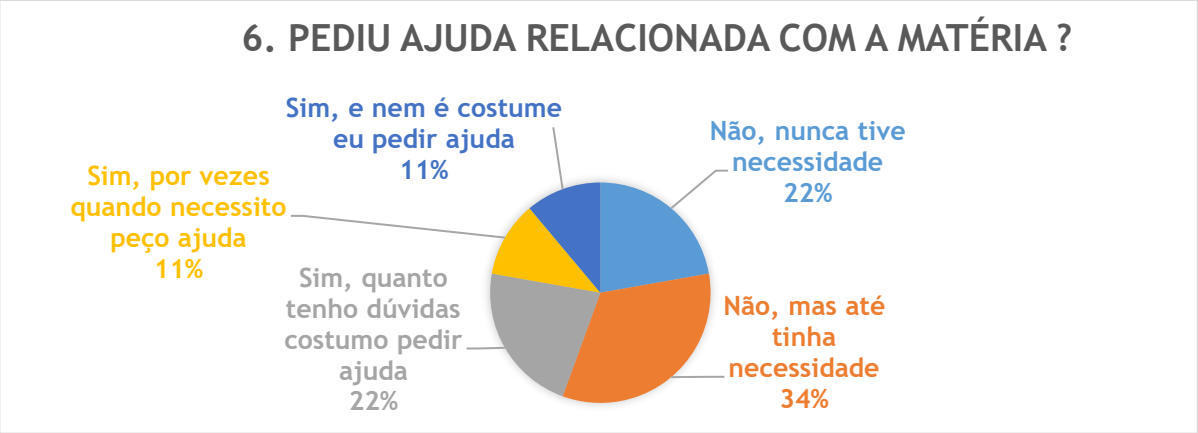


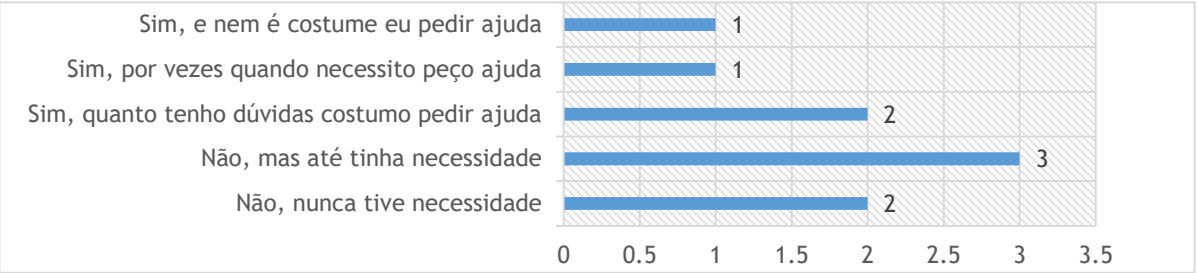

3. FOI FÁCIL E INTUITIVO USAR ESTA PLATAFORMA, SEGUINDO TODOS OS PASSOS?
10- Extremamente fácil e intuitivo 33%
7 22%
8 11%
9 34%
10- Extremamente fácil e intuitivo
9
8
7
6
5
4
3
2
1
0- Nada fácil nem intuitivo
3
3
1
2
0
0
0
0
0
0
0
0
0 0.5 1 1.5 2 2.5 3 3.5

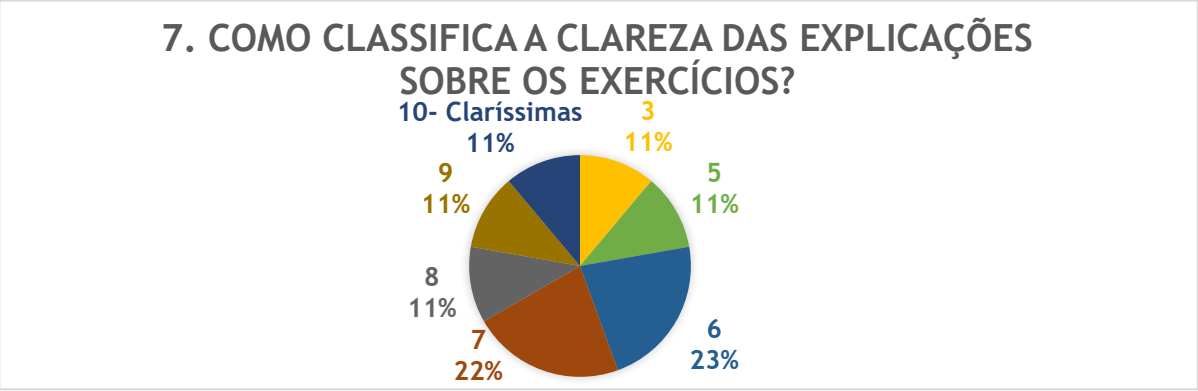

4. SENTIU NECESSIDADE DE PEDIR AJUDA RELACIONADA COM QUESTÕES TÉCNICAS DE PROGRAMAÇÃO, PARA CONSEGUIR AVANÇAR NOS EXERCÍCIOS?
Sim, 1 ou 2 vezes 33%
Não 34%
Sim, mas acabei por não pedir 33%
Sim, várias vezes
Sim, 1 ou 2 vezes
Sim, mas acabei por não pedir
Não
0
3
3
3
0 0.5 1 1.5 2 2.5 3 3.5

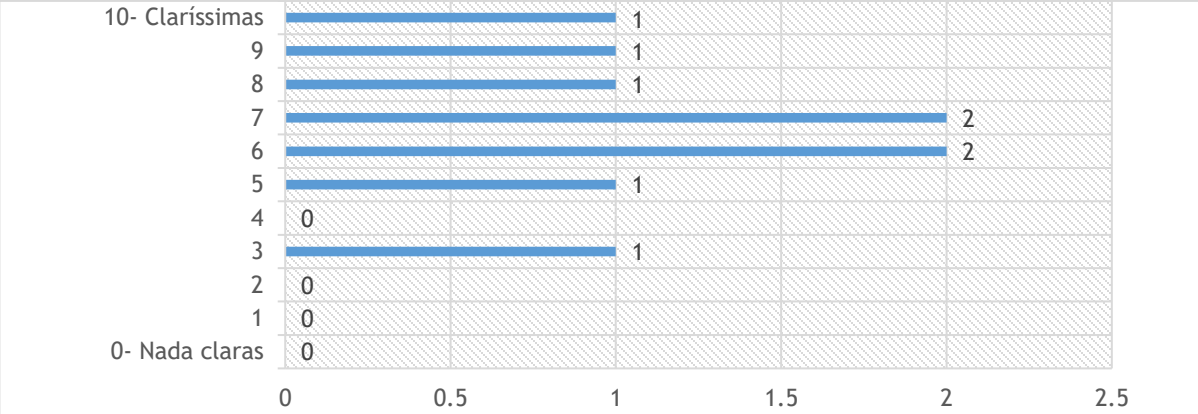
5. Quais foram as maiores dificuldades que teve?

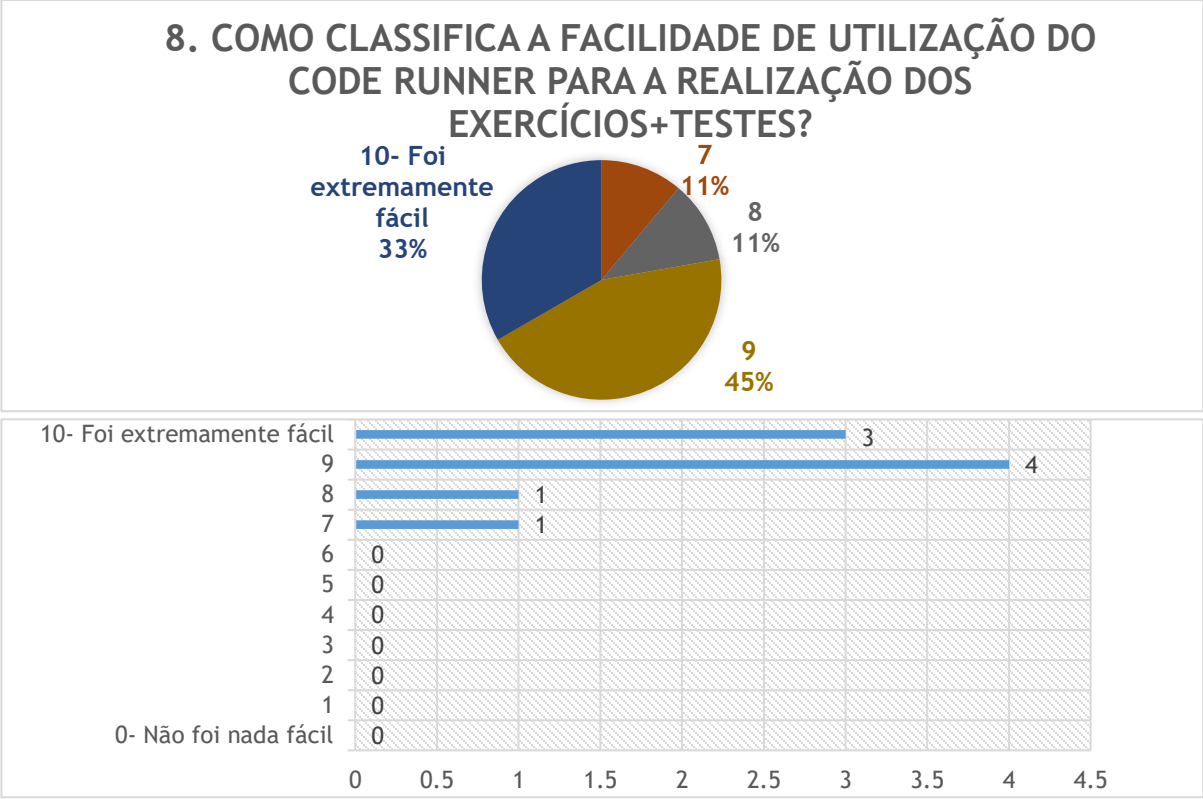
- O tipo de exercícios deixava-me muito confuso (completar código já feito e dificuldade na interpretação do enunciado). Não estava a conseguir progredir na matéria por isso decidi abandonar esta plataforma;
- As minhas dificuldades devem-se à falta de prática com a linguagem, no entanto houve também um período de adaptação ao *Code Runner* por minha parte dado que a minha forma de escrever código às vezes entra em conflito com o código pré-implementado. Por exemplo:
  Costumo usar:
  typedef struct ponto { } ponto2D;
  Em vez dos:
  typedef struct ponto ponto2D;
  struct ponto { };
- Escrever sintaxe;
- O facto de o enunciado e do *Code Runner* já com algumas coisas preenchidas estarem todos no mesmo sítio às vezes tornou que as coisas estivessem muito misturadas e confusas;
- Por falta de tempo, a experiência revelou-se curta;
- No geral na interpretação das perguntas;
- ; e * em sítios errados;
- Sinceramente, não tive nenhuma dificuldade elevada até onde usei;
- Às vezes na interpretação do enunciado, mas como não fiz muito não consegui solidificar a minha opinião.

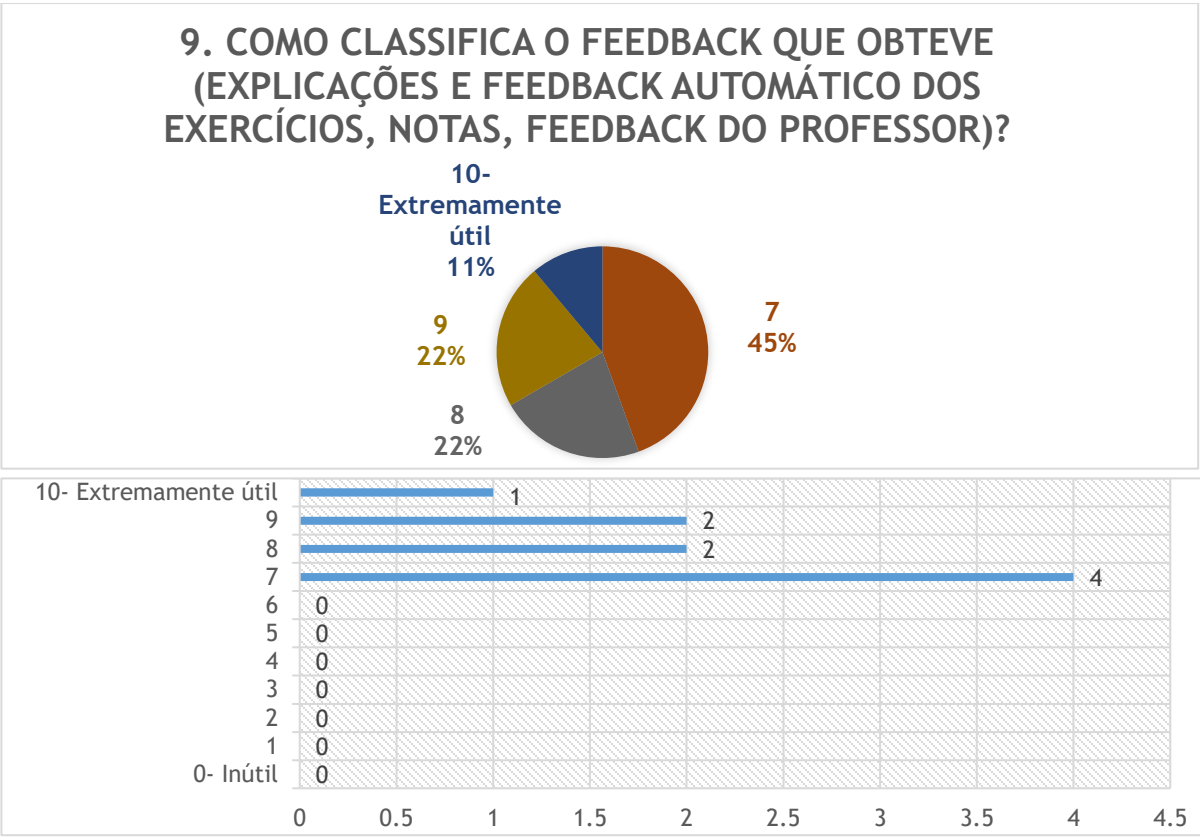

6. PEDIU AJUDA RELACIONADA COM A MATÉRIA ?
Sim, e nem é costume eu pedir ajuda 11%
Não, nunca tive necessidade 22%
Sim, por vezes quando necessito peço ajuda 11%
Sim, quanto tenho dúvidas costumo pedir ajuda 22%
Não, mas até tinha necessidade 34%

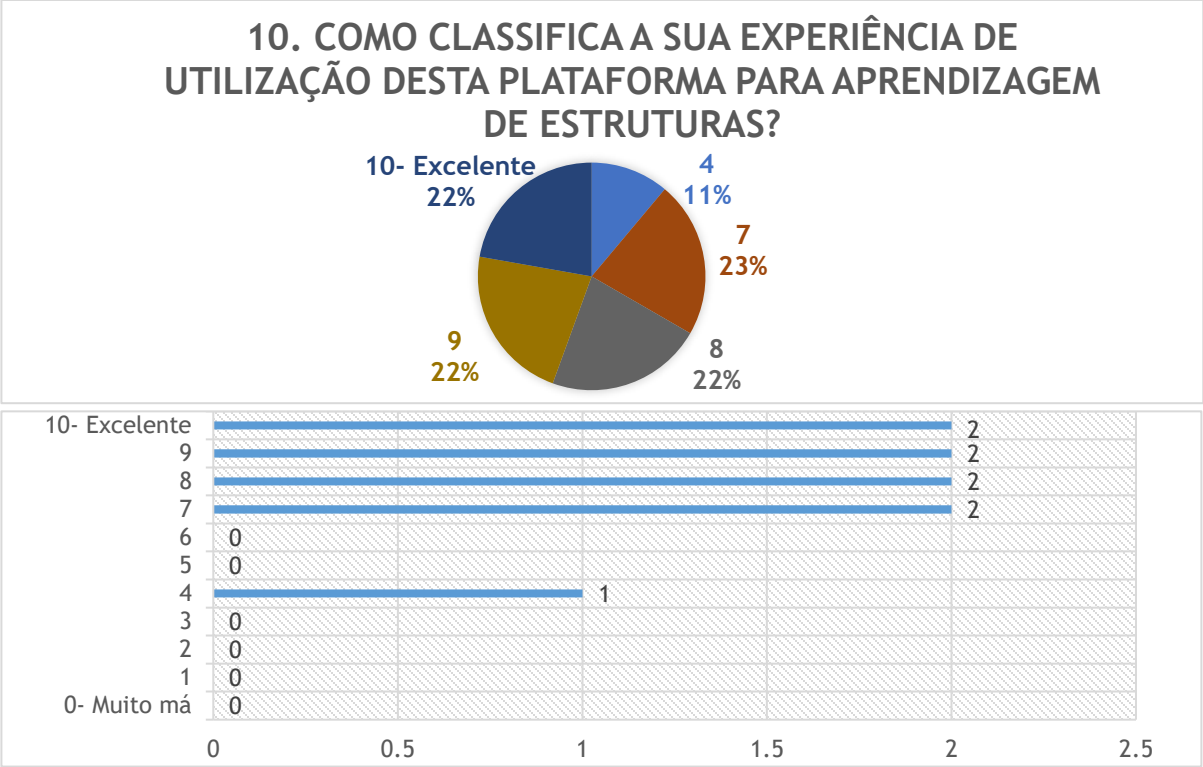

Sim, e nem é costume eu pedir ajuda 1
Sim, por vezes quando necessito peço ajuda 1
Sim, quanto tenho dúvidas costumo pedir ajuda 2
Não, mas até tinha necessidade 3
Não, nunca tive necessidade 2
0 0.5 1 1.5 2 2.5 3 3.5


7. COMO CLASSIFICA A CLAREZA DAS EXPLICAÇÕES SOBRE OS EXERCÍCIOS?
10- Claríssimas 11%
3 11%
5 11%
6 23%
7 22%
8 11%
9 11%


10- Claríssimas 1
9 1
8 1
7 2
6 2
5 1
4 0
3 1
2 0
1 0
0- Nada claras 0
0 0.5 1 1.5 2 2.5

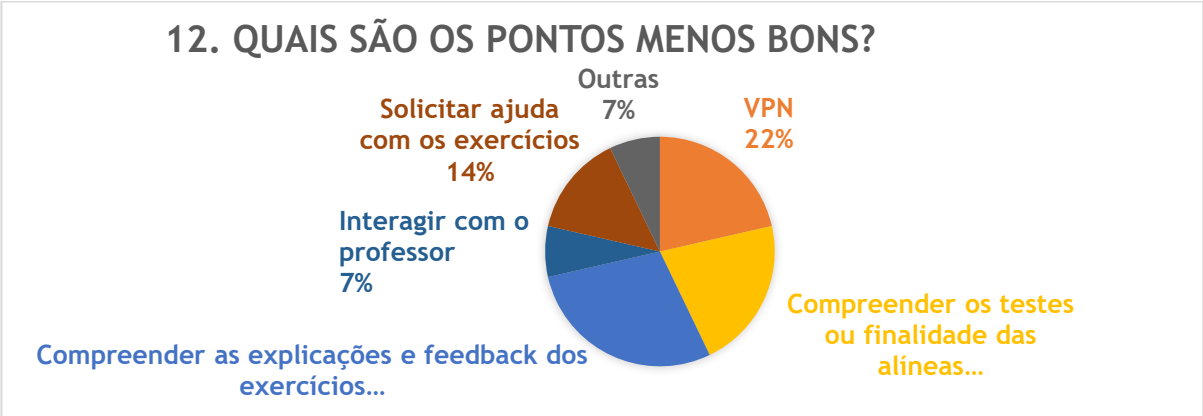

8. COMO CLASSIFICA A FACILIDADE DE UTILIZAÇÃO DO CODE RUNNER PARA A REALIZAÇÃO DOS EXERCÍCIOS+TESTES?
10- Foi extremamente fácil 33%
7 11%
8 11%
9 45%
10- Foi extremamente fácil
9
8
7
6
5
4
3
2
1
0- Não foi nada fácil
3
4
1
1
0
0
0
0
0
0
0
0 0.5 1 1.5 2 2.5 3 3.5 4 4.5

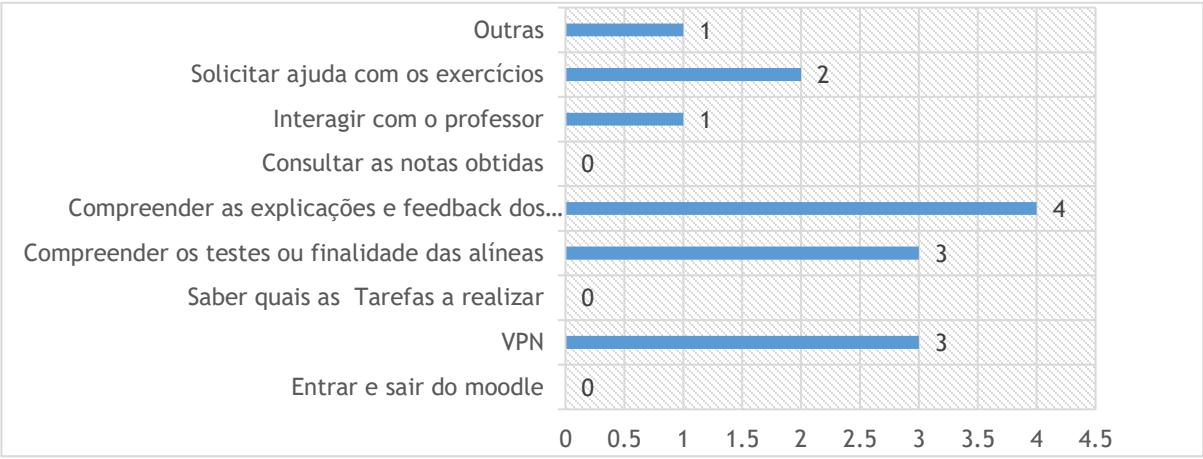

9. COMO CLASSIFICA O FEEDBACK QUE OBTEVE (EXPLICAÇÕES E FEEDBACK AUTOMÁTICO DOS EXERCÍCIOS, NOTAS, FEEDBACK DO PROFESSOR)?
10- Extremamente útil 11%
9 22%
8 22%
7 45%
10- Extremamente útil
9
8
7
6
5
4
3
2
1
0- Inútil
1
2
2
4
0
0
0
0
0
0
0
0 0.5 1 1.5 2 2.5 3 3.5 4 4.5



11. Quais são os pontos positivos desta experiência?

- Foi uma maneira de estudar divertida e que funcionou. Também gostei de ver o ranking, para quem é competitivo é mais uma razão para fazer os exercícios;
- Praticar. É realmente a chave para aprender;
- Experimentar a funcionalidade desta plataforma e ter feito alguns exercícios;
- Podemos praticar, com base regular a programação bem como a utilização do *Code Runner*. O facto de haver prazos e estarem delimitadas datas para fazer os exercícios penso que beneficia bastante os alunos;
- A facilidade ao acesso da matéria, a motivação para estudar devido ao placar de pontos e níveis;
- É uma boa ferramenta de estudo;
- Apesar do que respondi na alínea 5, o código pré-implementado é uma grande ajuda pois permite saber os passos que temos de seguir para atingir os resultados. A funcionalidade do *'Pre Check'* também é bastante interessante pois permite identificar os erros;
- Maior contacto com o código de programação;
- Temos soluções feitas por profissionais, além de que temos os exercícios divididos por passos, o que pode tornar o exercício mais fácil de resolver.





13. Tem sugestões de melhoria?

- Se existisse um "modo escuro" seria mais agradável para a vista.;
- Não;
- Honestamente não tenho. Não fiz muito porque preferi resolver diretamente pelos guiões. Tentarei ser mais ativo nas próximas tarefas;
- Acho o projeto interessante e sugiro continuidade;
- Completar código faz-me muita confusão. Acho que poderia haver exercícios mais "livres";
- No momento nenhuma;
- Não;
- Seria interessante aumentar a largura do bloco relativa ao *Syntaxe Errors*. Mesmo em erros pequenos, é obrigatório andar a deslocar a barra para conseguir ler tudo;
- Não tenho.

2.3. Perguntas da segunda parte do inquérito

1. Considera que a utilização da plataforma de apoio ao ensino contribuí-o para melhorar o seu conhecimento sobre estruturas de dados em C?

Respostas possíveis: 0- Não, nada; 1; 2; 3; 4; 5; 6; 7; 8; 9; 10- Sim, claramente.

2. Relativamente aos níveis de conhecimento que tem obtido nos anteriores tópicos de C, acha que os seus conhecimentos em Estruturas de Dados estão:

Respostas possíveis: 1.Muito piores; 2. Piores; 3. Semelhantes; 4. Melhores; 5. Muito melhores.

3. Aconselharia aos seus colegas a utilização desta plataforma para os próximos tópicos de C?

Respostas possíveis: 1.Não; 2.Talvez; 3.Sim.

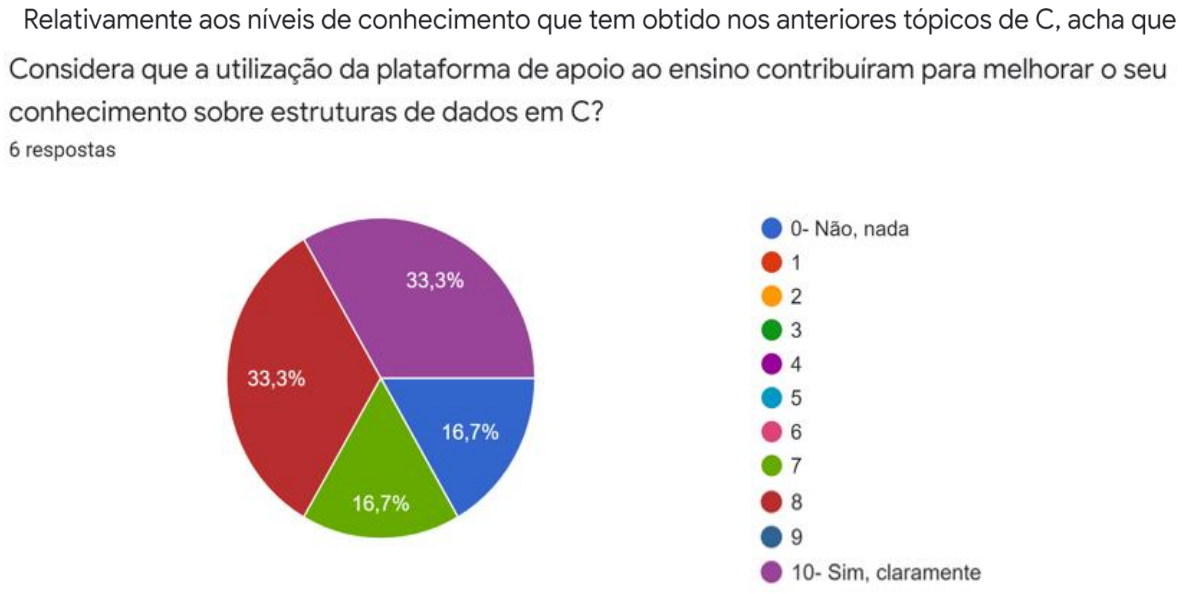
4. Gostaria de voltar a participar na próxima experiência, relacionada com Listas ligadas/Estruturas dinâmicas?

Respostas possíveis: Não; Talvez; Sim.

5. Existe algum ponto adicional que gostasse de realçar ou melhorar na experiência?

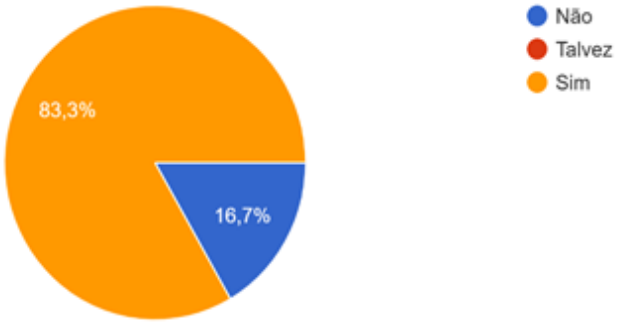
Respostas possíveis: Resposta curta.

2.4. Resultados da segunda parte do inquérito



Aconselharia aos seus colegas a utilização desta plataforma para os próximos tópicos de C?

6 respostas



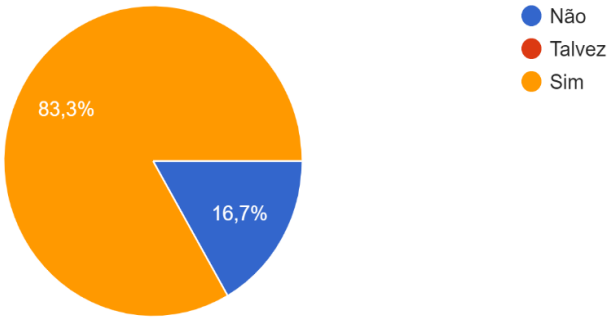
Existe algum ponto adicional que gostasse de realçar ou melhorar na experiência?

6 respostas

Não
Qualidade do material/perguntas muito muito mau: enunciados mal formatados com erros gramaticais e texto confuso/contraditório; poucos test cases que em média nem sequer metade da implementação necessária era testado; test cases simplesmente errados; exercícios com implementações repetidas desnecessariamente; soluções fornecidas simplesmente erradas e com qualidade de código péssimo. Muito muito mau...
Na.
Sim.
Acho que está tudo ótimo

Gostaria de voltar a participar na próxima experiência, relacionada com Listas ligadas/Estruturas dinâmicas ?

6 respostas



Anexo B: Plataforma *LMSOps* - fase 2

Nesta secção encontra-se descritas as questões e a totalidade dos resultados relativos aos inquéritos da segunda fase de experimentação da plataforma.

1. Inquérito final

Relativamente ao inquérito final realizado pelos alunos foi submetida uma resposta, conforme mostra a Figura 1.

Estatísticas dos inqueritos		
Submissões	Inquerito	Data prevista de finalização
1	Fase2 - Inquerito final	0

Figura 1 - Participação inquéritos

1.1. Questões

1. Realizou todo este programa/tarefas?
   Resposta possíveis: Realizei todos os passos cuidadosamente; Realizei todos os passos, mas a dada altura já não os fiz com atenção; Saltei alguns passos; Saltei todos ou quase todos os passos que podia até chegar ao fim; Não cheguei ao fim.

2. Acha que as instruções dadas por esta plataforma eram completas e claras?
   Respostas possíveis: 0- Nada claras e muito incompletas; 1; 2; 3; 4; 5; 6; 7; 8; 9; 10- Claríssimas e muito completas.

3. Foi fácil e intuitivo usar esta plataforma, seguindo todos os passos?
   Respostas possíveis: 0- Nada fácil nem intuitivo; 1; 2; 3; 4; 5; 6; 7; 8; 9; 10- Extremamente fácil e intuitivo.

4. Sentiu necessidade de pedir ajuda relacionada com questões técnicas de programação, para conseguir avançar nos exercícios?
   Respostas possíveis: Não; Sim, mas acabei por não pedir; Sim, 1 ou 2 vezes; Sim, várias vezes

5. Quais foram as maiores dificuldades que teve?
   Respostas possíveis: Resposta curta de desenvolvimento.

6. Pediu ajuda relacionada com a matéria?
   Respostas possíveis: Não, nunca tive necessidade; Não, mas até tinha necessidade; Sim, quanto tenho dúvidas costumo pedir ajuda; Sim, por vezes quando necessito peço ajuda; Sim, e nem é costume eu pedir ajuda.

7. Como classifica a clareza das explicações sobre os exercícios?

Respostas possíveis: 0- Nada claras; 1; 2; 3; 4; 5; 6; 7; 8; 9; 10- Claríssimas.

8. Como classifica a facilidade de utilização do *Code Runner* para a realização dos exercícios+testes?
   Respostas possíveis: 0- Não foi nada fácil; 1; 2; 3; 4; 5; 6; 7; 8; 9; 10- Foi extremamente fácil.

9. Como classifica o feedback que obteve (explicações e feedback automático dos exercícios, notas, feedback do professor)?
   Respostas possíveis: 0- Inútil; 1; 2; 3; 4; 5; 6; 7; 8; 9; 10- Extremamente útil.

10. Como classifica a sua experiência de utilização desta plataforma para aprendizagem de Estruturas?
    Respostas possíveis: 0- Muito má; 1; 2; 3; 4; 5; 6; 7; 8; 9; 10- Excelente.

11. Quais são os pontos positivos desta experiência?
    Respostas possíveis: Resposta curta de desenvolvimento.

12. Quais são os pontos menos bons?
    Respostas possíveis: Entrar e sair do moodle; VPN; Saber quais as Tarefas a realizar; Compreender os testes ou finalidade das alíneas; Compreender as explicações e feedback dos exercícios; Consultar as notas obtidas; Interagir com o professor; Solicitar ajuda com os exercícios; Outras.

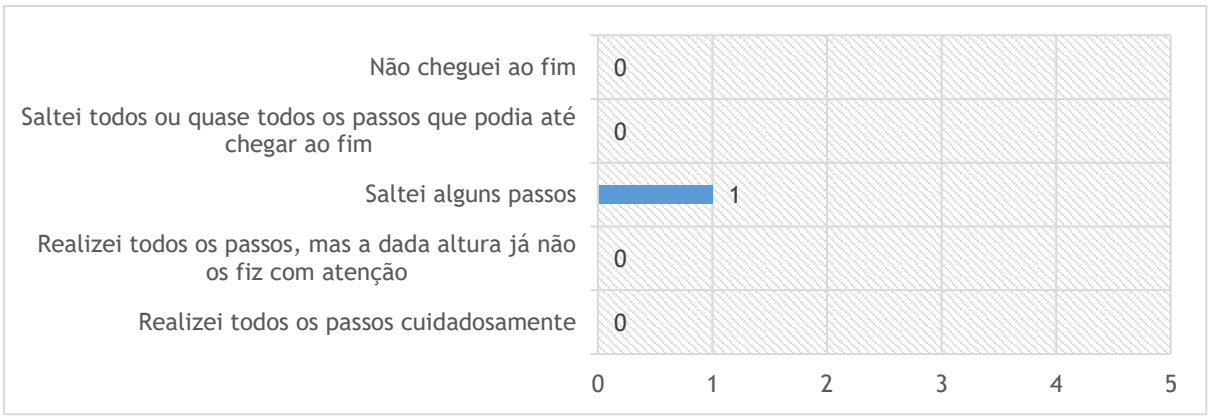
13. Tem sugestões de melhoria?
    Respostas possíveis: Resposta curta.

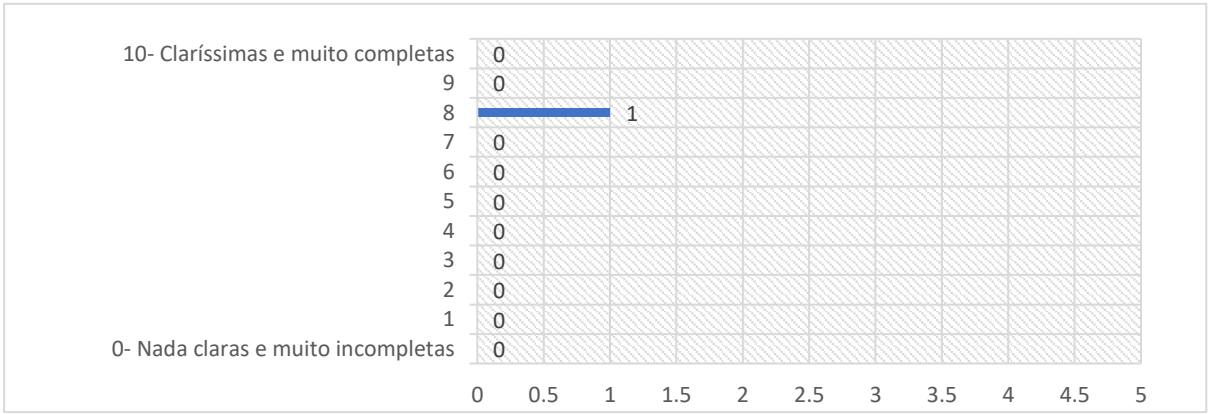
14. Foi mais fácil e intuitivo usar a plataforma nesta segunda fase, seguir os passos e acompanhar as tarefas?
    Respostas possíveis: 0- Menos fácil e intuitivo; 1; 2; 3; 4; 5- Equivalente à primeira fase; 6; 7; 8; 9; 10- Mais fácil e intuitivo.

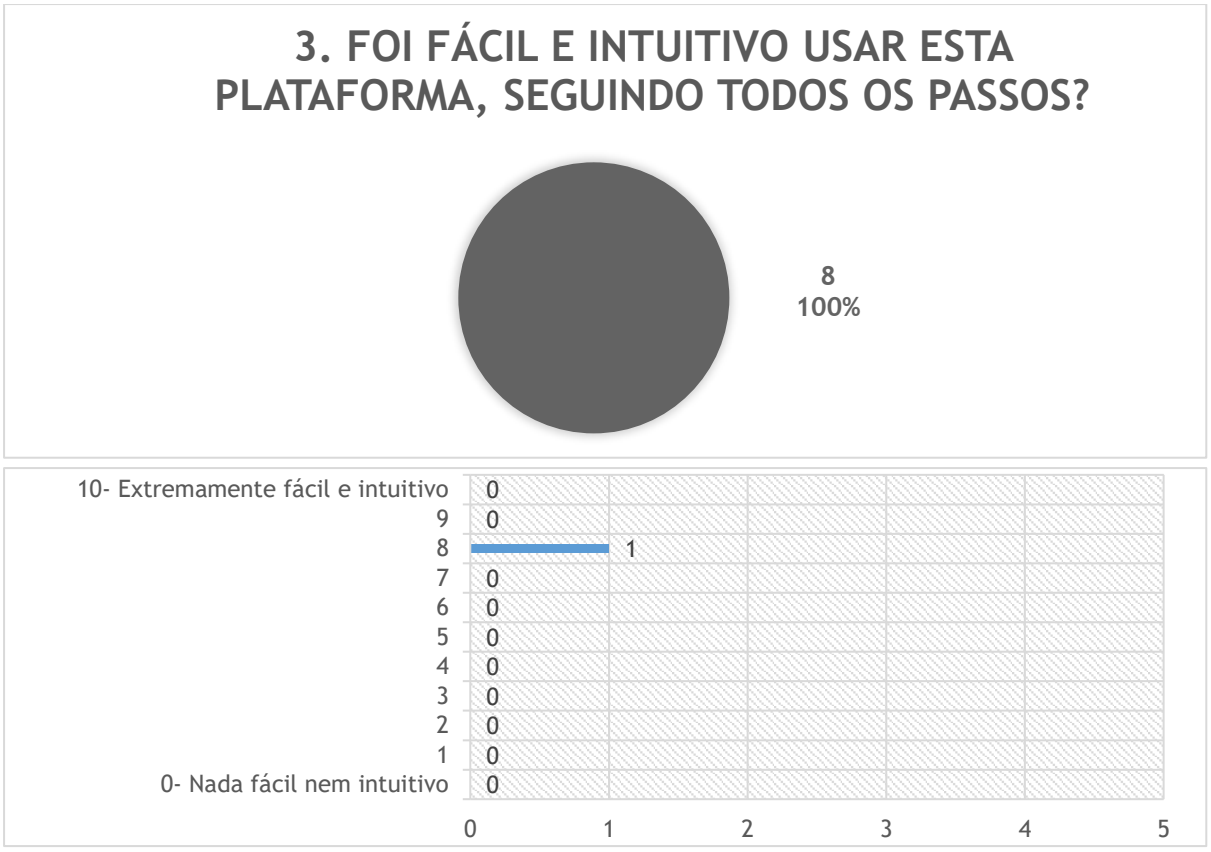
15. Acha que plataforma melhorou em relação à primeira fase? Existe alguma nota final que gostaria de deixar em relação à plataforma, às ideias ou a outros aspetos da experiência?
    Respostas possíveis: Resposta curta.

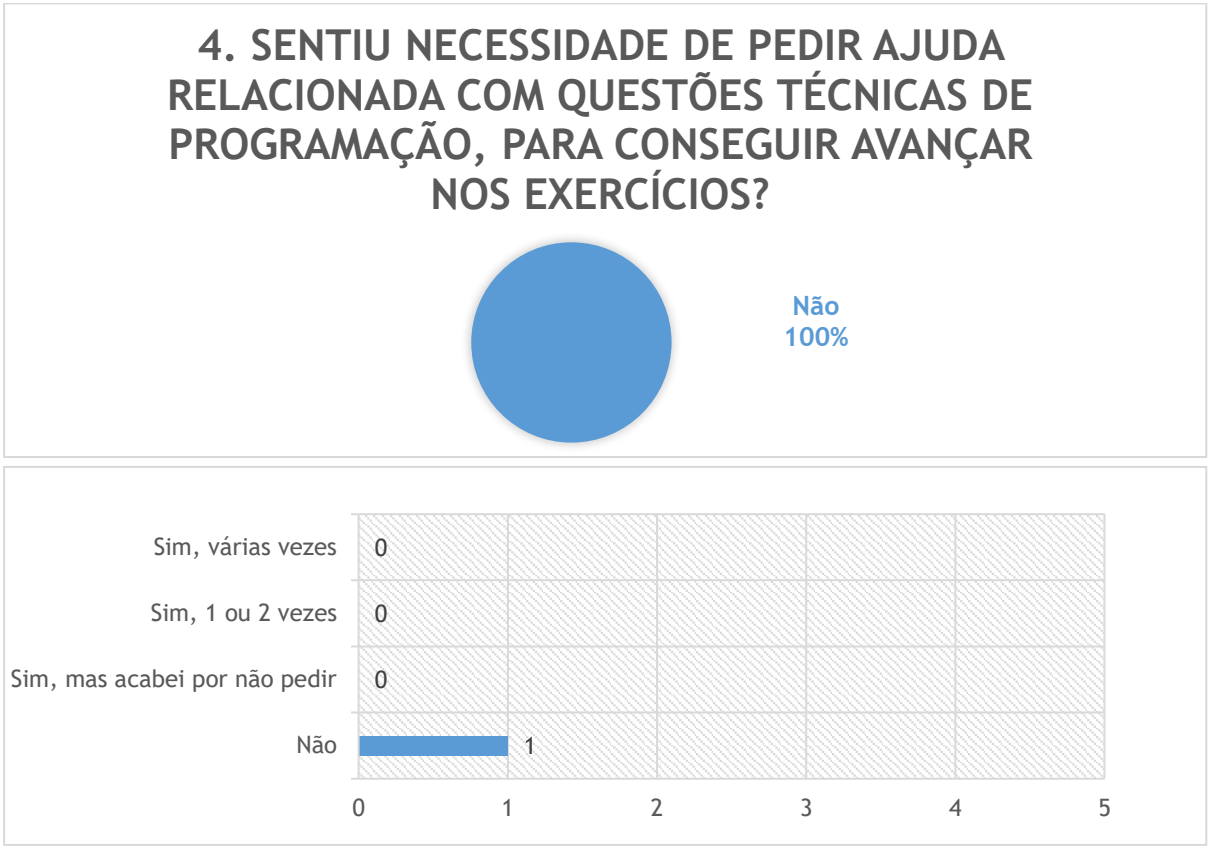
16. Existe mais algum ponto positivo ou negativo ou comentário final, relativo a toda a experiência que gostasse de realçar?
    Respostas possíveis: Resposta curta.

1.2. Resultados



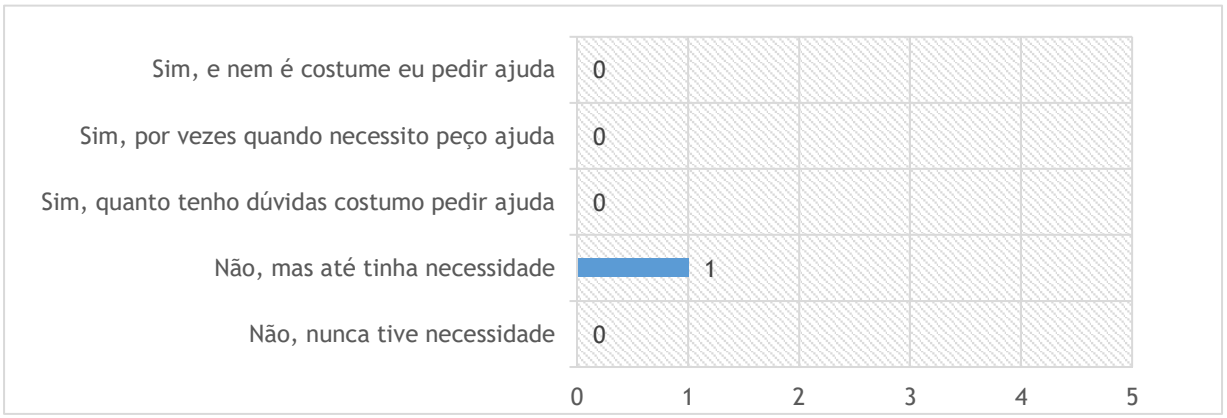


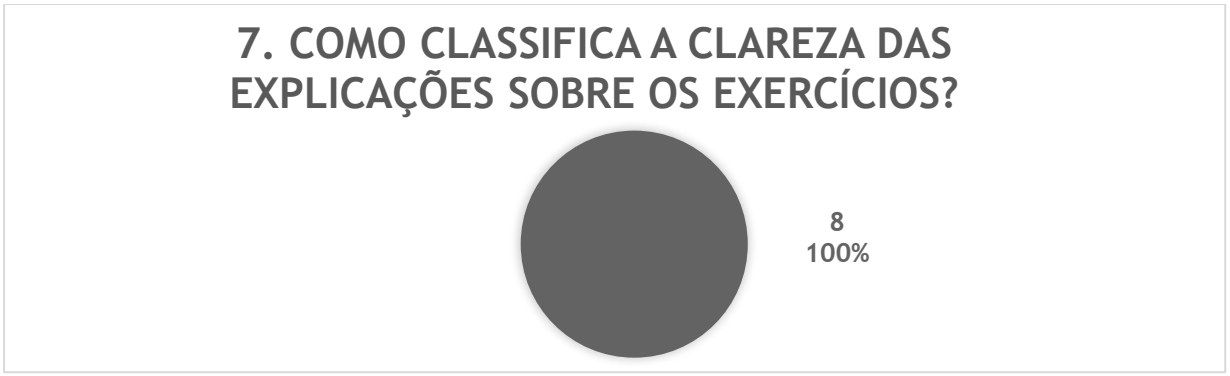

3. FOI FÁCIL E INTUITIVO USAR ESTA PLATAFORMA, SEGUINDO TODOS OS PASSOS?
8
100%
10- Extremamente fácil e intuitivo
9
8
7
6
5
4
3
2
1
0- Nada fácil nem intuitivo
0
0
1
0
0
0
0
0
0
0
0
0 1 2 3 4 5

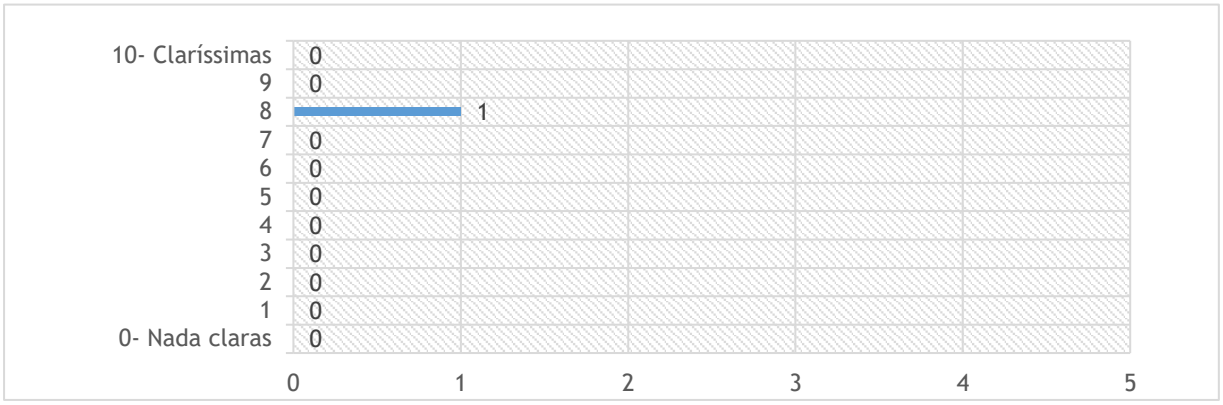

4. SENTIU NECESSIDADE DE PEDIR AJUDA RELACIONADA COM QUESTÕES TÉCNICAS DE PROGRAMAÇÃO, PARA CONSEGUIR AVANÇAR NOS EXERCÍCIOS?
Não
100%
Sim, várias vezes
Sim, 1 ou 2 vezes
Sim, mas acabei por não pedir
Não
0
0
0
1
0 1 2 3 4 5

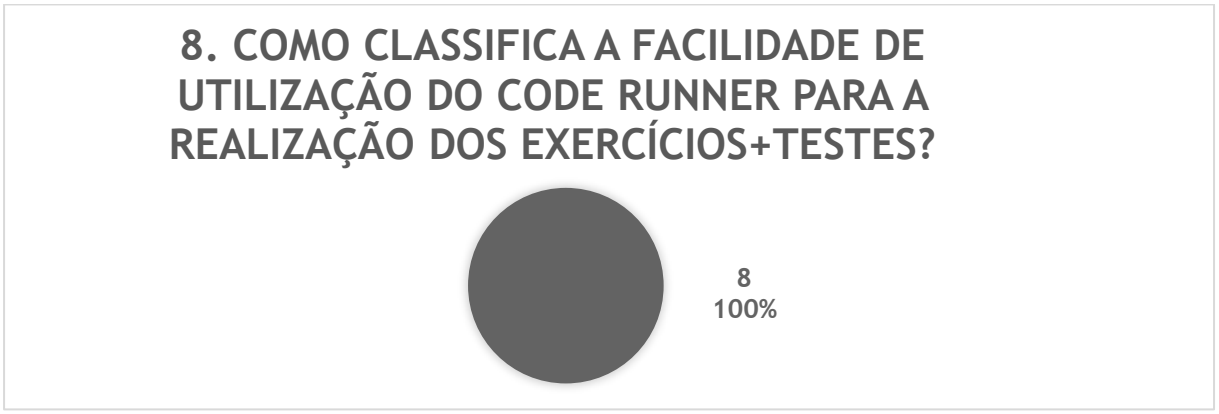
5. Quais foram as maiores dificuldades que teve?

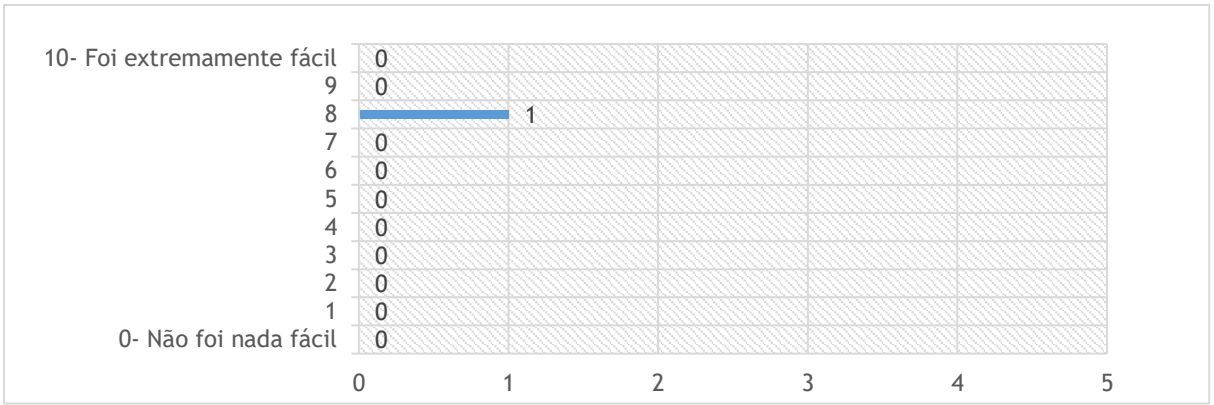
Básicas e ultrapassáveis.

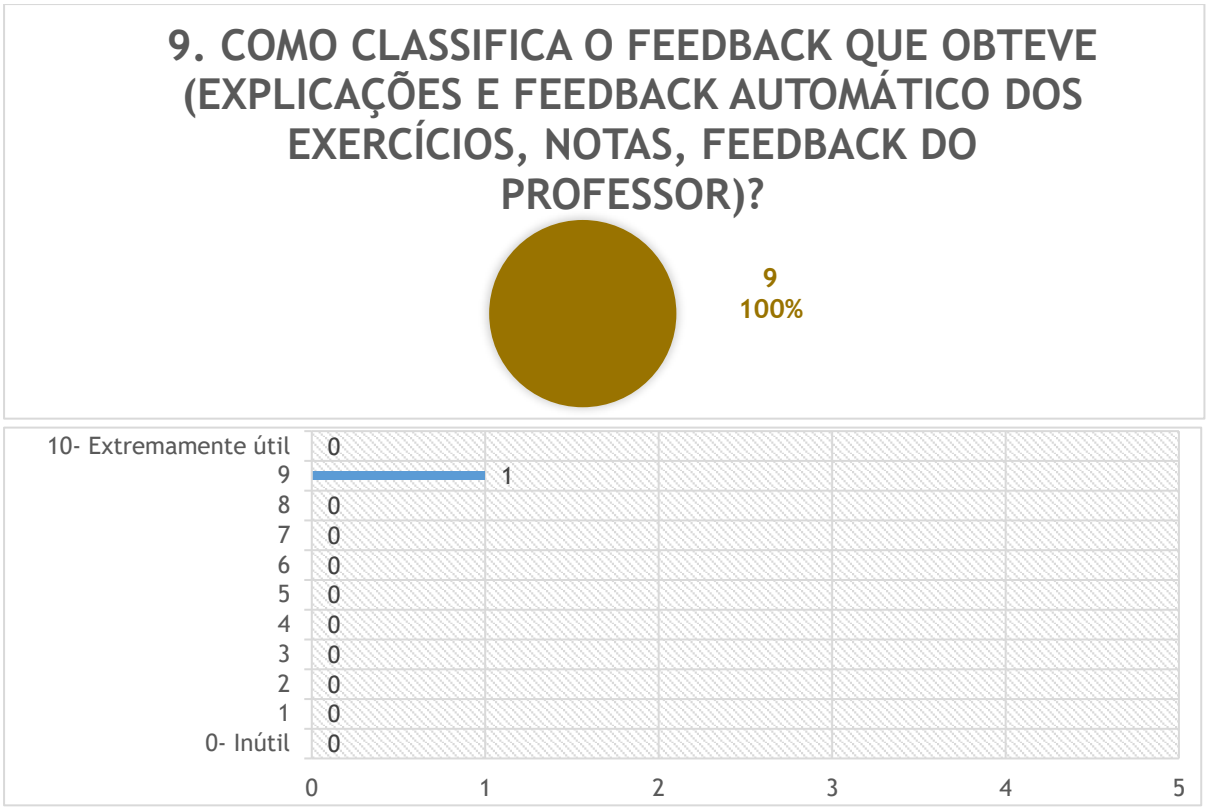



7. COMO CLASSIFICA A CLAREZA DAS EXPLICAÇÕES SOBRE OS EXERCÍCIOS?
8
100%


10- Claríssimas
9
8
7
6
5
4
3
2
1
0- Nada claras
0
1
2
3
4
5

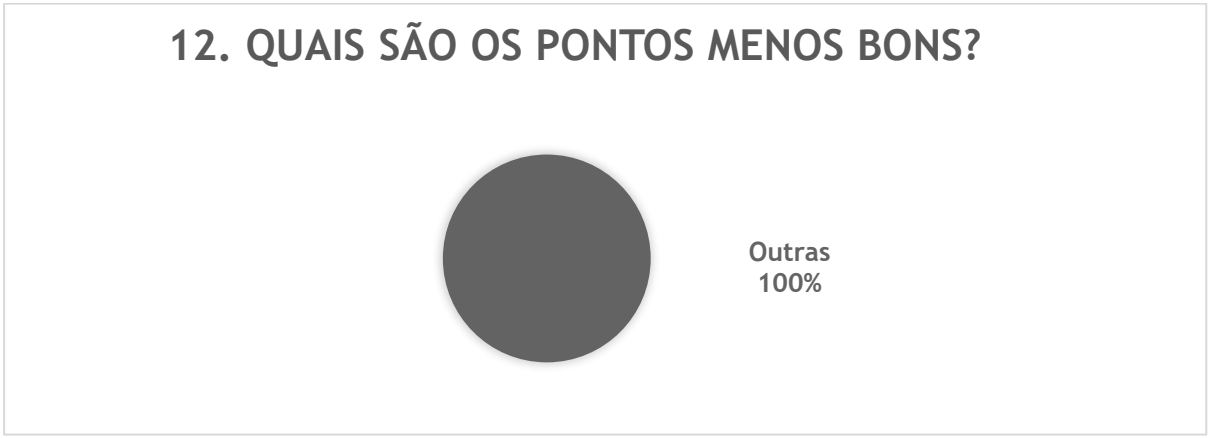

8. COMO CLASSIFICA A FACILIDADE DE UTILIZAÇÃO DO CODE RUNNER PARA A REALIZAÇÃO DOS EXERCÍCIOS+TESTES?
8
100%

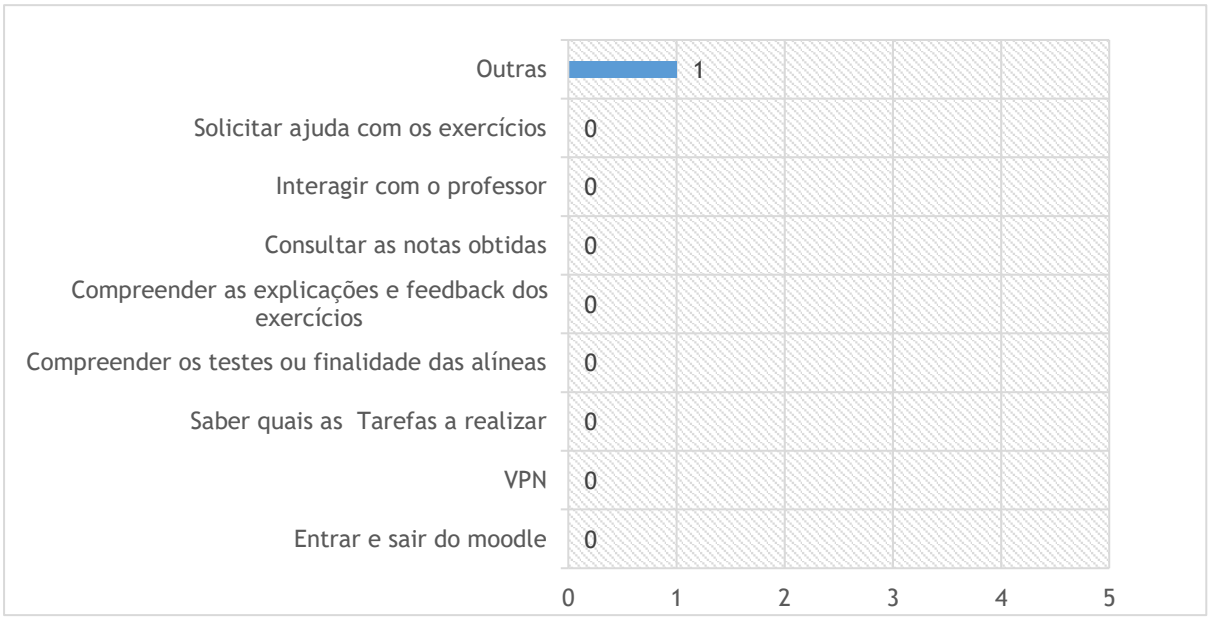

10- Foi extremamente fácil
9
8
7
6
5
4
3
2
1
0- Não foi nada fácil
0
1
2
3
4
5



11. Quais são os pontos positivos desta experiência?

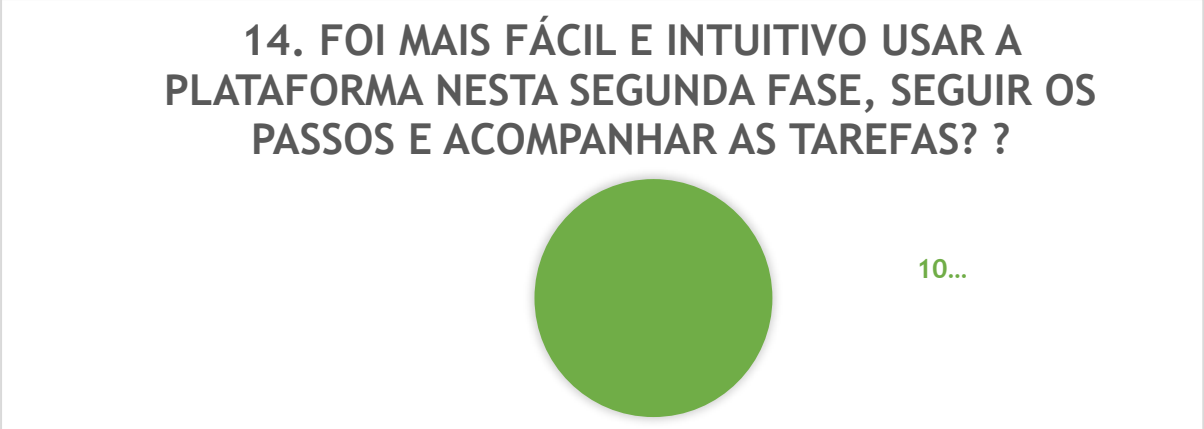
- Programar.

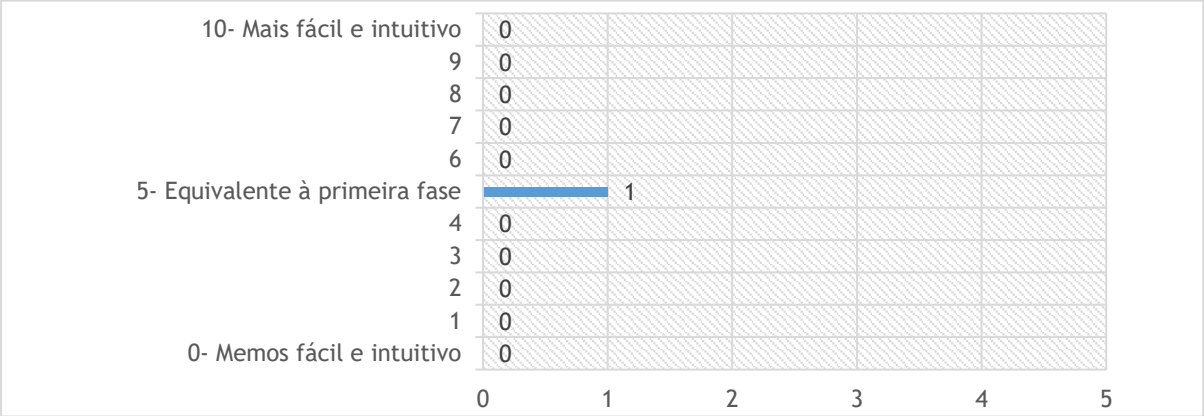




13. Tem sugestões de melhoria?

- Acho que o projeto ajuda a incentivar ao desafio de programar, deve continuar.





15. Acha que plataforma melhorou em relação à primeira fase? Existe alguma nota final que gostaria de deixar em relação à plataforma, às ideias ou a outros aspetos da experiência?

- Na.

16. Existe mais algum ponto positivo ou negativo ou comentário final, relativo a toda a experiência que gostasse de realçar?

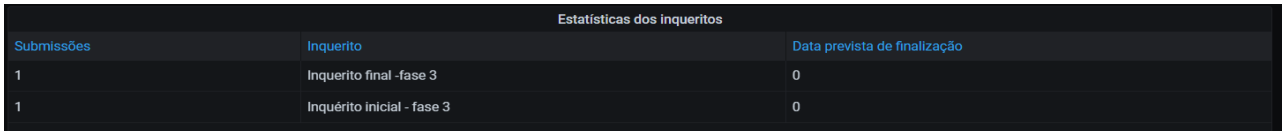
- Tudo o que possa envolver os alunos é sempre positivo.

Anexo C: Plataforma LMSOps - fase 3

Nesta secção descritas as questões e a totalidade dos resultados relativos aos inquéritos da terceira fase de experimentação da plataforma.

1. Inquérito Inicial

Relativamente ao inquérito inicial realizado aos alunos foi submetida 1 resposta, conforme mostra a Figura 1.



Estatísticas dos inqueritos

Submissões	Inquerito	Data prevista de finalização
1	Inquerito final -fase 3	0
1	Inquérito inicial - fase 3	0

Figura 1 - Participação inquéritos

1.1. Questões

1. Há quanto tempo aprendeu a programar?
Respostas possíveis: Este ano, 1-2 anos, 3-4 anos, >4 anos

2. Numa escala de 0-10 quanto gosta de programar?
Respostas possíveis: Não consigo saber se gosto ou não, 0-4 - Detesto programar / Não gosto particularmente de programar, 5-7 - Não gosto nem desgosto, 8-10 - Adoro programar

3. Já realizou a cadeira de introdução à programação? Se sim, em que escala se enquadra a sua nota?
Respostas possíveis: Não fiz avaliação, <8, 8 – 9, 10 – 12, 13 – 15, 16 – 18, 19 - 20

4. Se estiver a repetir Programação, que nota teve antes?
Respostas possíveis: Esta é a minha primeira matrícula, Não é a minha primeira matrícula, mas não frequentei/não fui avaliado a Programação, <8, 8 – 9, 10 – 12, 13 – 15, 16 – 18, 19 - 20

5. Quais são as maiores dificuldades que tem sentido em Programação (pode assinalar várias)?
Respostas possíveis: - Sintaxe da linguagem; - Compreender os Problemas; - Desenvolver os Algoritmos; - Turma muito lotada ou dificuldade em manter a concentração ou muito barulho; - Dificuldade em acompanhar o ritmo da aula ou pouco

acompanhamento por parte do professor; - Codificar; - Testar; - Debug; - Outras...

6. Se anteriormente respondeu "Outras", pode indicar quais?
Respostas possíveis: Resposta curta;

7. Quais os aspetos que considera mais o ajudarem a aprender programação (pode assinalar várias)?
Respostas possíveis: - Estudar cuidadosamente e compreender as bases teóricas; - Realizar muitos exercícios; - As explicações dadas pelos professores das aulas teóricas; - O apoio dos professores das aulas laboratoriais; - Trabalhar em grupo e o apoio dos colegas; - Os fóruns de discussão ou outras plataformas (ex: - *stackoverflow.com*, *youtube.com*, *udemy.com*); - Utilização de uma ferramenta ou plataforma (ex. *CodeRunner*) com testes pré-preparados e feedback imediato; - Outros...

8. Se anteriormente respondeu "Outros", pode indicar quais?

Respostas possíveis: Resposta curta;

9. Quanto tempo costuma dedicar, semanalmente, ao estudo de programação?

Respostas possíveis: <30min; 30min-1h; 1-2h; 2-3h; >3h;

10. Quanto tempo costuma dedicar, semanalmente, na realização de exercícios de programação? Respostas possíveis: <30min; 30min-1h; 1-2h; 2-3h; >3h;

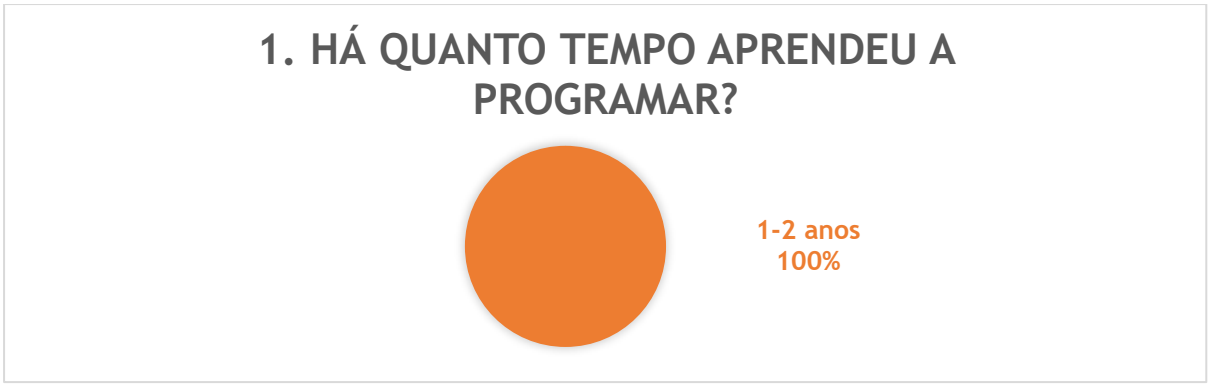
11. Costuma frequentar as aulas teóricas?
Respostas possíveis: Não costumo frequentar; <25%; 25-50%; 50-75%; >75%;

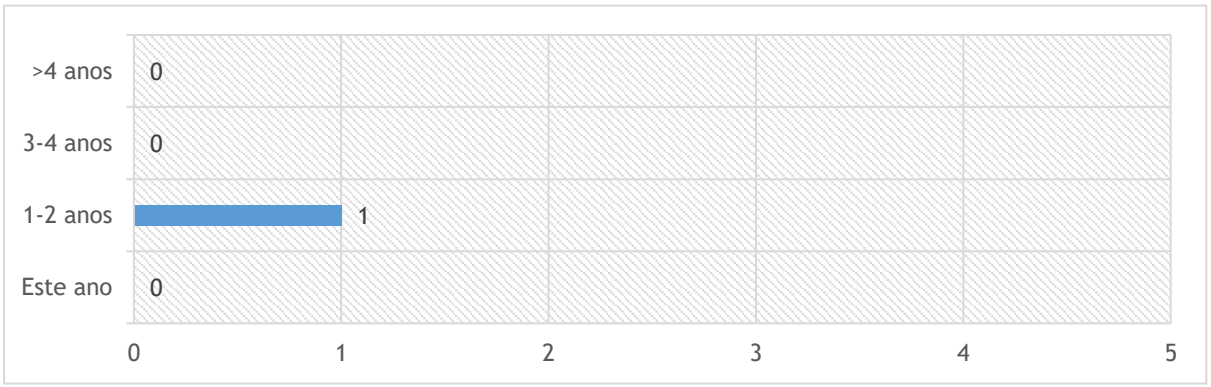
12. Costuma frequentar as aulas laboratoriais?
Respostas possíveis: Não costumo frequentar; <25%; 25-50%; 50-75%; >75%;

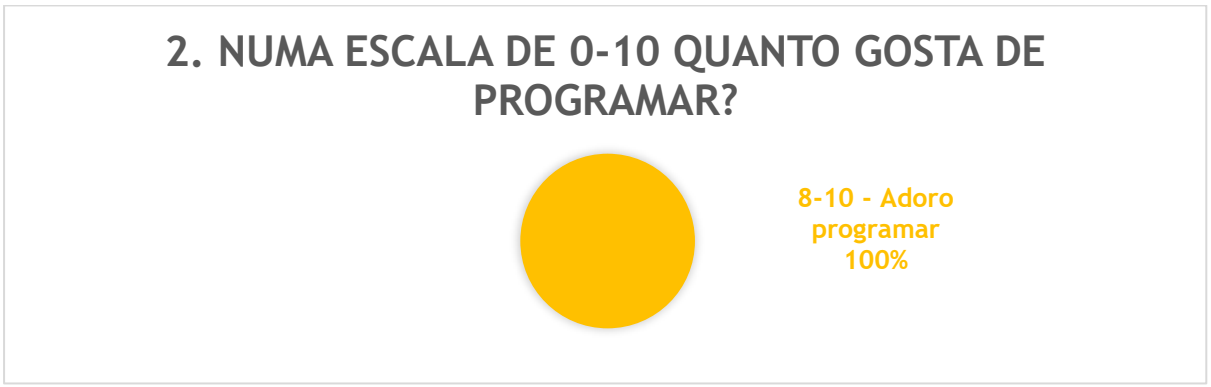
13. Quais são os aspetos mais positivos no ensino-aprendizagem de programação no ISEC?
Respostas possíveis: Resposta curta;

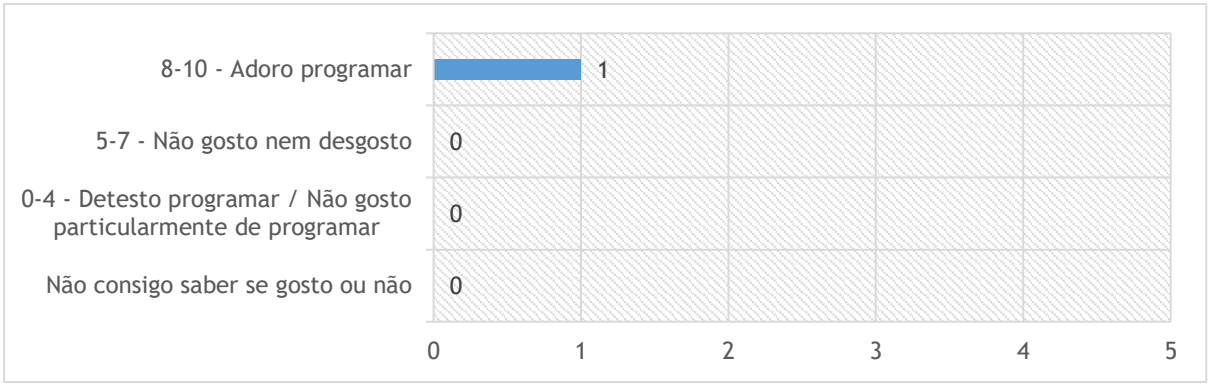
14. O que considera que deveria mudar no ensino da programação no ISEC de modo a melhorar a sua aprendizagem?
Respostas possíveis: Resposta curta;

1.2. Resultados

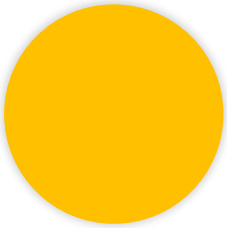

1. HÁ QUANTO TEMPO APRENDEU A PROGRAMAR?
1-2 anos
100%


>4 anos
0
3-4 anos
0
1-2 anos
1
Este ano
0
0
1
2
3
4
5

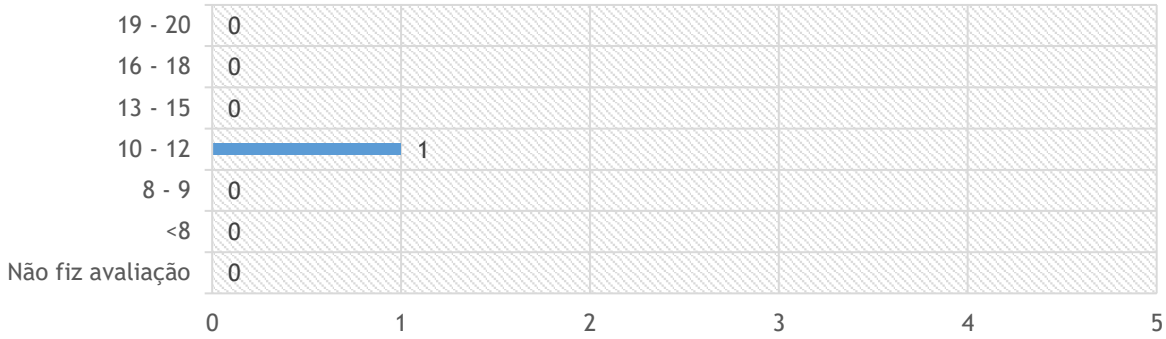

2. NUMA ESCALA DE 0-10 QUANTO GOSTA DE PROGRAMAR?
8-10 - Adoro programar
100%


8-10 - Adoro programar
1
5-7 - Não gosto nem desgosto
0
0-4 - Detesto programar / Não gosto particularmente de programar
0
Não consigo saber se gosto ou não
0
0
1
2
3
4
5

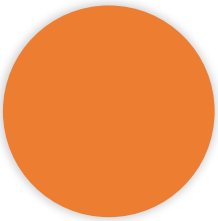
3. JÁ REALIZOU A CADEIRA DE INTRODUÇÃO À PROGRAMAÇÃO? SE SIM, EM QUE ESCALA SE ENQUADRA A SUA NOTA?



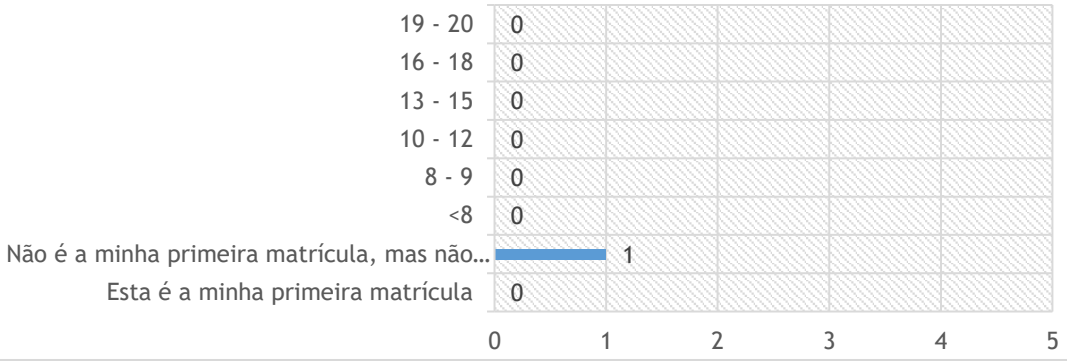
10 - 12
100%

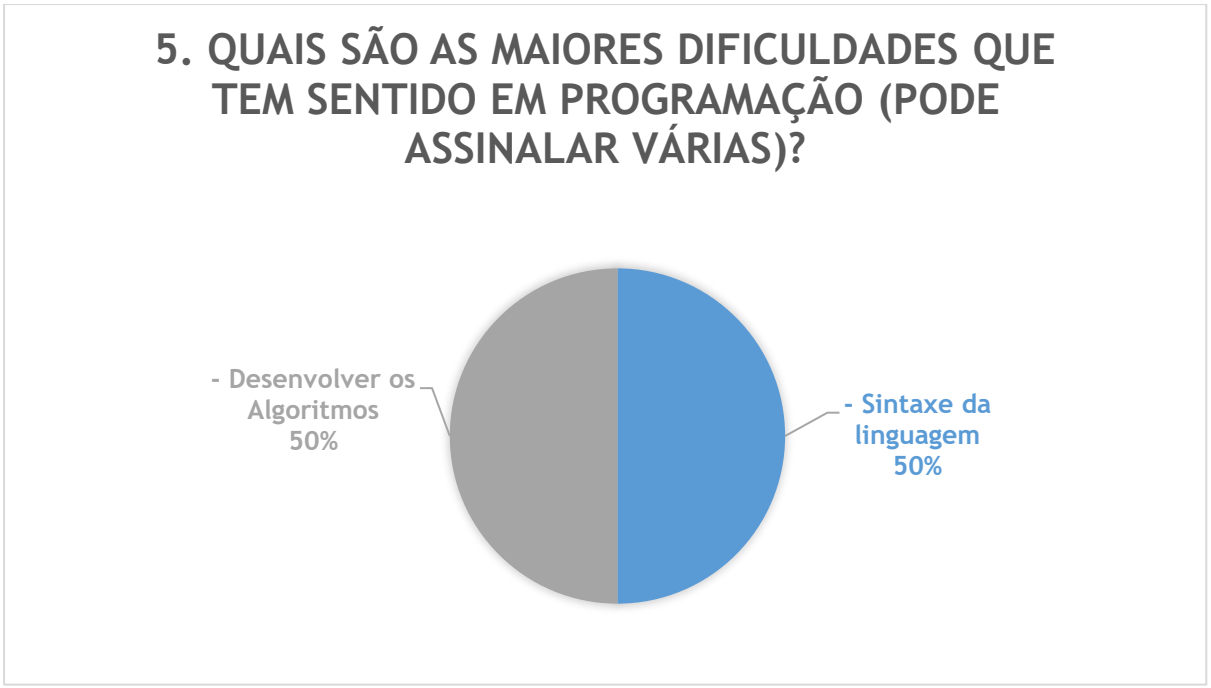


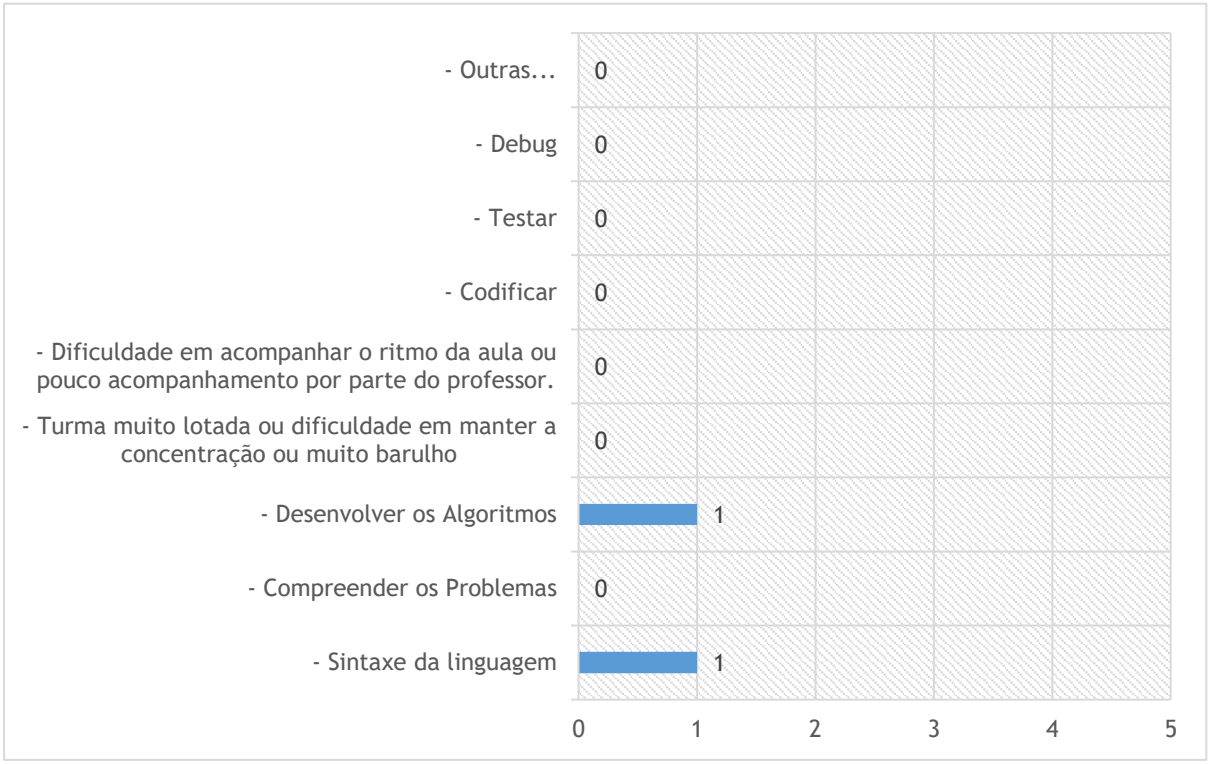
4. SE ESTIVER A REPETIR PROGRAMAÇÃO, QUE NOTA TEVE ANTES?

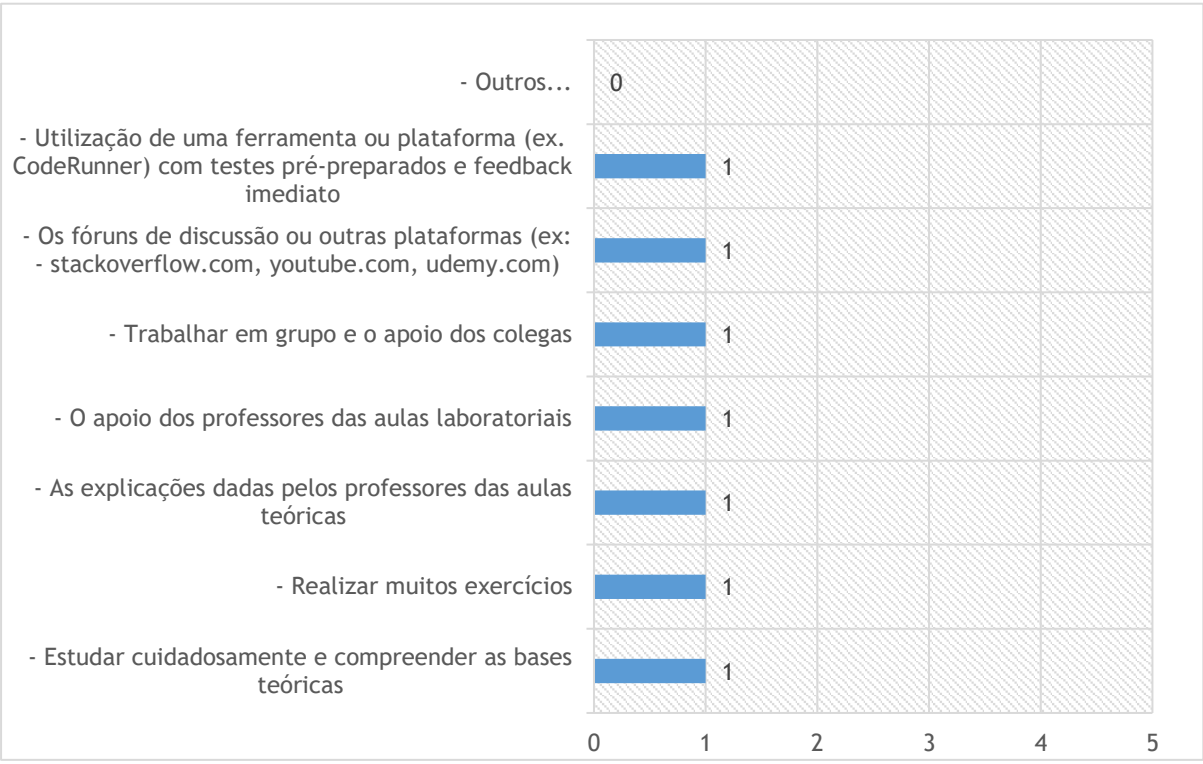


Não é a minha primeira matrícula, mas não frequentei/não fui avaliado a Programação
100%



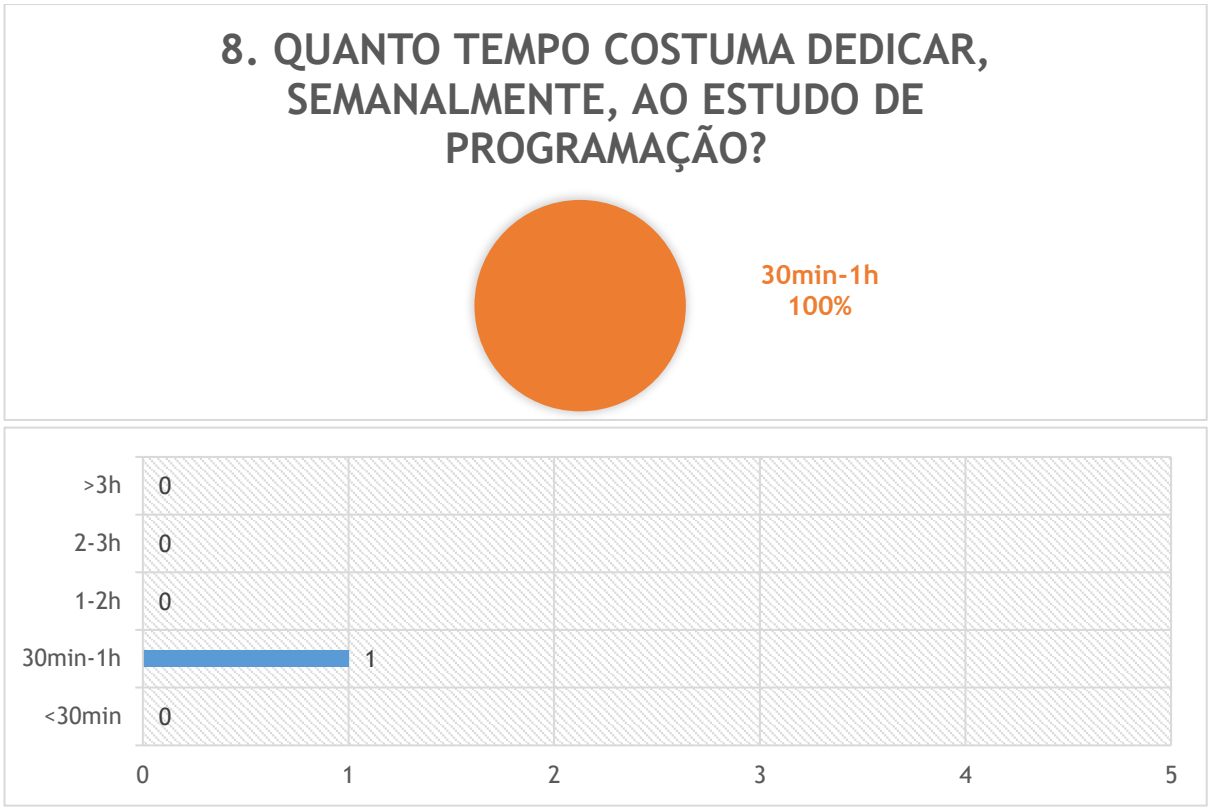

5. QUAIS SÃO AS MAIORES DIFICULDADES QUE TEM SENTIDO EM PROGRAMAÇÃO (PODE ASSINALAR VÁRIAS)?
- Desenvolver os Algoritmos 50%
- Sintaxe da linguagem 50%

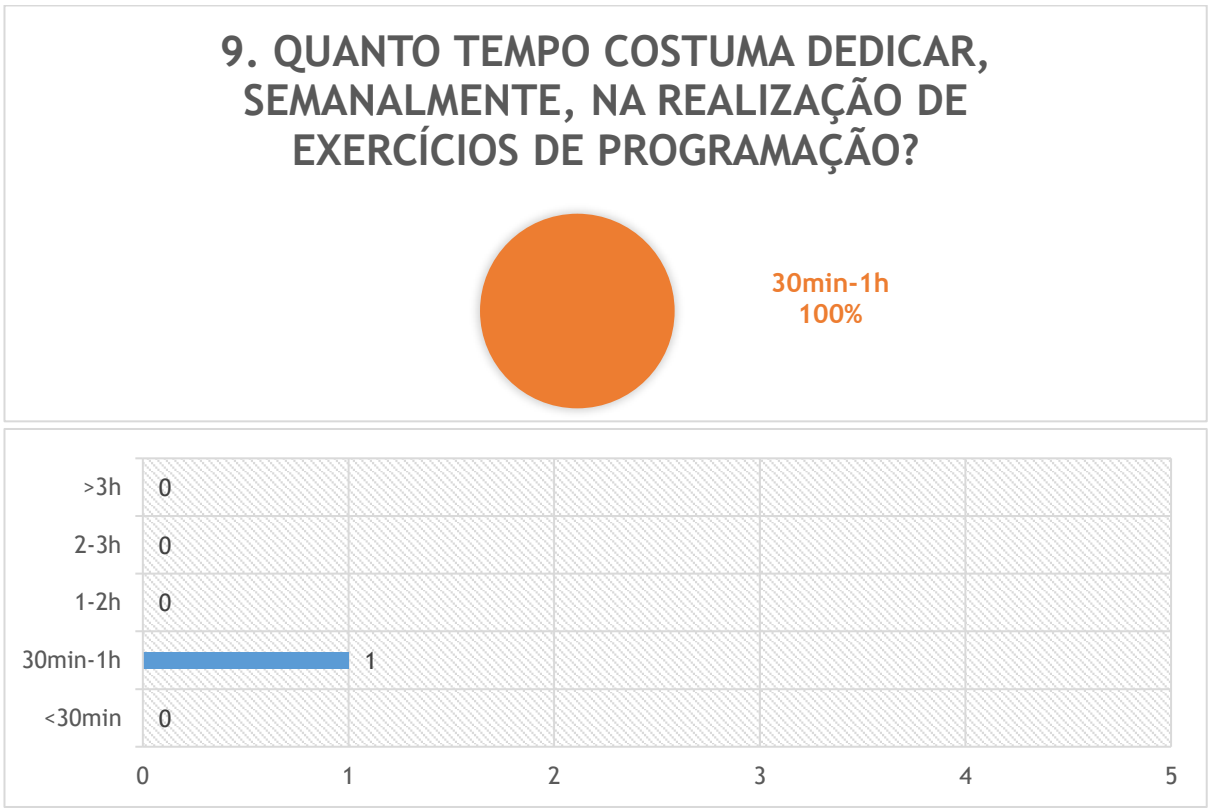

- Outras... 0
- Debug 0
- Testar 0
- Codificar 0
- Dificuldade em acompanhar o ritmo da aula ou pouco acompanhamento por parte do professor. 0
- Turma muito lotada ou dificuldade em manter a concentração ou muito barulho 0
- Desenvolver os Algoritmos 1
- Compreender os Problemas 0
- Sintaxe da linguagem 1
0 1 2 3 4 5

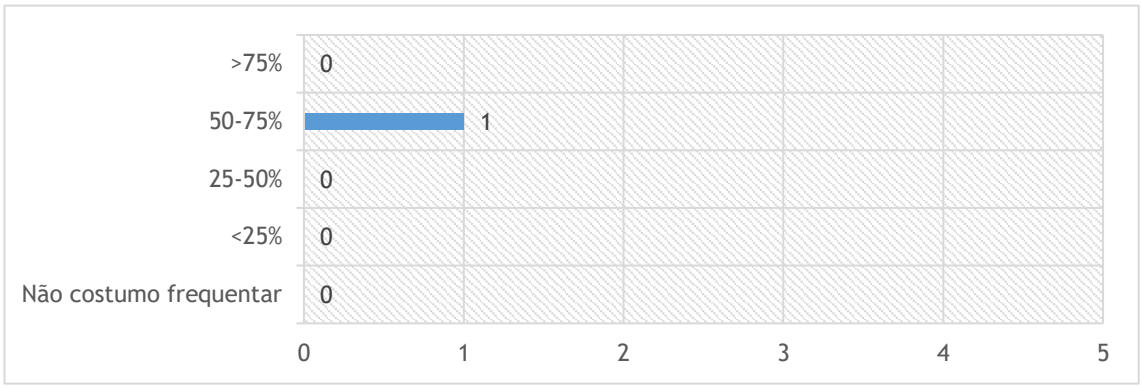


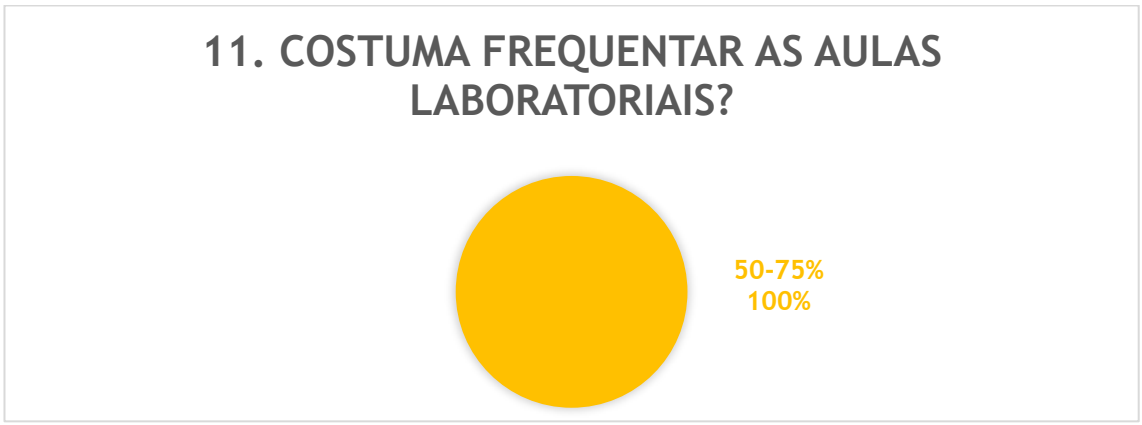
7. Se anteriormente respondeu "Outros", pode indicar quais?

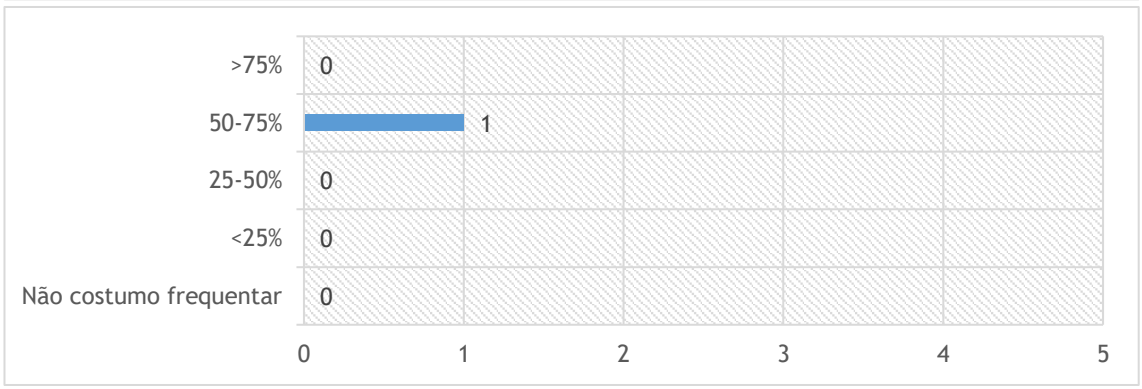
- Sem resposta.


8. QUANTO TEMPO COSTUMA DEDICAR, SEMANALMENTE, AO ESTUDO DE PROGRAMAÇÃO?
30min-1h
100%
>3h
2-3h
1-2h
30min-1h
<30min
0
0
0
1
0
0
1
2
3
4
5


9. QUANTO TEMPO COSTUMA DEDICAR, SEMANALMENTE, NA REALIZAÇÃO DE EXERCÍCIOS DE PROGRAMAÇÃO?
30min-1h
100%
>3h
2-3h
1-2h
30min-1h
<30min
0
0
0
1
0
0
1
2
3
4
5







12. Quais são os aspetos mais positivos no ensino-aprendizagem de programação no ISEC?

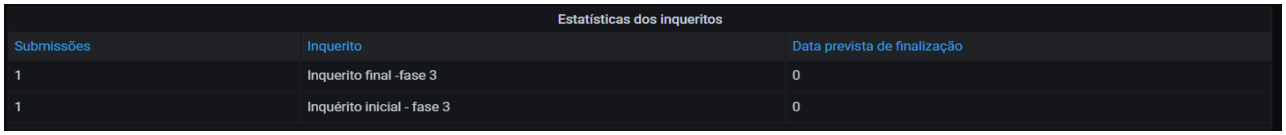
- Um dos aspetos mias positivos no ensino-aprendizagem de programação no ISEC é a facilidade com que somos integrados na linguagem de programação principalmente para aqueles que vinham de um curso de Ciências e Tecnologias que em nada tem em comparação com linguagem de programação como é o meu caso.

13. O que considera que deveria mudar no ensino da programação no ISEC de modo a melhorar a sua aprendizagem?

- Deveria voltar o método de ensino presencial, principalmente nas cadeiras de programação, visto que é uma forma mais fácil de interagir com o professor em caso de dúvidas.

2. Inquérito Final

Relativamente ao inquérito final realizado aos alunos, foi submetida 1 resposta.



Estatísticas dos inqueritos

Submissões	Inquerito	Data prevista de finalização
1	Inquerito final -fase 3	0
1	Inquérito inicial - fase 3	0

Figura 2 - Participação inquéritos

2.1. Questões

1. Realizou todo este programa/tarefas?

Respostas possíveis: Realizei todos os passos cuidadosamente; Realizei todos os passos, mas a dada altura já não os fiz com atenção; Saltei alguns passos; Saltei todos ou quase todos os passos que podia até chegar ao fim; Não cheguei ao fim.

2. Acha que as instruções dadas por esta plataforma eram completas e claras?

Respostas possíveis: 0- Nada claras e muito incompletas; 1; 2; 3; 4; 5; 6; 7; 8; 9; 10- Claríssimas e muito completas.

3. Foi fácil e intuitivo usar esta plataforma, seguindo todos os passos?

Respostas possíveis: 0- Nada fácil nem intuitivo; 1; 2; 3; 4; 5; 6; 7; 8; 9; 10- Extremamente fácil e intuitivo.

4. Sentiu necessidade de pedir ajuda relacionada com questões técnicas de programação, para conseguir avançar nos exercícios?

Respostas possíveis: Não; Sim, mas acabei por não pedir; Sim, 1 ou 2 vezes; Sim, várias vezes.

5. Quais foram as maiores dificuldades que teve?

Respostas possíveis: Resposta curta.

6. Pediu ajuda relacionada com a matéria?

Respostas possíveis: Não, nunca tive necessidade; Não, mas até tinha necessidade; Sim, quanto tenho dúvidas costumo pedir ajuda; Sim, por vezes quando necessito peço ajuda; Sim, e nem é costume eu pedir ajuda.

7. Como classifica a clareza das explicações sobre os exercícios?

Respostas possíveis: 0- Nada claras; 1; 2; 3; 4; 5; 6; 7; 8; 9; 10- Claríssimas.

8. Como classifica a facilidade de utilização do *Code Runner* para a realização dos exercícios+testes?

Respostas possíveis: 0- Não foi nada fácil; 1; 2; 3; 4; 5; 6; 7; 8; 9; 10- Foi extremamente fácil.

9. Como classifica o feedback que obteve (explicações e feedback automático dos exercícios, notas, feedback do professor)?

Respostas possíveis: 0- Inútil; 1; 2; 3; 4; 5; 6; 7; 8; 9; 10- Extremamente útil.

10. Como classifica a sua experiência de utilização desta plataforma para aprendizagem de Estruturas?

Respostas possíveis: 0- Muito má; 1; 2; 3; 4; 5; 6; 7; 8; 9; 10- Excelente.

11. Quais são os pontos positivos desta experiência?

Respostas possíveis: Resposta curta.

12. Quais são os pontos menos bons?

Respostas possíveis: Resposta curta.

13. Tem sugestões de melhoria?

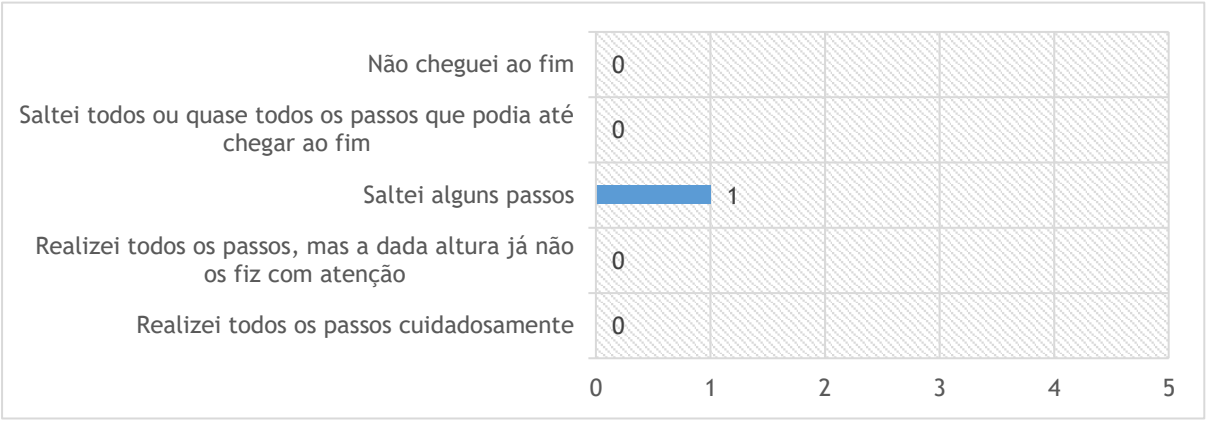
Respostas possíveis: Resposta curta.

14. Existe mais algum ponto positivo ou negativo ou comentário final, relativo a toda a experiência que gostasse de realçar?

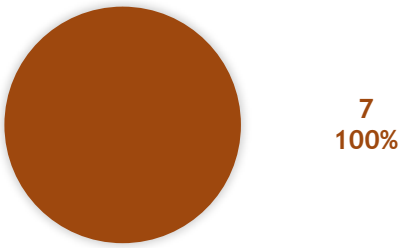
Respostas possíveis: Resposta curta.

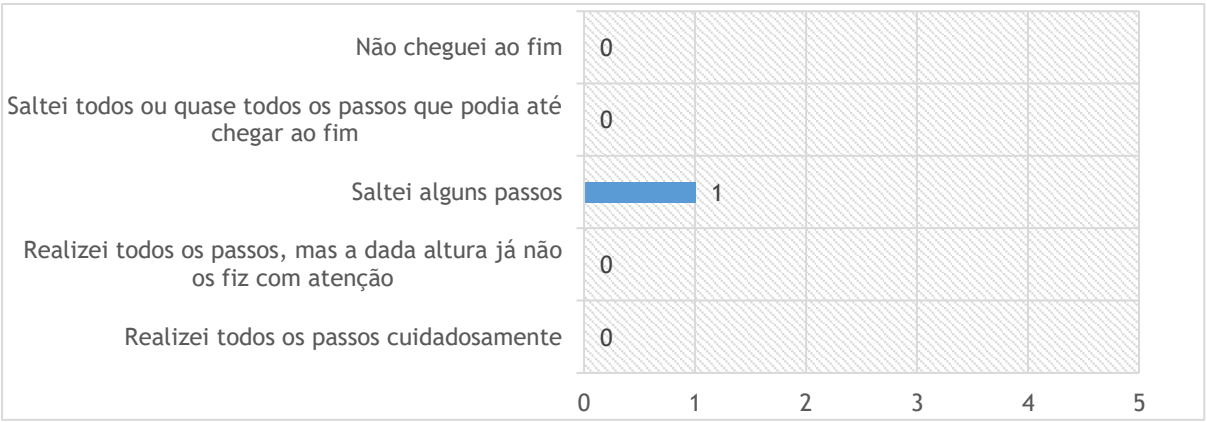
2.2. Resultados

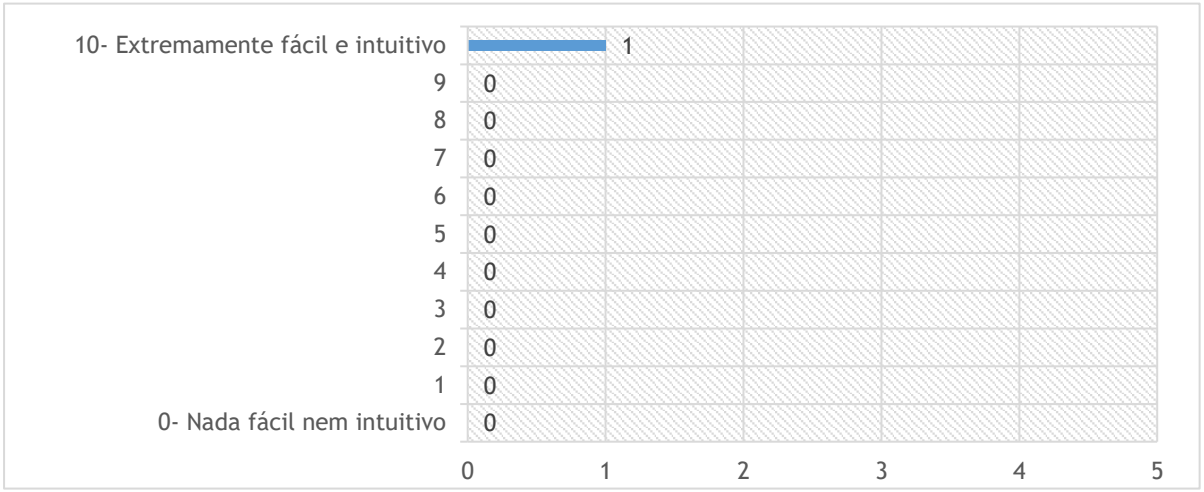
1. REALIZOU TODO ESTE PROGRAMA/TAREFAS?

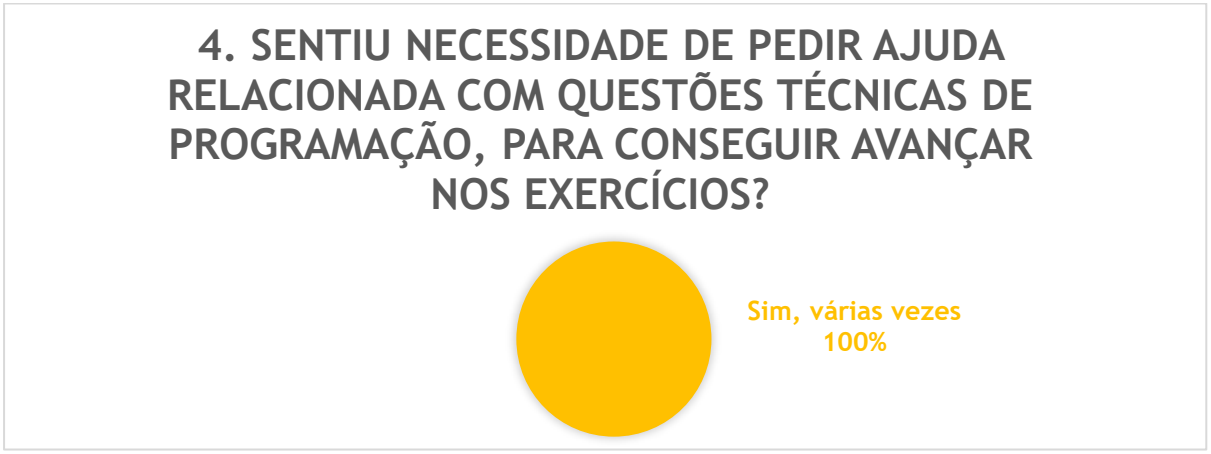


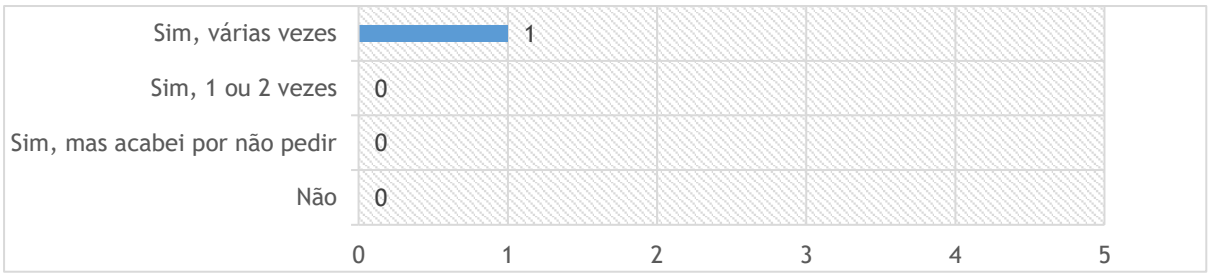
2. ACHA QUE AS INSTRUÇÕES DADAS POR ESTA PLATAFORMA ERAM COMPLETAS E CLARAS?







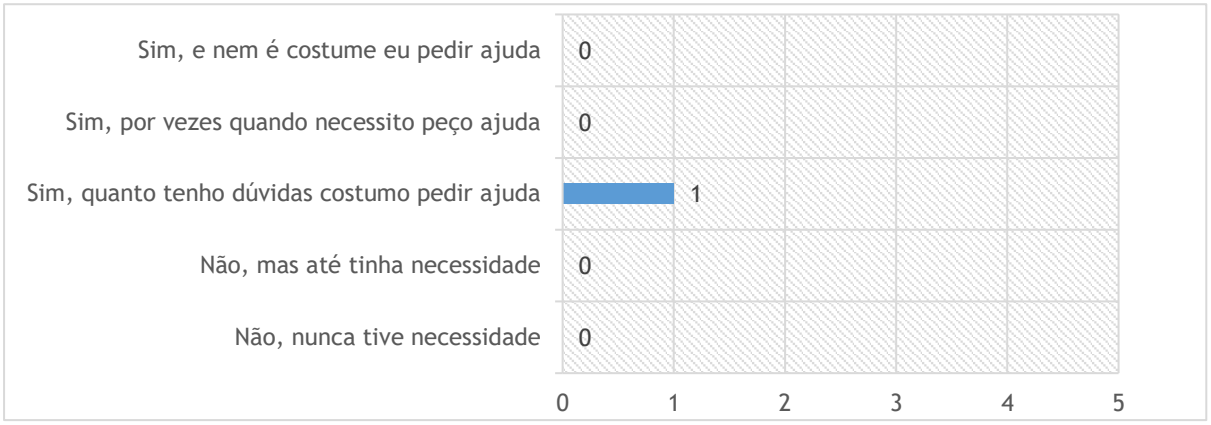


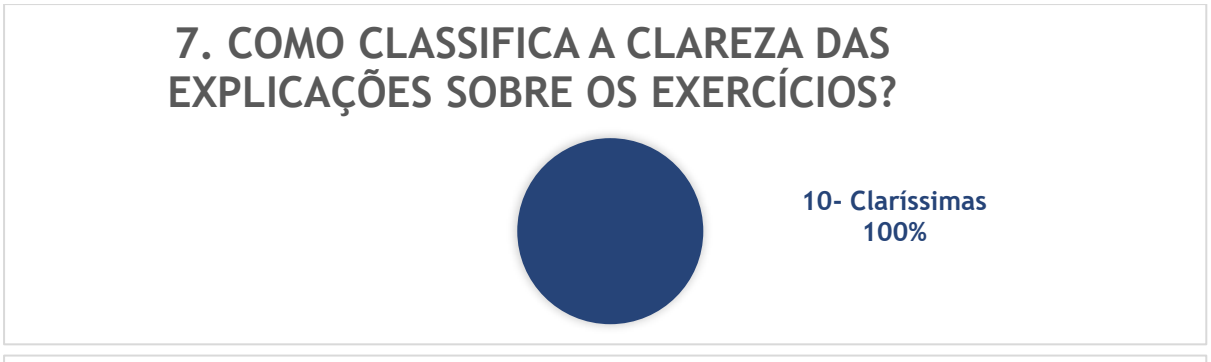


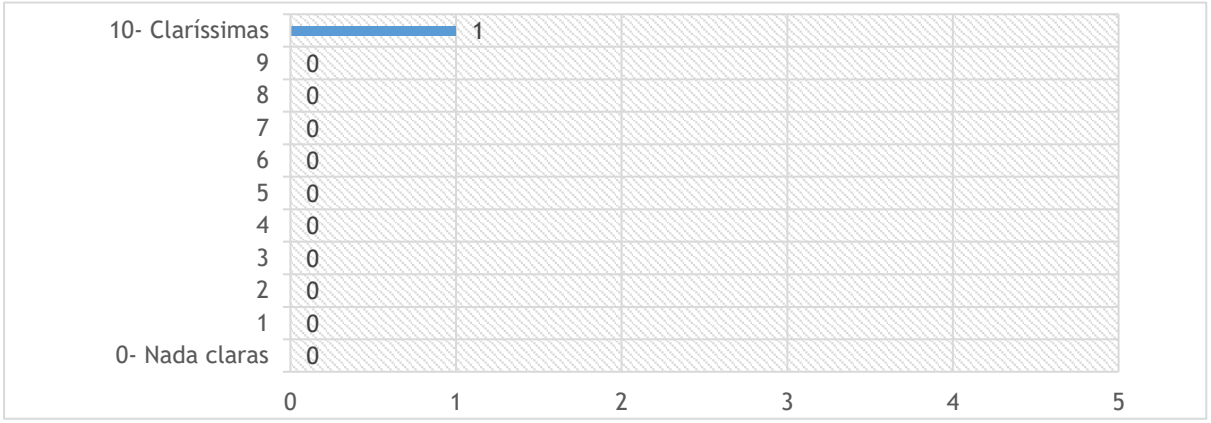
5. Quais foram as maiores dificuldades que teve?

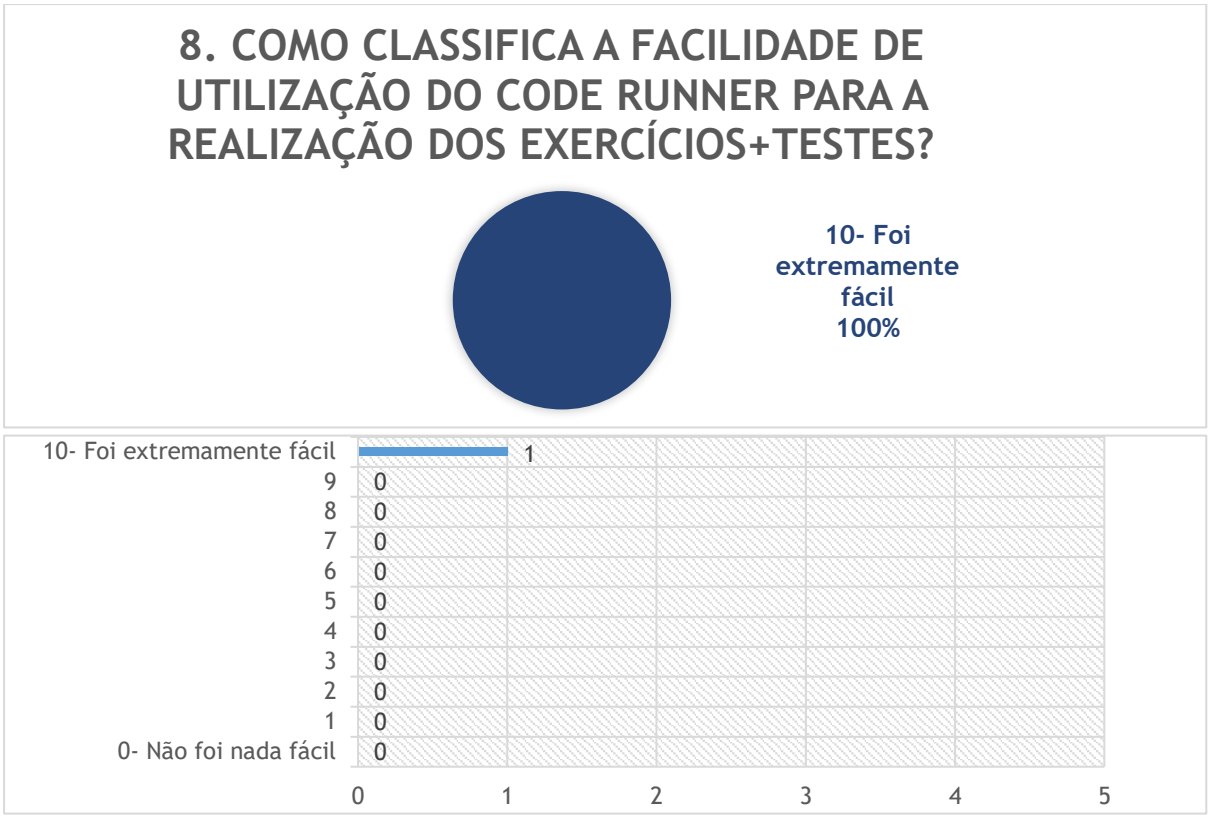
- Fui me adaptando á matéria e a todos os tipos de exercícios presentes

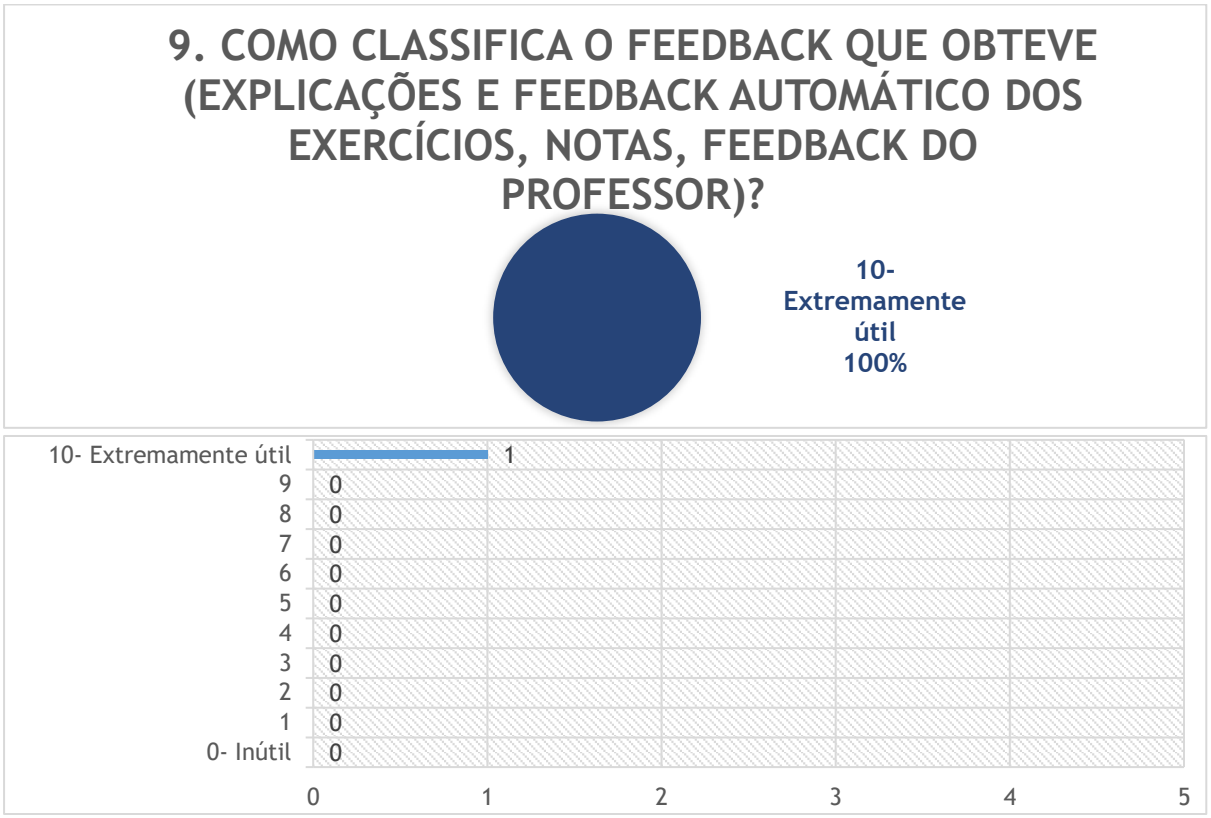

6. PEDIU AJUDA RELACIONADA COM A MATÉRIA ?
Sim, quanto tenho dúvidas costumo pedir ajuda
100%

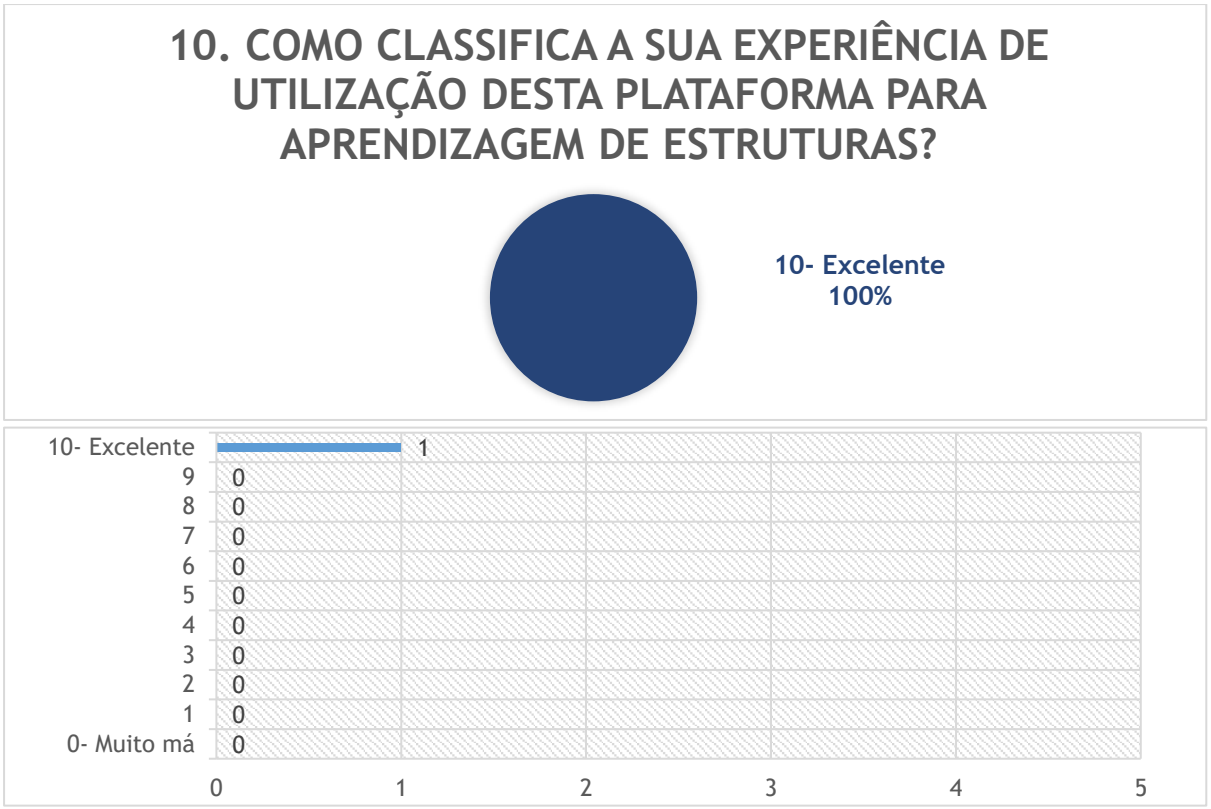

Sim, e nem é costume eu pedir ajuda
Sim, por vezes quando necessito peço ajuda
Sim, quanto tenho dúvidas costumo pedir ajuda
Não, mas até tinha necessidade
Não, nunca tive necessidade
0
0
1
0
0
0
1
2
3
4
5


7. COMO CLASSIFICA A CLAREZA DAS EXPLICAÇÕES SOBRE OS EXERCÍCIOS?
10- Claríssimas
100%


10- Claríssimas
9
8
7
6
5
4
3
2
1
0- Nada claras
1
0
0
0
0
0
0
0
0
0
0
1
2
3
4
5

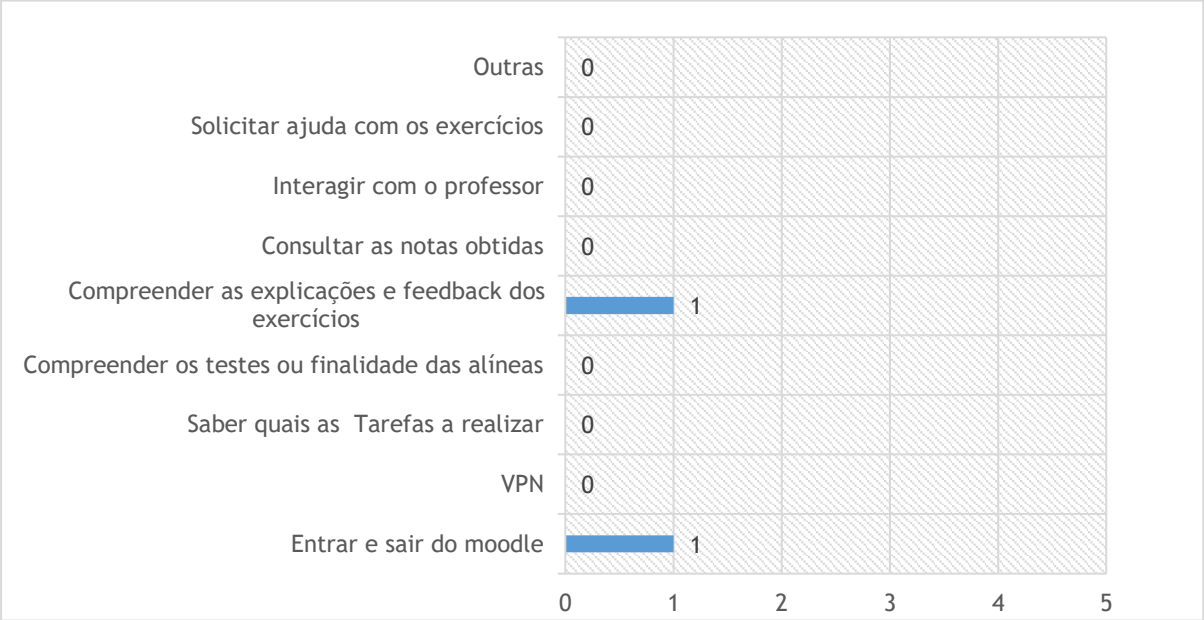

8. COMO CLASSIFICA A FACILIDADE DE UTILIZAÇÃO DO CODE RUNNER PARA A REALIZAÇÃO DOS EXERCÍCIOS+TESTES?
10- Foi extremamente fácil 100%
10- Foi extremamente fácil 1
9 0
8 0
7 0
6 0
5 0
4 0
3 0
2 0
1 0
0- Não foi nada fácil 0
0 1 2 3 4 5


9. COMO CLASSIFICA O FEEDBACK QUE OBTEVE (EXPLICAÇÕES E FEEDBACK AUTOMÁTICO DOS EXERCÍCIOS, NOTAS, FEEDBACK DO PROFESSOR)?
10- Extremamente útil 100%
10- Extremamente útil 1
9 0
8 0
7 0
6 0
5 0
4 0
3 0
2 0
1 0
0- Inútil 0
0 1 2 3 4 5



11. Quais são os pontos positivos desta experiência?

- Resolução integral dos exercícios efetuados.



13. Tem sugestões de melhoria?

- O tempo de submissão é bastante demorado

14. Existe mais algum ponto positivo ou negativo ou comentário final, relativo a toda a experiência que gostasse de realçar?

- Queria deixar um agradecimento especial ao colega André Vicente e ao professor Francisco Pereira pela ferramenta disponibilizada.

Anexo D: Moodle LMSOps V1.0 Release Notes

Índice Anexo D

1. Introdução

Esta versão pretende:

- Criar plano de estudo com competências que correspondem aos diversos tópicos de uma disciplina;
- Criar exercícios codificados que possam validar as resoluções submetidas, retornar feedback e atribuir uma nota;
- Permitir aos professores receber notificação das resoluções e rever as notas;
- Criar um sistema que possa atribuir pontos e níveis a cada aluno mediante a sua utilização da plataforma;
- Criar um relatório de resolução do exercício para o professor;
- Criar uma pauta que possa conter um resumo de todas as resoluções e notas;
- Criar/iniciar pipeline que possa automatizar ao máximo todos as atividades realizadas no moodle.

2. Sobre a release

O objetivo desta entrega inicial é fazer uso da plataforma moodle, das suas funcionalidades e de todos os plugins que possam ser relevantes no intuito de melhorar a disponibilização de exercícios codificados.

3. *Scope*

De acordo com as fases definidas, e atendendo a uma escala crescente de não contemplado, parcialmente contemplado e contemplado, as fases do pipeline mapeadas nos passos da plataforma seriam as seguintes:

1. Fase *Plan* (definição de roteiros/percurso de aprendizagem):
   1.1. O professor define os objetivos de aprendizagem por tema e/ou semanais – contemplado
      - Cria disciplina, faz upload do conteúdo por tópicos de aprendizagem, cria ou importa as perguntas codificadas para o banco de perguntas da disciplina.
      - Define pauta e pesos das questões e restantes parâmetros.
      - Dá destaque ao tópico/conteúdos programáticos e aos exercícios que pretender.
   1.2. O professor identifica os elementos de estudo para cada tema e/ou semana - parcialmente contemplado
      - Pode definir questionários teóricos ou definir conteúdos a ver através do fórum, outros avisos ou mensagens do moodle. O *plugin* 'Tarefas a realizar' cumpre necessariamente a finalidade de lista de tarefas.
   1.3. O professor define as tarefas a cumprir por tema e/ou semana - parcialmente contemplado

- É definido globalmente uma barra de progresso com os exercícios que é pretendido que sejam realizados pelos alunos. Essa barra pode ir sendo alargada conforme os exercícios são disponibilizados. A barra é geral para todos os alunos inscritos na disciplina.
- Abordagens mais individuais não são ainda suportadas. No entanto é possível através de grupos disponibilizar mais exercícios aos alunos que tenham dificuldades em cada tópico.
- Data de término nos exercícios.

2. Fase *Code* (Plano de estudo):
   2.1. O aluno planeia o seu estudo para a semana ou para o tema - contemplado
   - Usa o plugin 'Tarefas a realizar' para verificar as novas tarefas a realizar. Normalmente terá os seus tópicos alinhados pelo professor à semana, pelo conteúdo disponibilizado ou pelas datas de término dos exercícios.
   - Pode consultar a barra de tarefas e a data de término das tarefas a realizar. Desbloqueia novos exercícios pela resolução de exercícios anteriores.
3. Fase *Build and Test* (Estudo):
   3.1. O aluno estuda elementos - contemplado
   - Visualiza e estuda os elementos teóricos disponibilizados na disciplina e vê e realiza exercícios codificados, questionários ou outros recursos.

   3.2. O aluno realiza exercícios - contemplado
   - As questões implementadas são do tipo 'Teste/Exercício' com *Code Runner* e questionários.

   3.3. O aluno visualiza os elementos de estudo (conteúdo teórico ou pratico) e resolve os exercícios – contemplado
   - Conteúdo teórico.
   - Conteúdo pratico, os exercícios são criados através do recurso 'Teste/Exercício', que faz registo do progresso e submissões feitas em cada exercício;

   3.4. O aluno tem feedback automático (ex: testes, análise estática de código) - parcialmente contemplado
   - Após realizar os exercícios é retornado feedback sobre a resolução submetida, através da indicação de exercícios e conteúdo teórico semelhante presente nos slides teóricos e do objetivo do exercício. Mediante a nota alcançada no exercício são também aconselhados alguns passos como repetir o exercício ou visualizar novamente o conteúdo teórico;
   - É apenas validado o input e output do programa, não existindo analise estática de código. No entanto, o aluno pode ser forçado a usar certar estruturas ou variáveis através das pré-implementações.

   3.5. O aluno pode pedir ajuda ao professor - parcialmente contemplado
   - Pode contactar o professor através do sistema de mensagens integrado do Moodle;

- O plugin '*Message my Teacher'* presente na página da disciplina apresenta um shortcut para o chat com o(s) professor(es) da disciplina;
- Poderá também enviar email ou participar no fórum da disciplina.

4. Fase *Release* (Submissão):
   4.1. O aluno submete os trabalhos que são dados como concluídos na plataforma - contemplado
   - São disponibilizadas questões do tipo 'Teste/Exercício' com *Code Runner* e questões de tipo questionário. Nos exercícios codificados é atribuída uma nota automaticamente após a submissão de resolução e enviada notificação ao professor.

   4.2. O aluno pode voltar ao ponto '3. Fase *Build* e *Test*/Estudo', até terminar todas as tarefas definidas pelo docente - contemplado
   - Vários exercícios disponíveis em cada tópico/unidade programática.
   - O tempo despendido e número de tentativas do aluno em cada exercício é registado na submissão do teste/exercício.

5. Fase *Deploy and Operate* (Análise):
   5.1. O professor analisa os relatórios disponíveis na plataforma sobre a sua utilização e os resultados dos exercícios - contemplado
   - É recebida notificação após cada aluno finalizar a resolução de um exercício.
   - Pode visualizar relatórios de submissão/resolução no teste/exercício.

   5.2. O professor analisa manualmente os exercícios submetidos - contemplado
   - Através do link recebido na notificação é possível ir diretamente para a resolução do exercício submetida pelo aluno e rever a tentativa.

6. Fase *Monitor* (Feedback):
   6.1. O professor dá feedback semiautomático ao aluno, com base nos relatórios da ferramenta - parcialmente contemplado
   - É possível colocar comentários nas resoluções submetidas, no entanto, o aluno não recebe notificação desse feedback. Pode também ser utilizado outro recurso para este efeito como o sistema de mensagens.

   6.2. O professor dá feedback personalizado ao aluno, ajustando os próximos objetivos de aprendizagem - parcialmente contemplado
   - Não é possível definir os próximos passos individualmente, no entanto pode ser individualizado a grupos com base nas dificuldades sentidas pelos alunos. Esses grupos poderão ter acesso a mais exercícios daquele tópico/unidade programática.

4. *Features*

Foram considerados os seguintes elementos:

4.1 Disciplina

Uma disciplina constitui um espaço com professores e alunos onde é possível criar recursos de diversos tipos, publicar atividades, exercícios, avisos e fóruns.

Já são usadas disciplinas no moodle do ISEC para o leccionamento e disponibilização de conteúdos. Assim será utilizado este recurso para disponibilização dos conteúdos adicionais criados. Caso haja oportunidade de criar disciplinas, o tipo de disciplina que se pretendia criar seria no formato de semanas e com o número de tópicos igual aos tópicos principais dos conteúdos programáticos. Desta forma será possível dispor os conteúdos de cada unidade de conteúdo programático em tópicos diferentes e realçar o tópico pretendido em função do espaço temporal pretendido.

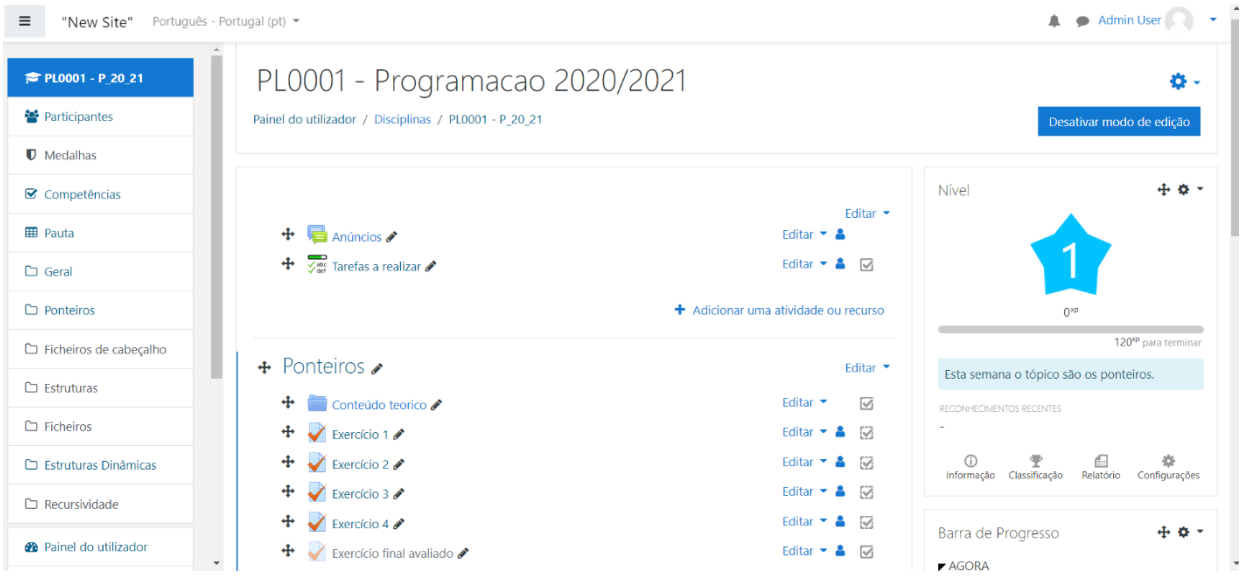


Figure 1 - Overview de disciplina

A displina servirá de dashboard às seguintes funcionalidades para os alunos:

- Repositório de conteúdo teórico e exercícios questionário ou de submissão;
- Repositório de exercícios codificados para resolver;
  - Através da configuração do plugin *CodeRunner*.
- Classificação mediante as atividades realizadas;
  - Através da configuração do plugin *LevelUP*.
- Barra de progresso para as tarefas a realizar;
  - Através da configuração do plugin *Progress Bar*.
  - Através da configuração do plugin *CheckList*.
- Contactar o professor(es);
  - Através da configuração do plugin *MyTeacher;*
  - Fórum, mensagem via moodle ou email.
- Aceder às medalhes que lhe foram atribuídas pela disciplina;
- Aceder as competências presentes na disciplina;

- Aceder a sua pauta e notas.

A displina servirá de dashboard às seguintes funcionalidades para os professores:

- Receber notificações sobre a resolução de exercícios e ou mensagens dos alunos;
- Configurar a pauta e os pesos para cada um dos exercícios/atividades disponibilizadas;
- Disponibilizar exercícios codificados;
    - Através da configuração do plugin *Code Runner*.
- Disponibilizar questionários ou exercícios de submissão;
- Configurar tarefas a realizar, barra de progresso, classificação e competências.
- Visualizar relatórios de atividade.
- Verificar relatórios de competência e avaliar os alunos.

4.2 Competências

Uma 'Competência' refere-se ao nível de compreensão ou proficiência num determinado tema ou habilidade. O *Moodle* permite a implementação de modelos de educação baseados em competências, permitindo criar quadro/planos de competências que podem ser associados a disciplinas e adquiridos pelos alunos.

O administrador pode configurar os quadro/planos de competências e atribuí-las aos alunos. O professor pode associar competências às suas disciplinas, bem como a recursos e atividades de modo que os alunos quando resolvam essas atividades adquiram proficiência nas competências que foram atribuídas. O objetivo será ter um exercício final por cada tópico de conteúdo programático que permitirá ao aluno cimentar o seu conhecimento naquele tópico obtendo proficiência na competência e uma determinada valorização na disciplina.

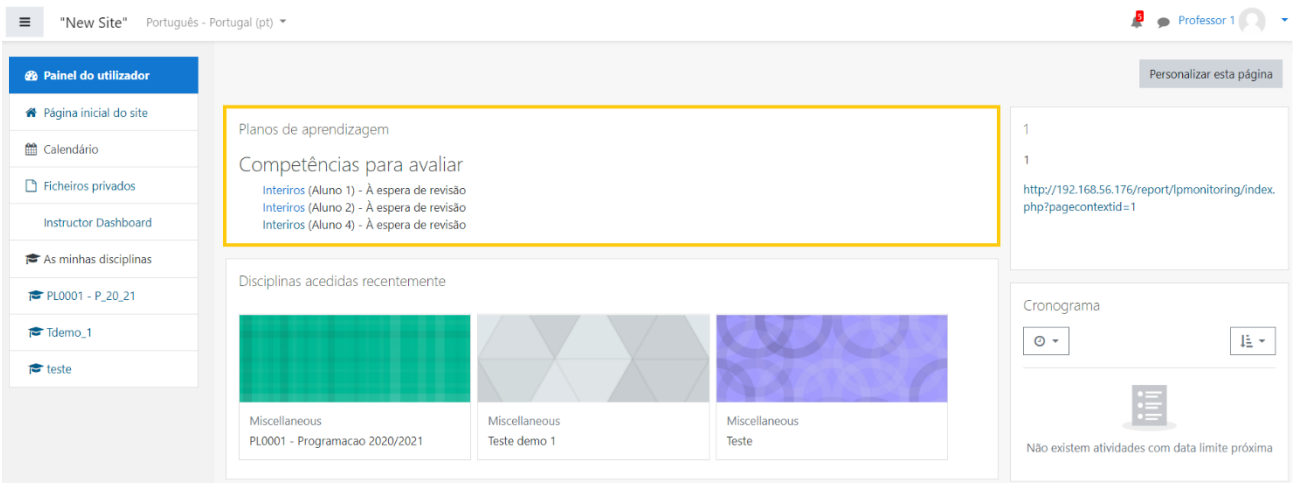


Figure 2 - Visão professor 'Competências'

O professor deve configurar quais os exercícios que são considerados para as ‘Competencias’ assonciadas as disciplinas.

Figure 3 - Configuração de ‘Competências’

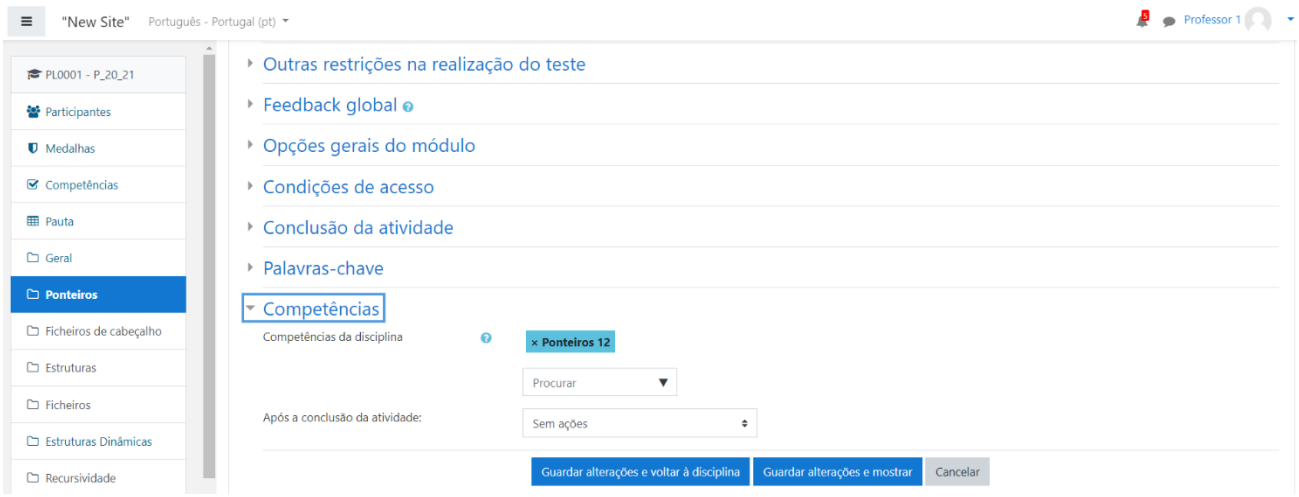


Figure 4 - Associação de ‘Competências’ a exercícios

4.3 Pauta

É possível configurar uma pauta com diversas sessões e atribuir um peso específico. É também possível configurar o peso de cada um dos exercícios. Cada uma destas sessões ou exercícios tem também uma cotação final para a classificação definida (ex_0-20, 0-5…).

É pretendido criar um número de sessões igual aos tópicos principais dos conteúdos programáticos. Assim os exercícios estarão agrupados pelo seu conteúdo. Cada uma destas sessões terá depois um peso específico de acordo com o pretendido pelo professor. Será também possível visualizar as notas obtidas pelos alunos e verificar especificamente cada um dos exercícios.

Figure 5 - Visão geral Pauta *Moodle*

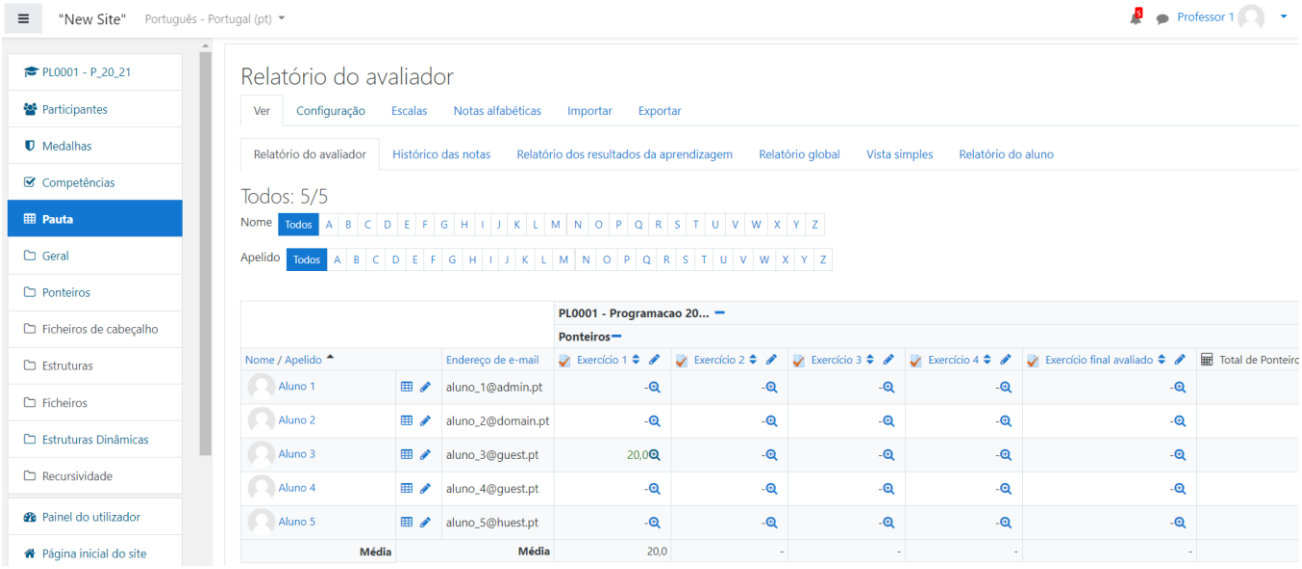


Figure 6 - Relatório de avaliador/professor Pauta

4.4 Code Runner

Plugin que permite criar questões codificadas de uma determinada linguagem de programação para serem resolvidas pelos alunos. São suportadas diversas linguagens de programação, sendo possível configurar feedback e codificar testes para validar e avaliar o código submetido pelos alunos.

Esta funcionalidade pretende ser usada para criar diversos exercícios de programação que possam ser resolvidos pelos alunos. Estes exercícios poderão ser dispostos em vários recursos do tipo 'Teste' e estar organizados por unidade de conhecimento a que dizem respeito. O objetivo do aluno será resolver os exercícios podendo correr o seu código contra os testes do exercício, indo de encontro ao conceito *sandbox* sobre o qual se pretende apenas programar. O aluno poderá correr o seu código e terminar a submissão para obter uma nota.

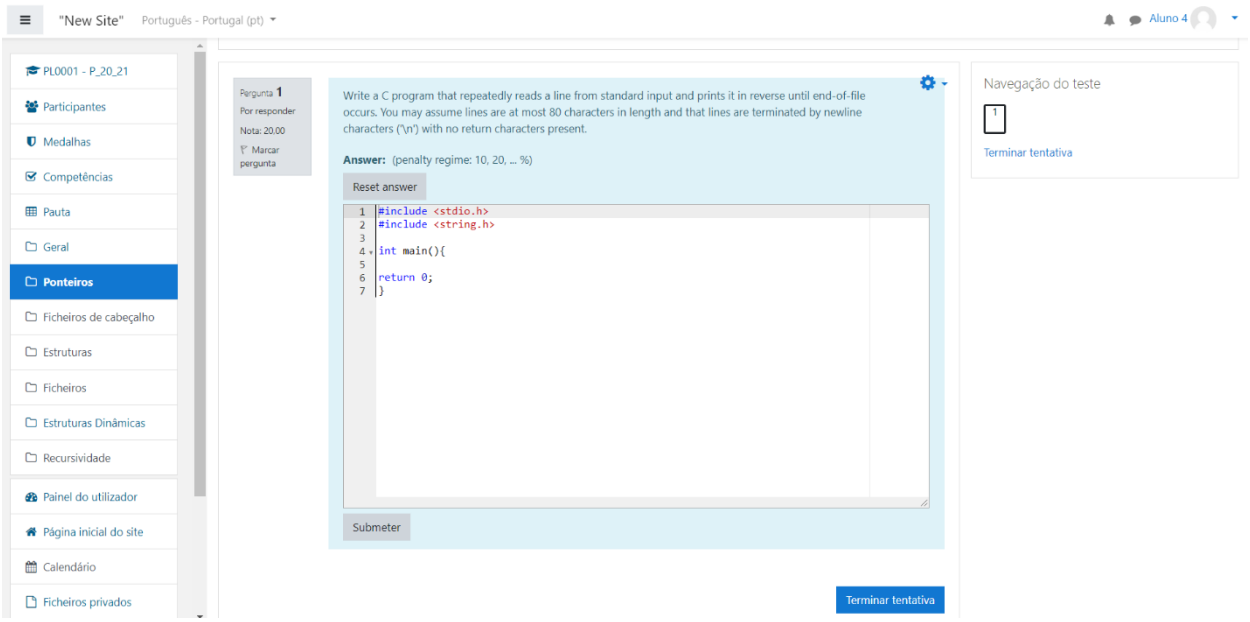


Figure 7 - Visão aluno '*Code Runner*'

O professor pode editar o teste e as opções da questão do '*Code Runner*'. Podendo definir o tipo de pergunta, resposta, casos de teste, feedback e instruções que são carregadas para o aluno quando inicia a resolução.

Figure 8 - Visão professor '*Code Runner*'

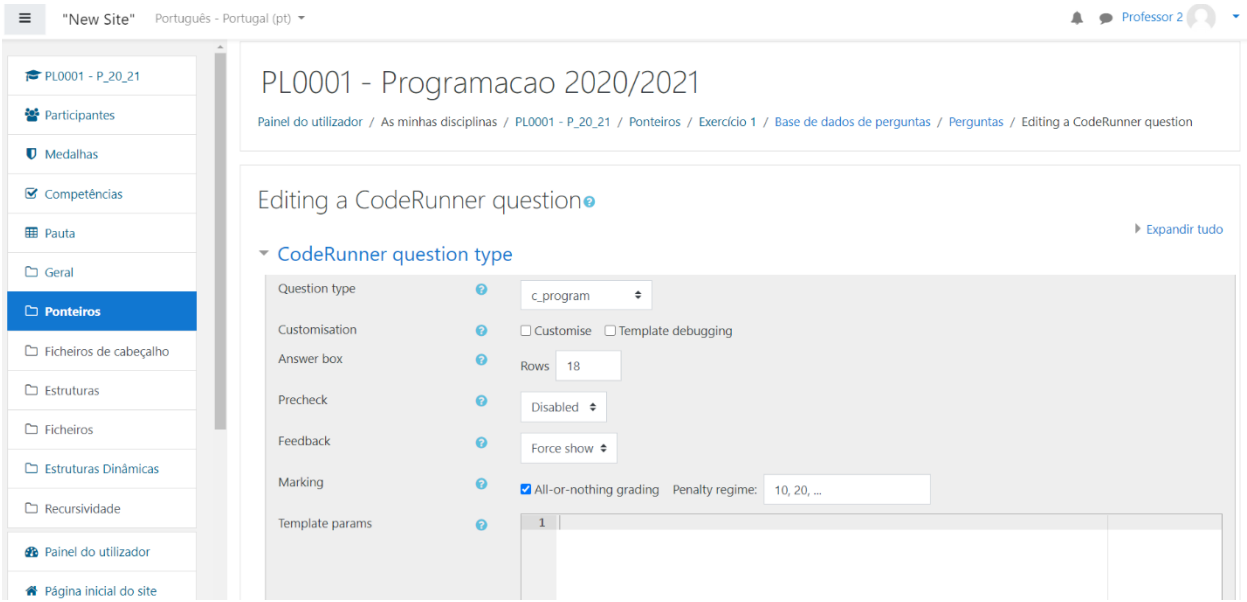


Figure 9 - Tipo de questão '*Code Runner*'

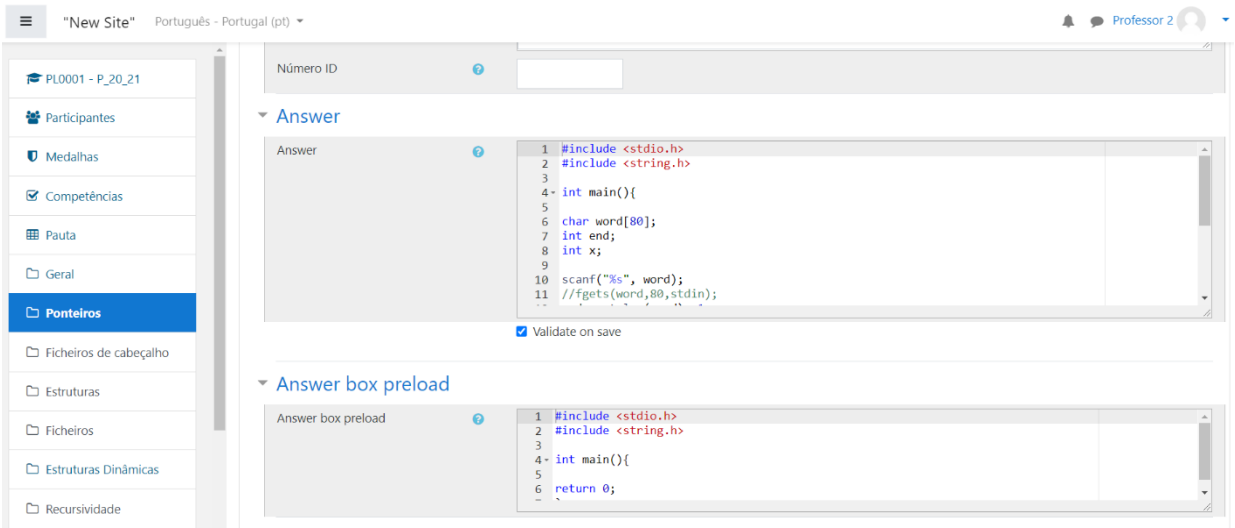


Figure 10 - Caixa de resposta e de *preload* '*Code Runner*'

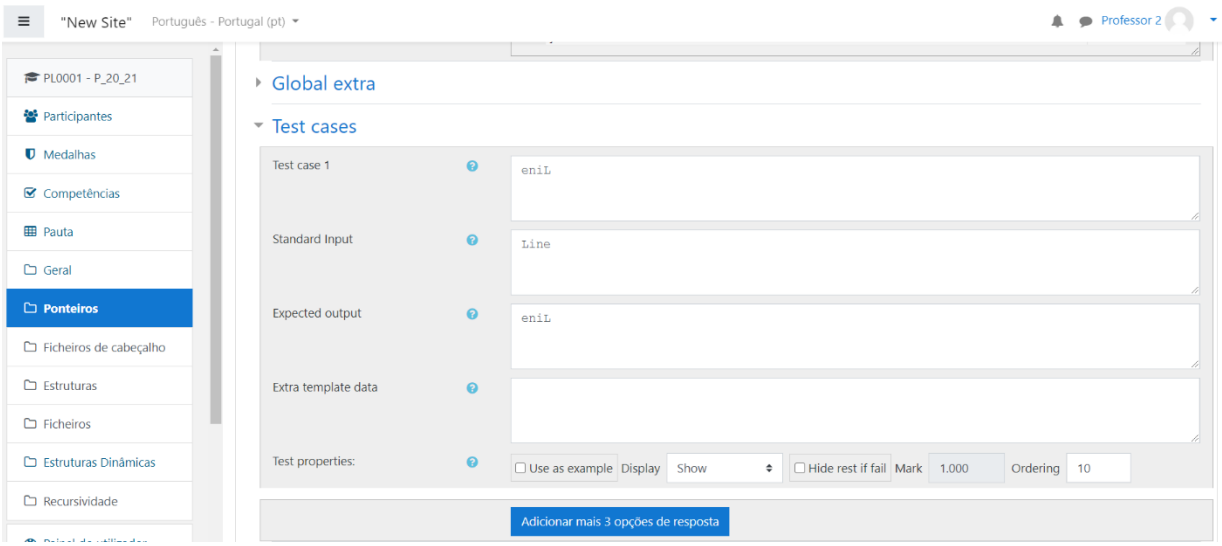


Figure 11 - Testes '*Code Runner*'

4.5 Nível

Plugin que pretende introduzir o conceito de *gamification* na disciplina do *Moodle*.

Esta funcionalidade pretende fomentar o interesse dos alunos na resolução das atividades e exercícios. É também possível realizar anúncios neste plugin e os alunos poderão ver a sua classificação em relação a outros alunos.

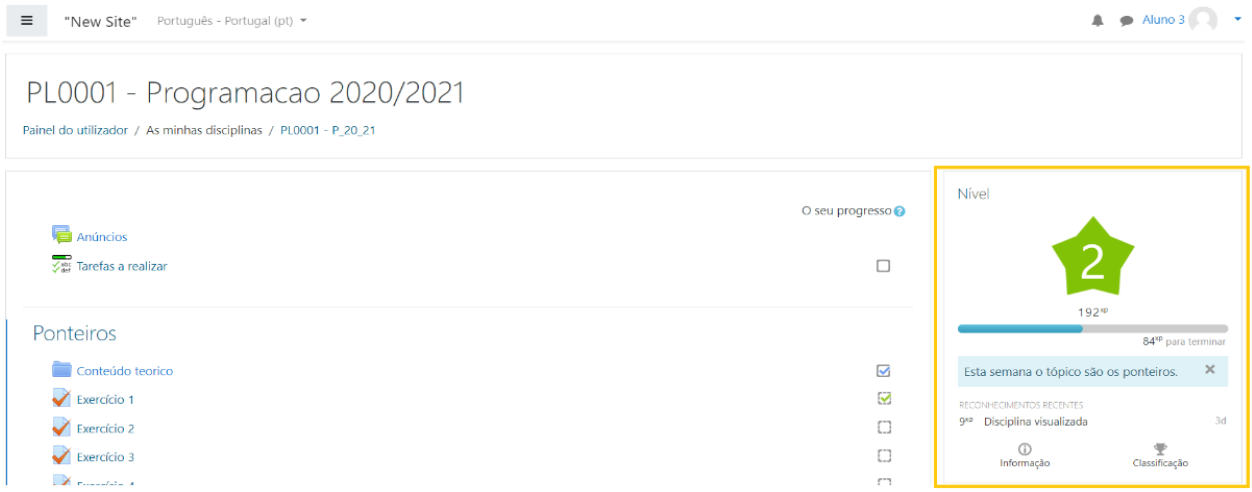


Figure 12 - Visão aluno '*LevelUp*'

Para além destas opções o professor poderá ainda obter relatatorio relativos aos alunos e configurar esta funcionalidade.

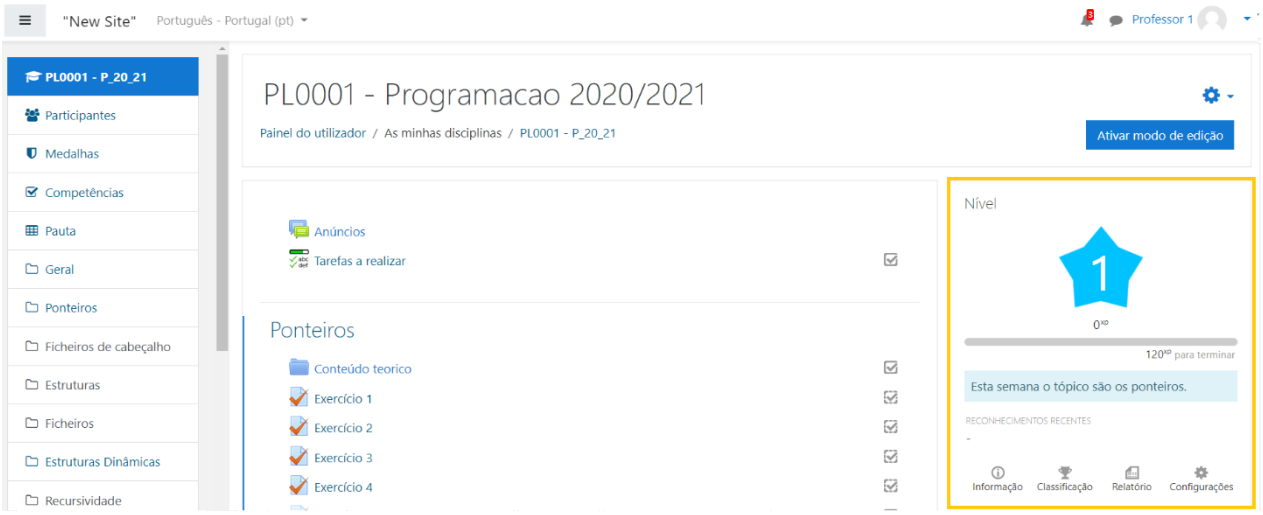


Figure 13 - Visão professor '*LevelUp*'

É possível configurar os níveis existentes, pontos ganhos por ação, nome e descrição.

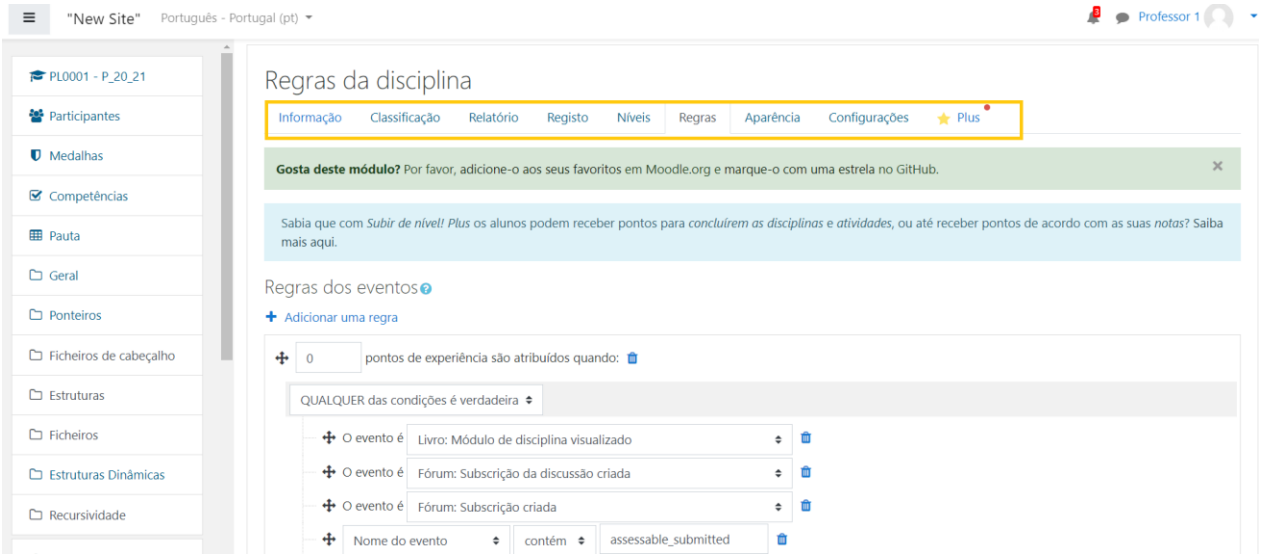


Figure 14 - Administração *'LevelUp'* regras dos eventos

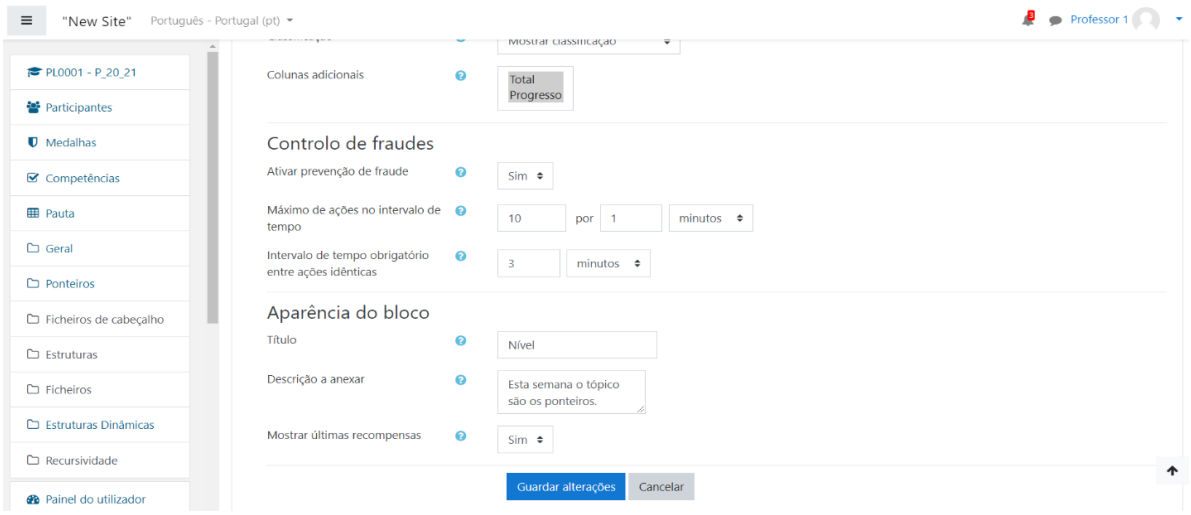


Figure 15 - Administração *'LevelUp'* aparência do bloco

4.6 Barra de progresso

O *plugin* '*Barra de progresso*' tem uma representação visual das atividades da disciplina assim como uma representação de completas ou incompletas. Mostra o progresso dos alunos na sua *dashboard* e permite aos professores identificar as atividades realizadas por cada aluno.

Esta funcionalidade pretende dar uma visualização gráfica do progresso de cada aluno.

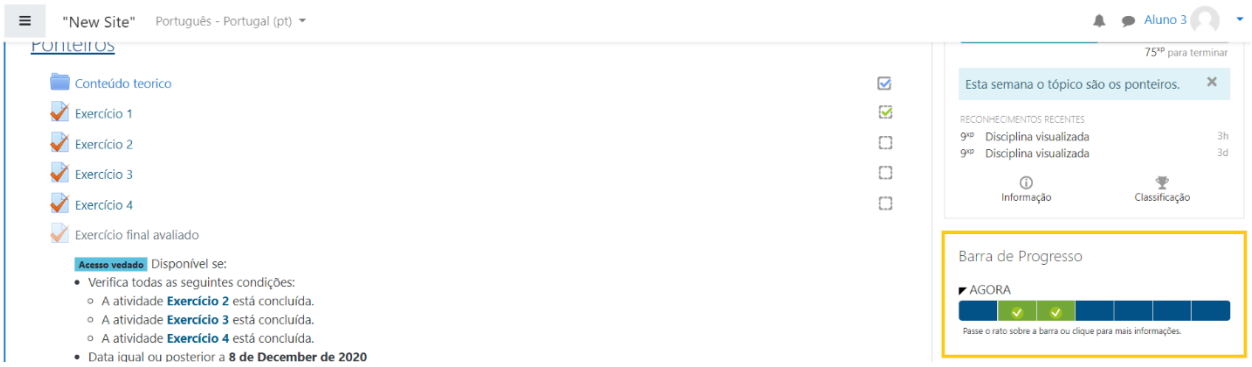


Figure 16 - Visão aluno '*Completion Progress*'

O professor pode visualizar o progresso geral de todos os alunos.

Figure 17 - Visão professor '*Completion Progress*'

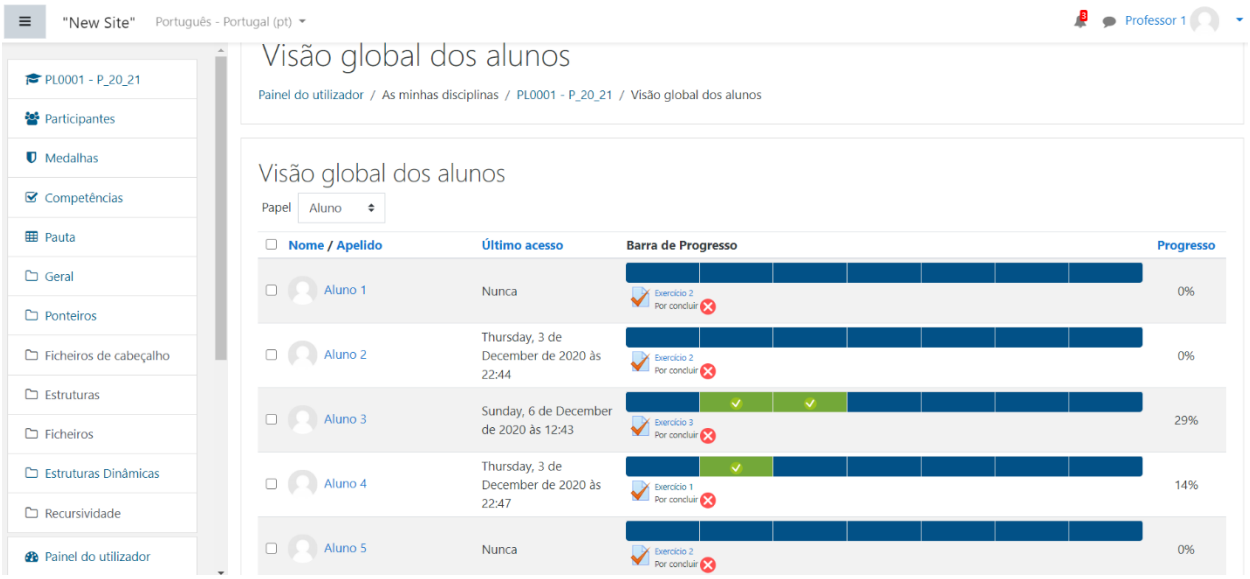


Figure 18 - Visão global professor '*Completion Progress*'

4.7 Mensagem ao professor

O *plugin 'Message my Teacher'* permite aos alunos interagir rapidamente com os professores da disciplina através da utilização do sistema de mensagens interno do *Moodle*.

O objetivo desta funcionalidade é permitir aos alunos contactar rapidamente os professores da disciplina caso tenham alguma dúvida ou questão relativamente as atividades.

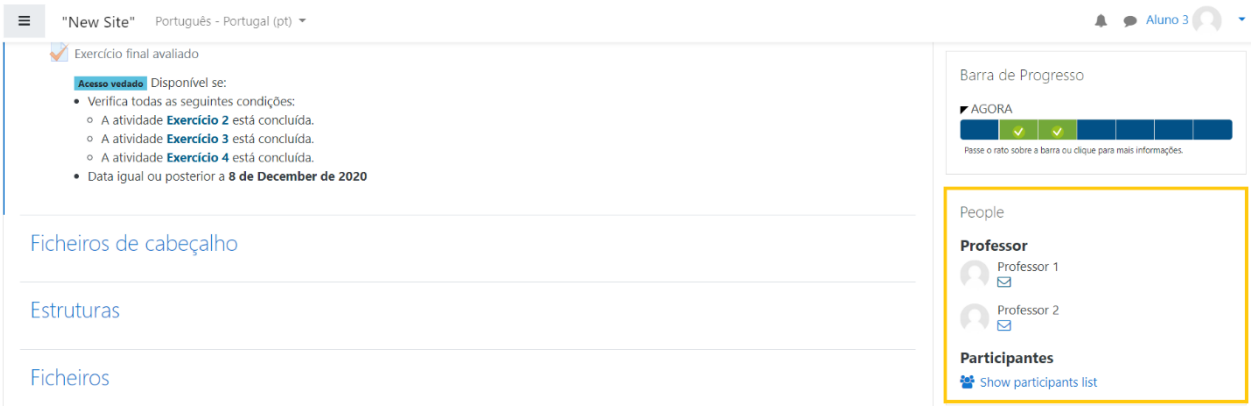


Figure 19 - Visão aluno '*Message my Teacher*'

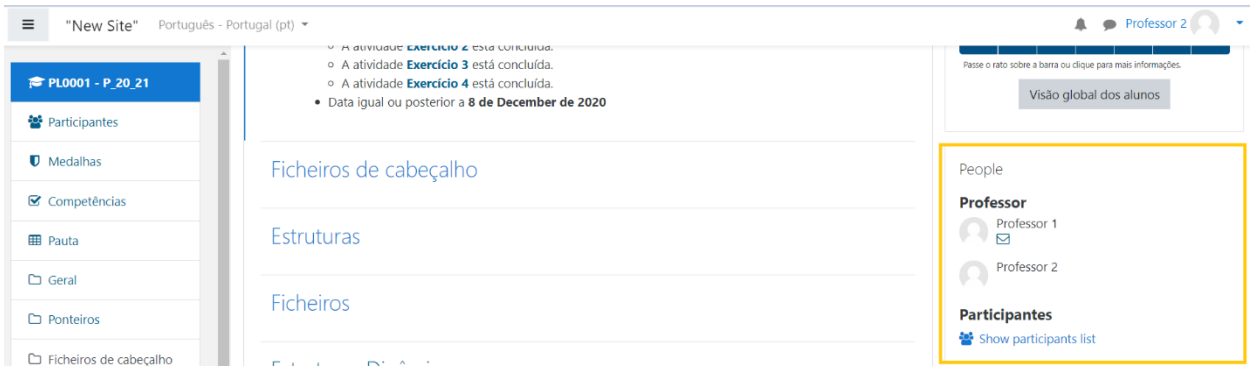


Figure 20 - Visão professor '*Message my Teacher*'

4.8 Medalhas

Com o objetivo de atribuir recompensas pela realização de atividades/exercícios realizados na plataforma, é possível definir medalhas. Esta recompensa pretende ajudar o conceito de *gamification*.

Este plugin poderá ser utilizado para reforçar a recompensar os alunos que realizam mais atividades.

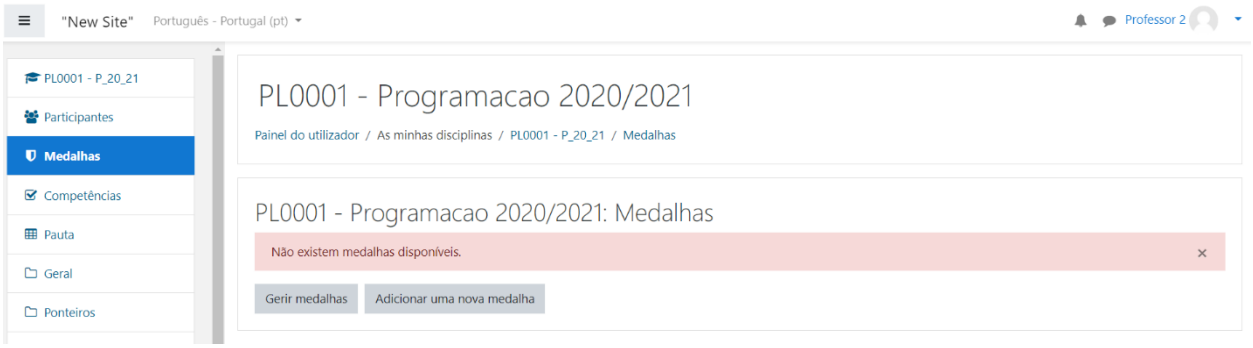


Figure 21 - Medalhas moodle

5. Limitações

5.1 Disciplina

Não é possível automatizar a criação da disciplina, assim este processo é feito através da criação pela interface gráfica.

5.2 Pauta

Não é possível exportar ou importar as categorias da pauta. É apenas possível importar e exportar as notas dos alunos ou recuperar a pauta de outra disciplina.

5.3 Exercícios/criação de conteúdo

A criação de conteúdo e de exercícios ainda é feito de maneira manual e através da interface gráfica.

5.4 Relatórios

Após realização dos exercícios é possível ver um relatório de resolução do exercício. O relatório pode ser algo limitado e a informação pode ser por vezes difícil de compreender.

6. Mecânica/interação com o Moodle

6.1 Aluno

O objetivo do aluno é realizar as atividades previstas para si na disciplina. Cada unidade de conteúdo programático possui vários exercícios codificados e um exercício final no intuito de permitir ao aluno a aquisição de conhecimentos. Adicionalmente são disponibilizados outros tipos de atividades como exercícios teóricos ou de submissão. Consultando os blocos da disciplina poderá verificar a sua posição na resolução de atividades relativamente aos outros alunos. O objetivo é interessar o aluno na resolução de exercícios na plataforma onde obtém feedback imediato da sua resolução, desbloqueando exercícios e progredindo no seu percurso de aprendizagem.

O exercício final da unidade programática ficará disponível após completar parte/todos os restantes exercícios. Este comportamento será idêntico a outras plataformas de programação no qual apenas é permitido concluir a unidade conhecimento e fazer o exame/exercício final após completar vários exercícios preliminares.

O aluno terá realçado na plataforma as tarefas que tem de realizar. Caso seja revista a nota de algum exercício o aluno terá que manualmente consultar a nota dada.

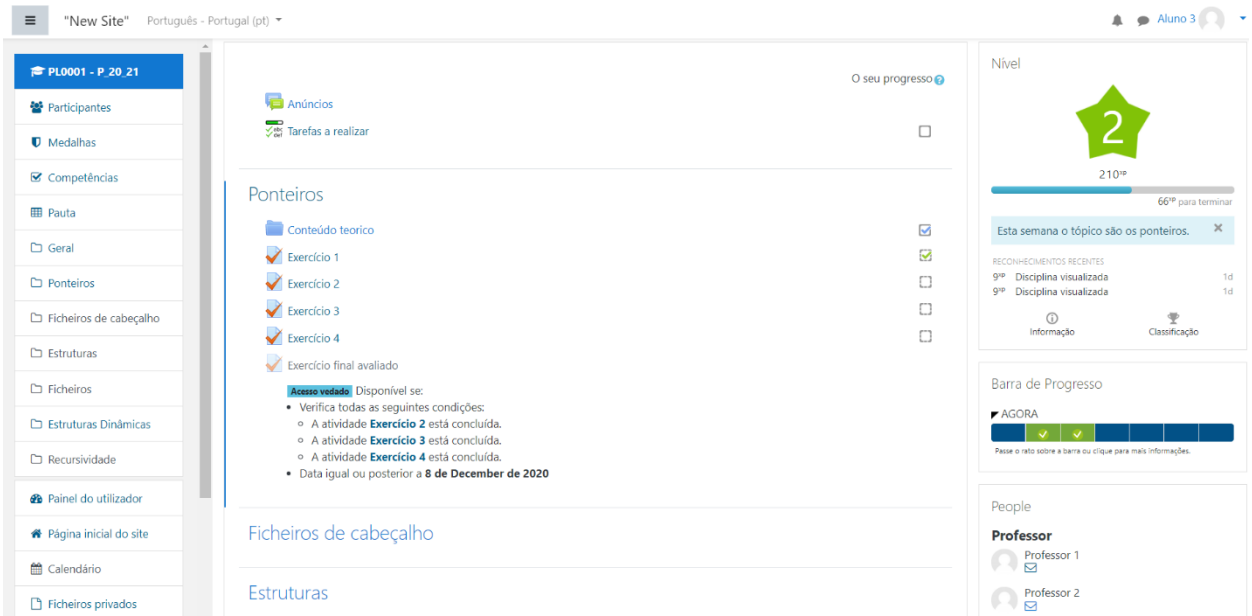


Figure 22 - Visão aluno da disciplina Moodle

6.2 Professor

O professor deverá configurar a disciplina criando o mesmo número de tópicos correspondente aos conteúdos programáticos. Deverá também criar as categorias na pauta de maneira a poder agrupar os

exercícios criados por tipo de conteúdo. Idealmente cada unidade programática poderá valer por exemplo 1 valor de nota final na disciplina. Os exercícios serão criados através do plugin *Code Runner* e a atividade do tipo 'Teste'. As questões poderão ser acrescentadas através da base de dados de perguntas, criadas de raiz ou importadas através de um *template XML*. Esta última opção deverá ser a privilegiada, podendo o professor criar um repositório de questões próprio, versionado e que facilita a automatização e importação de perguntas em qualquer disciplina. Deverá também configurar os blocos '*LevelUP'*, '*Progress', 'MessageMyTeacher'* e 'Tarefas a Realizar'. Estes plugins podem ajudar a melhorar a interação entre os alunos, professores e a disciplina.

Após serem criados os exercícios o professor deverá organizar as tarefas a realizar, podendo dar destaque aos exercícios que pretendem que sejam realizados, utilizando o plugin 'Tarefas a Realizar' e realçando a respetivo tópico. Por cada exercício realizado e submetido pelos alunos será recebida uma notificação com um link para a submissão onde a mesma poderá ser revista. Poderá também adicionar competências à sua disciplina que simbolizem o término e aquisição do conhecimento necessário de uma unidade programática. O plano de competências é criado pelo administrado do Moodle. As competências podem assim ser atribuídas aos exercícios finais, desencadeando uma aprovação de competência e revisão após cada submissão realizada. Através da atribuição de competências aos exercícios ficam disponíveis relatórios mais pormenorizados e com mais informação sobre as competências.

Quanto mais conteúdo o professor criar, mais exercícios poderão ser resolvidos pelos alunos obtendo feedback após resolução. Este conteúdo será extensível para outros anos através importando os *templates* das perguntas ou através do base de dados de perguntas.

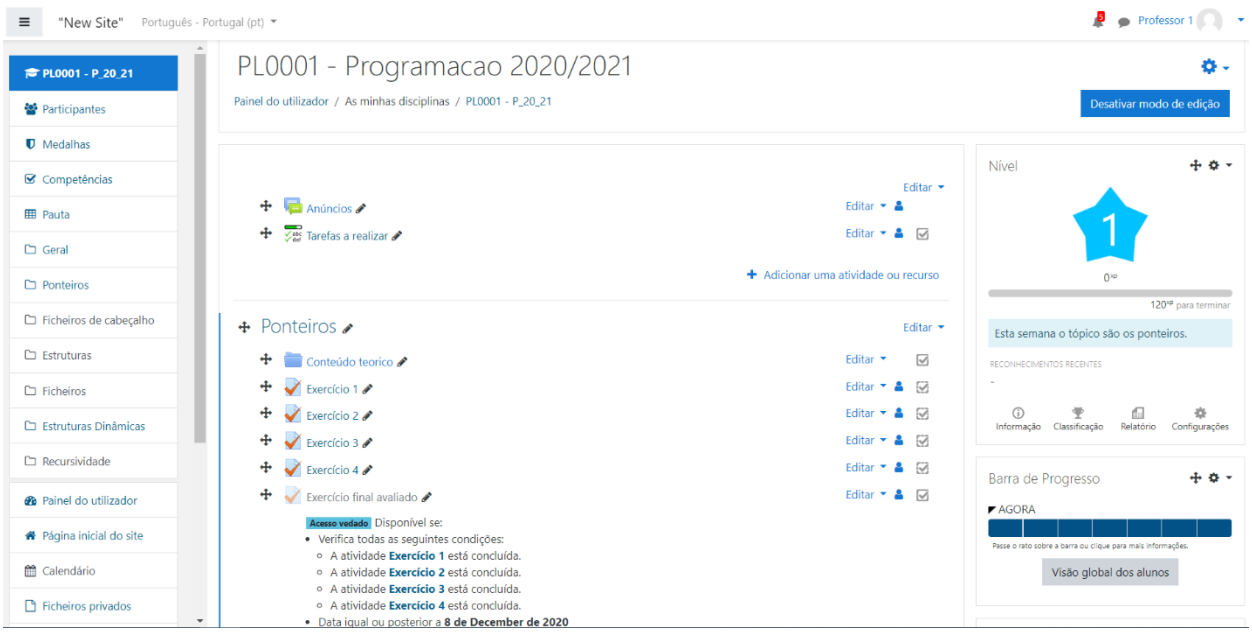


Figure 23 -Visão professor da disciplina *Moodle*

Anexo E: Moodle LMSOps V2.0 Release Notes

Índice Anexo E

1. Introdução

Esta versão pretende:

- Melhorar o acesso à plataforma.
- Permitir que os utilizadores recebam notificações sobre as atividades a decorrer;
- Permitir que os utilizadores customizem a plataforma.

2. Sobre a release

O objetivo desta entrega é melhorar a plataforma nos pontos menos positivos identificados na fase anterior da experiência.

3. Features

Foram adicionados os seguintes elementos:

3.1 Acesso à plataforma

Passa a ser possível aceder ao *Moodle* sem ser necessário ligação à *VPN* do ISEC. A plataforma está disponível em https://projmoodlemis.isec.pt/.

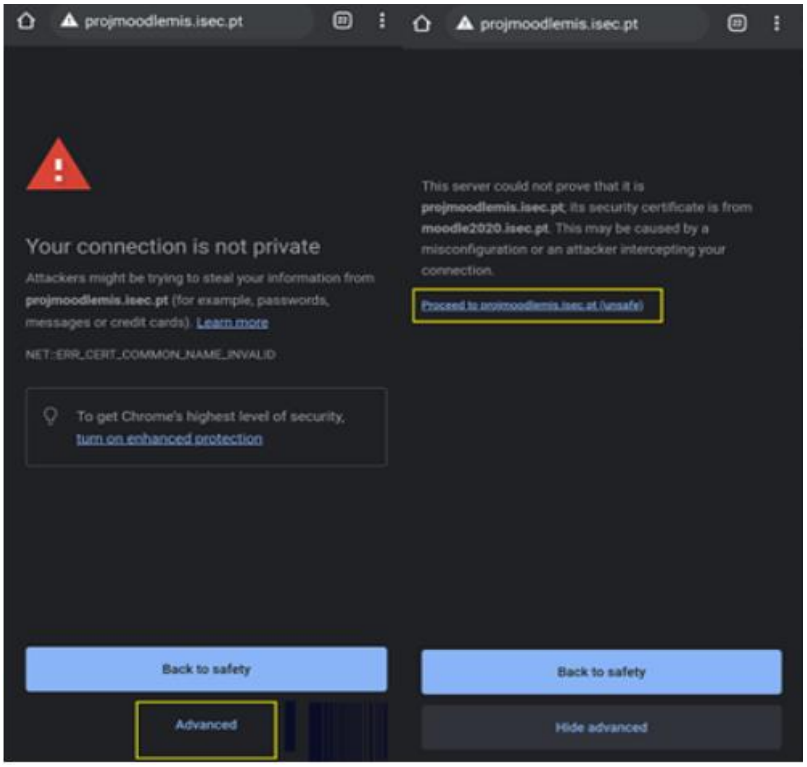


Figura 1 - Acesso a projmoodlemis.isec.pt

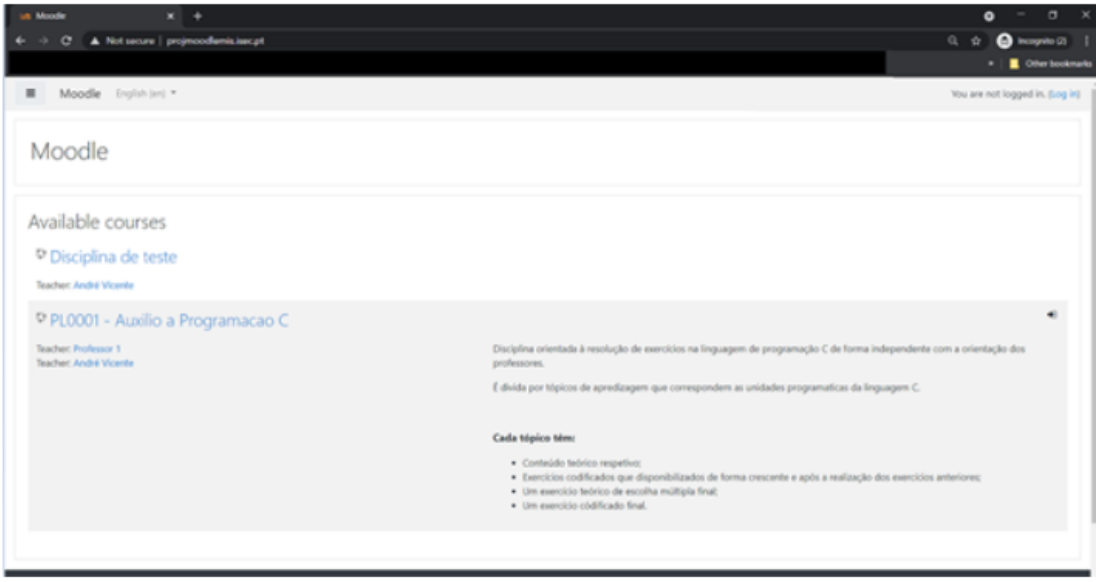


Figura 2 - Plataforma *LMSOps*

3.2 Notificações

Passa a estar disponível notificações via email onde é possível fazer o acompanhamento das atividades, das suas datas e das submissões. Caso exista alguma atividade em atraso será enviada notificação com um link para realizar a atividade.

Os emails serão enviados a partir do endereço 'Admin User (via Moodle) <afsvmail2@gmail.com> ' ou 'Do not reply to this email (via Moodle) afsvmail2@gmail.com'

Figura 3 - Notificação de data de exercício na plataforma

São também enviadas notificações por cada mensagem recebida no sistema de mensagens da plataforma.

Figura 4 - Notificação de mensagem na plataforma

Foram criadas contas de emails específicas para este projeto e cada conta de utilizador foi previamente configurada com uma das contas de email criadas. Os restantes dados de acesso à conta de email serão fornecidos através da plataforma.

3.3 Visualização

É possível alterar o tema da plataforma em Utilizador>Preferências>Editar perfil ou em https://projmoodlemis.isec.pt/user/preferences.php.

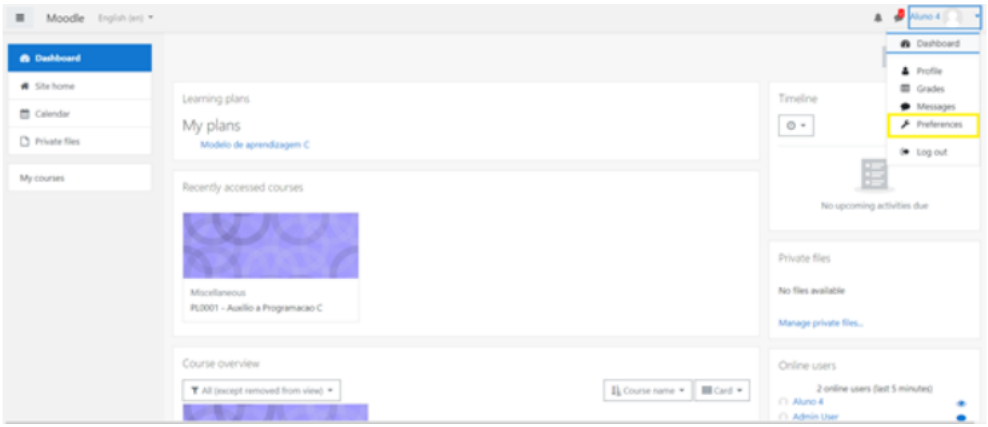


Figura 5 - Alterar tema da plataforma 1

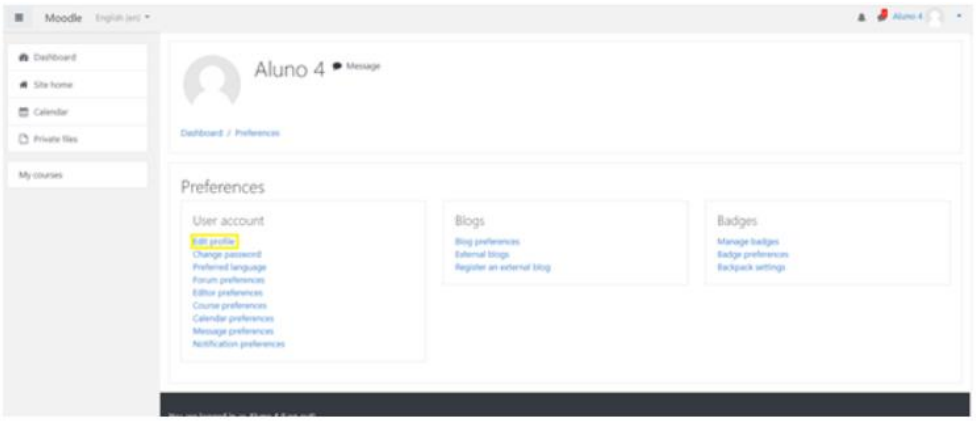


Figura 6 - Alterar tema da plataforma 2

Existem 2 temas que podem ser selecionados:

- *Default/Boost Campus* (tema *default*)

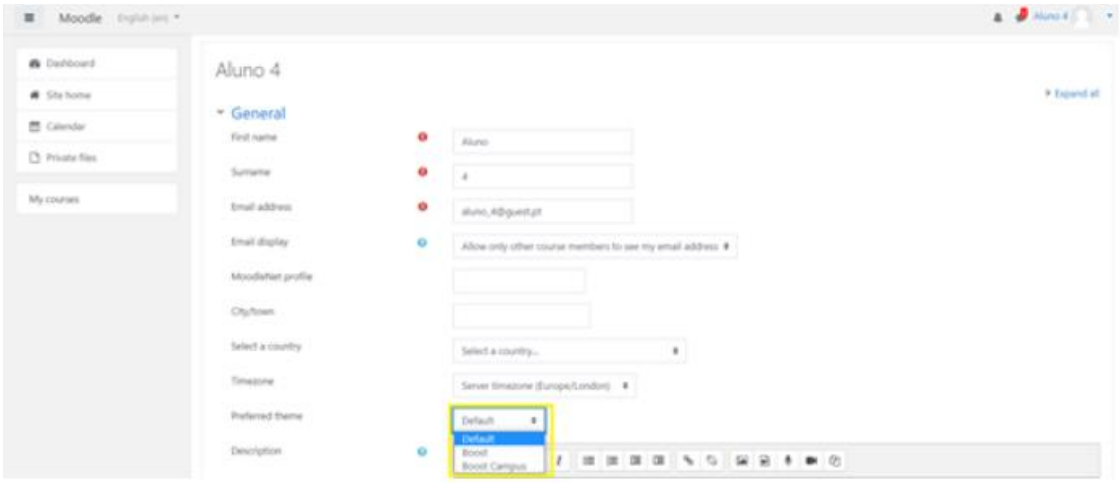


Figura 7 - Tema *Default/Boost Campus*

- *Boost* (tema *default* em *dark mode*)

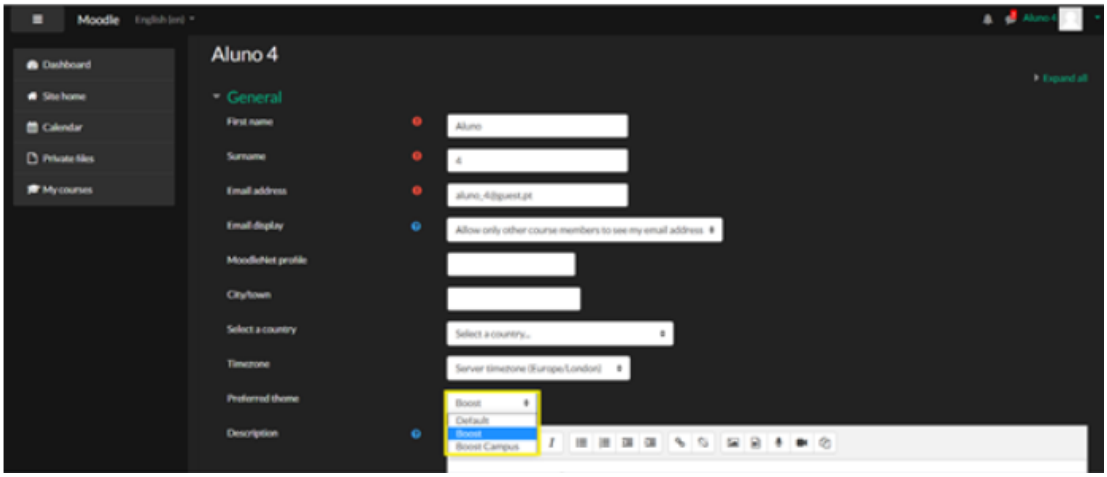


Figura 8 - Tema *dark mode*

Após selecionar o tema é necessário carregar em “Atualizar perfil” para o novo tema ficar definido.

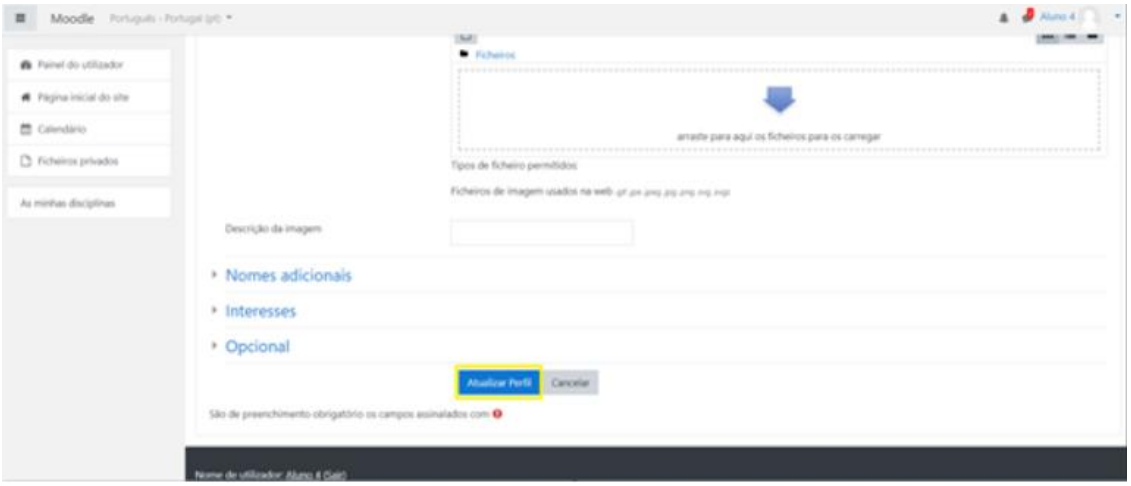


Figura 9 - Atualizar definições

Anexo F: Moodle LMSOps V3.0 Release Notes

Índice Anexo F

1. Introdução

Esta versão pretende:

- Melhorar a informação disponível relativa aos alunos e a resolução de exercícios, através de consultas a BD, outros mecanismos;

2. Sobre a release

O objetivo desta entrega é acrescentar um modulo quer permita melhorar a administração e visualização de estatísticas da plataforma e dos seus exercícios.

3. Features

Foram adicionados os seguintes elementos:

3.1 Plataforma de visualização de estatísticas *grafana*

A plataforma encontra-se disponível em http://10.84.128.180:3000/ apenas com a *VPN* do ISEC ligada.
Foram desenvolvidos 4 *dashboards* para ajudar no acompanhamento da resolução de atividades no moodle por parte dos alunos.

3.1.1 LMSOps_dashboard_indicators

Dashboard com elementos de visualização referentes aos acessos à disciplina e ao estado dos alunos em relação as atividades.

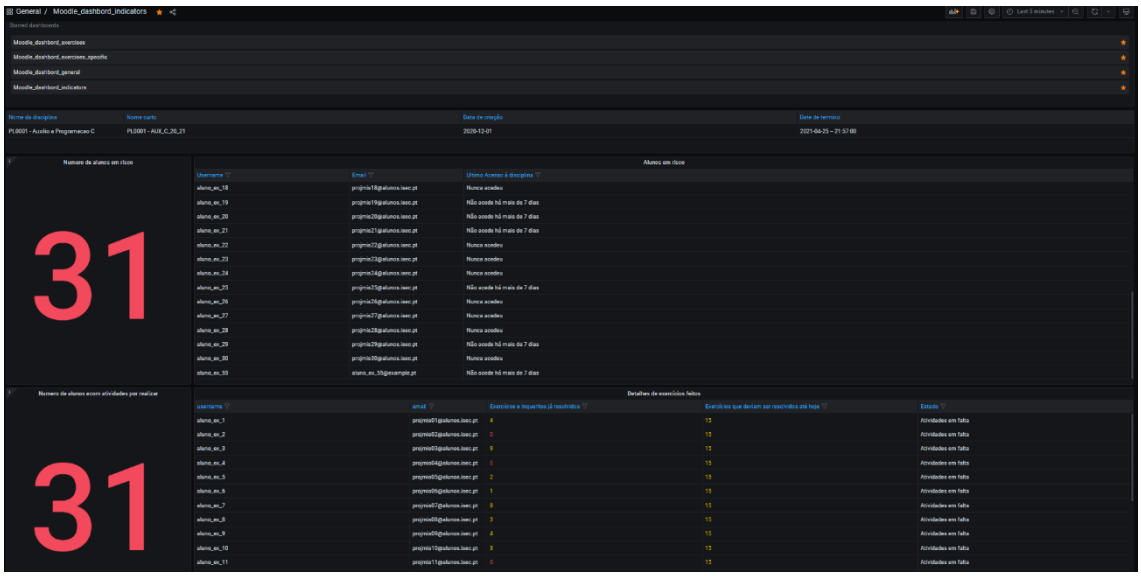


Figura 1 - *Dashboard* de indicadores da plataforma 1

É possível navegar para alguns submenus da plataforma através das descrições dos elementos de visualização.

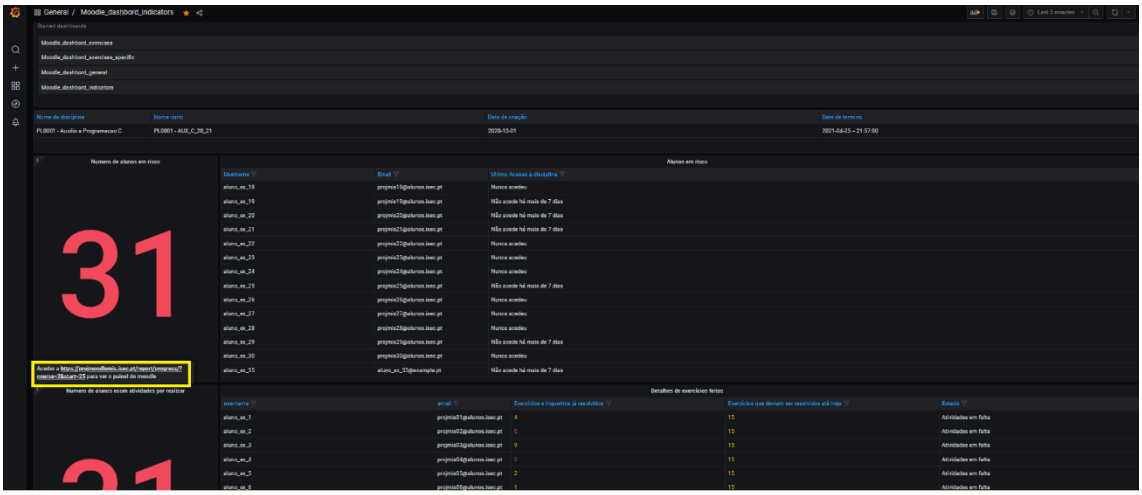


Figura 2 - *Dashboard* de indicadores da plataforma 2

3.1.2 LMSOps_dashbord_general

Dashboard com elementos de visualização referentes aos alunos inscritos na disciplina e aos exercícios/atividades existentes e o seu estado.

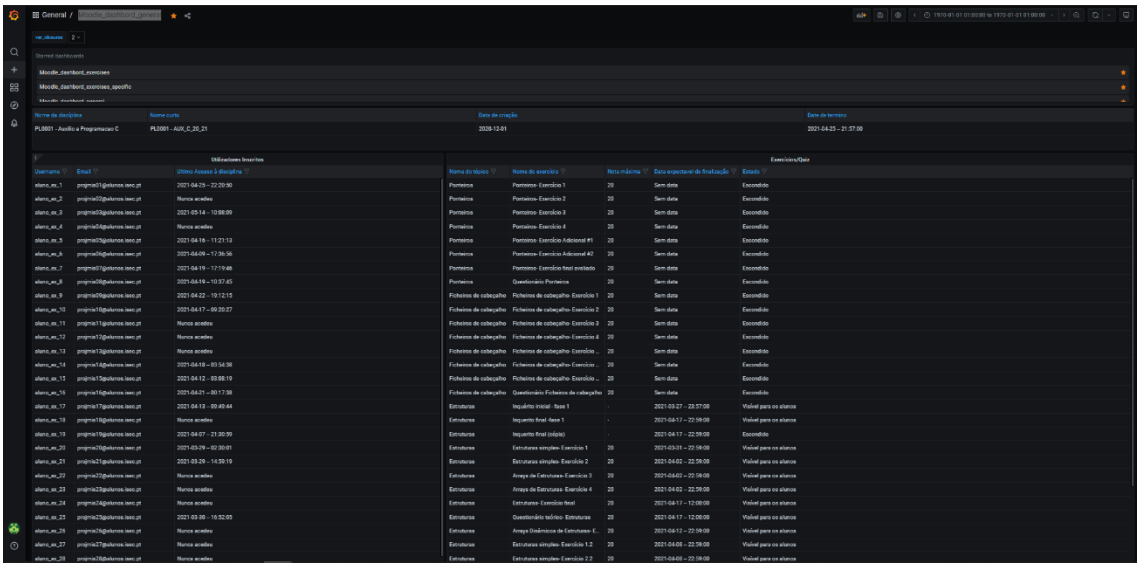


Figura 3 - *Dashboard* com dados gerais da plataforma

3.1.3 LMSOps_dashboard_exercises

Dashboard com elementos de visualização referentes as notas, tentativas e estatísticas dos exercícios que podem ser realizados pelos alunos.

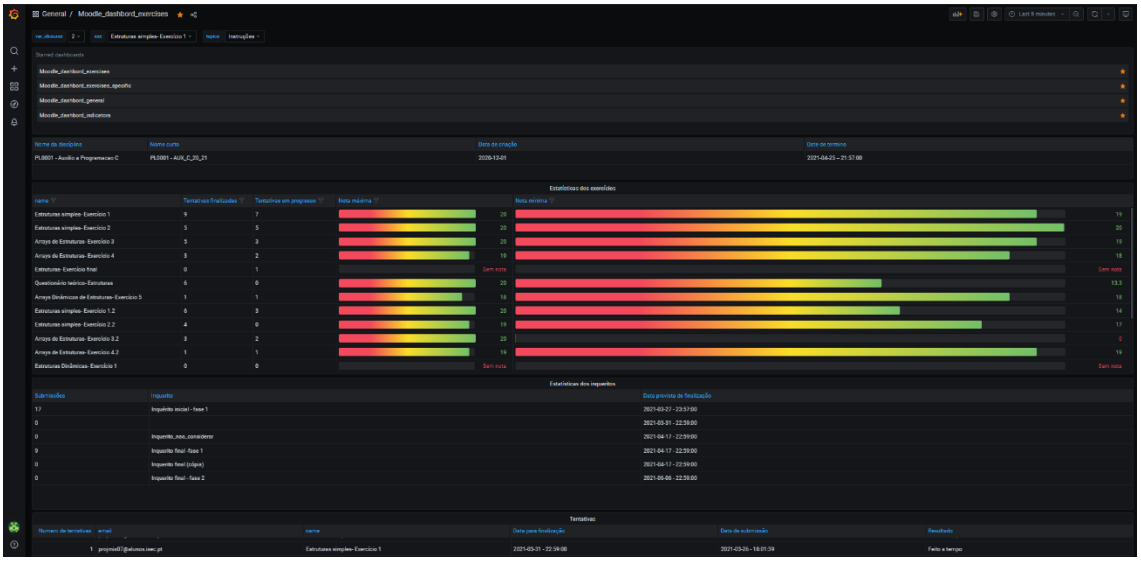


Figura 4 - *Dashboard* com dados dos exercícios

3.1.4 LMSOps_dashboard_exercises_specific

Dashboard com elementos de visualização referentes referente as estatísticas, tempos de resolução de cada exercício.

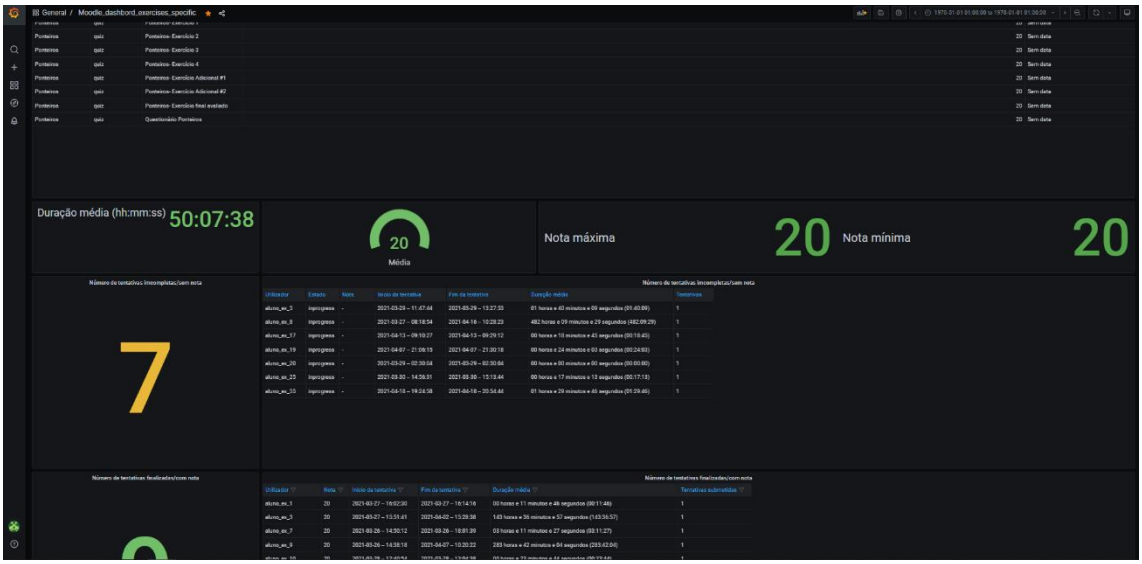


Figura 5 - *Dashboard* com dados específicos dos exercícios

Selecionar o exercício que se pretende ver as estatísticas em cima na variável *dashboard*.

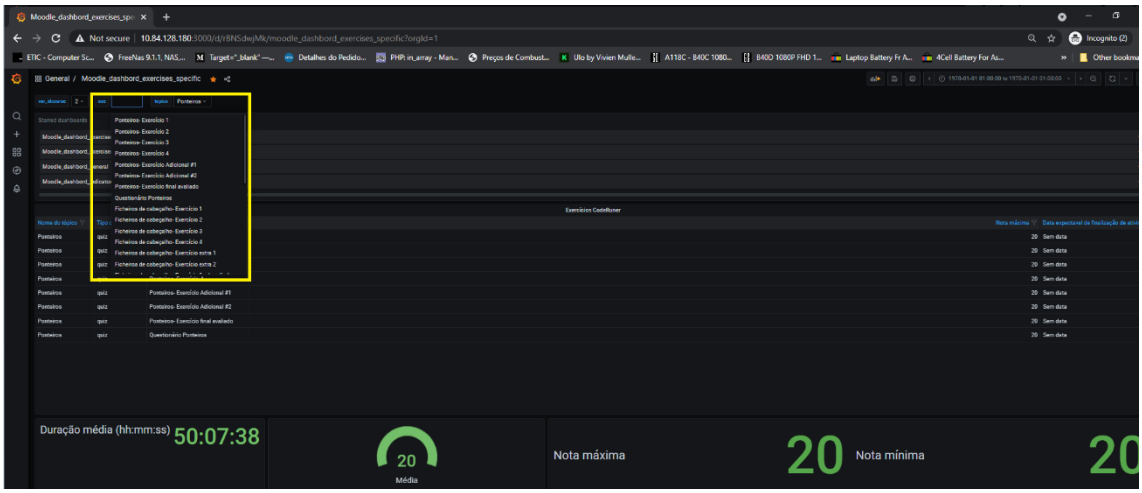


Figura 6 - Selecionar um exercício

Anexo G: Artigo

Applying the DevOps methodology for a more efficient process of teaching-learning computer programming

André Vicente
Polytechnic of Coimbra,
Coimbra Institute of Engineering
Rua Pedro Nunes – Quinta da Nora,
3030-199 Coimbra, Portugal
a21230112@alunos.isec.pt

João Cunha
Polytechnic of Coimbra,
Coimbra Institute of Engineering
Rua Pedro Nunes – Quinta da Nora,
3030-199 Coimbra, Portugal
jcunha@isec.pt

***Abstract*— Teaching and Learning programming languages in higher education, especially for the first-year students, is a major challenge due to two main factors: the ratio of students/teachers in these courses is usually very high, not allowing for a greater proximity between teachers and students; and these students have distinct backgrounds and logical reasoning, and therefore make progress at very different paces. Several studies show that the retention rates for such courses are high among the students in the introductory classes all across the globe. With the ever-increasing demand from the industry for professionals with programming skills, higher education institutions pressure the teachers for more positive results.**

The same demand for more efficiency and effectiveness is common in every sector of the industry, including software. DevOps is a software development methodology largely adopted by the software industry, that promotes the continuous improvement of the software development process. This methodology simplifies the problems across all development stages and promotes cooperation supported by the automation of tasks through tools and well-defined flows.

This paper presents a method that explores the possibility to improve the process of teaching-learning computer programming, by recurring to the principles of DevOps to create an improved approach able to individualize learning and reduce the teacher's effort. A Moodle-based platform was developed with several mechanisms that 1) automate tasks such as providing learning materials and exercises to each student, controlling the performance of the work, and providing feedback to student and teacher; 2) allow teachers to receive quick feedback and focus on what really matters, i.e., the difficulties of the students, the preparation of new learning resources for each student or group of students; and 3) let the students practice and study at their own pace. A pilot experience took place with one teacher and the students of an introductory course of Programming. Although preliminary and with the participation of a reduced number of students, the obtained results are promising.

Keywords—Teaching-learning computer programming, DevOps, Moodle, Teaching efficiency, Code Runner

I. INTRODUCTION

Computers, Smartphones, Tablets, TVs or even aeroplanes all have a set of computer programs integrated within, that execute certain functionalities. Computer programs can be defined as a set of instructions available on computing devices, that specify a certain set of tasks that follow an algorithm to achieve its purpose. Automation and information systems are increasingly entering in all sectors of society, increasing the need to program these systems. This can be perhaps one of the reasons why enrolments in computer science programs have largely surpassed other areas over the last years [1].

Several pieces of research from educational psychology suggest that teaching and learning are specific activities that depend on the topic. Therefore, learning computer programming has different challenges and techniques than learning for example physics, chemistry or even reading and writing. Computer engineering programming is a much younger discipline than others, therefore there are fewer studies or gathered evidence from the best strategies to teach and what in general work and what doesn't [2].

Programming is an "artistic" activity where programmers creatively apply patterns and algorithms for each problem they try to solve. This requires forming abstract representations of the process to resolve a problem and translating them into the correct code using one formal programming language, which represents two big challenges [3]. Considering these and other challenges that the students must face, it is correct to say that finding and implementing the appropriate methods to teach will be crucial for the process of learning [1][2]. Teachers apply different methods, but this process constitutes a difficult task that depends on the student. Each student learns at his own rhythm. Therefore, learners and teachers can benefit from using the correct and most adequate method [4]. These difficulties are visible in the quite high failure rates of the students regarding the introductory courses in computer programming. In the Bachelor in Informatics Engineering, from the Coimbra Institute of Engineering (ISEC in the Portuguese acronym), for example, there are failure rates of 42% in the curse of Introduction to Programming, 47% in Programming and 41% in Object-Oriented Programming [5, 6].

DevOps is an approach to accelerate the development of products or applications and maintain them, creating a stronger bond between the project parts (development and operations). The DevOps promotes shorter and controllable iterations through the adoption of practices, automation of simple and repetitive tasks and effective tools, focused on feedback from every part (project teams, stakeholders and clients). It is a methodology that allows fast delivery, and adaptation to the client and user needs.

This paper explores the possibility to adopt DevOps principles and practices to the process of teaching-learning computer programming. The aim is to make the process more efficient, by automating tasks such as delivering learning material and giving fast feedback to both students and teachers; and

more effective by allowing students to learn at their own pace, receiving orientations according to their own needs.

The remainder of this paper is organized as follows. Section II discusses the characteristics of teaching and learning computer programming. Section III introduces DevOps. Section IV discusses a possible approach to adapt DevOps to the teaching-learning programming with the help of a Moodle-based platform and some particular tools. Section V presents the experiment that took place with students from the Programming course. Section VI discusses the results and presents some conclusions.

II. Teaching and Learning Computer Programming

Students of computer programming, like any others, are influenced by several factors like passion, confidence, comfort, and satisfaction. Computer programming is a process that involves understanding the problem, designing an algorithm to solve it, writing code, testing, debugging and deploying. This requires more than mere familiarity and memorization strategies, like reusing solved exercises and learning the syntax of the programming language [4].

For the students, problem-solving practice is not easy. They face different difficulties on each problem in order to find a solution, each solution also being potentially different from the others. Each language used in programming has an abstract nature, as well as features and syntax that differ from the others. Sometimes even the programming environment is difficult to master, with several steps required to actually do any "programming" or run the code [7]. All these reasons cause many students to lose focus and motivation in the programming task [8]. Learning to program requires logical reasoning and a lot of training so it is therefore natural to have a big difference among the first-year students, who have different backgrounds and knowledge. Teaching to program involves teaching the syntax of the language and especially the creation of algorithms to solve problems that, depending on the level of knowledge and experience of the students, involves different approaches.

The lack of interest and confidence of the students in programming provides no satisfaction for the teachers, that must find approaches to engage and attract them to continue to program and overcome their difficulties [8]. Teachers and students can benefit from the use of the correct approach, which will improve the teaching and learning process making it more efficient [9]. A common strategy that the teacher apply is to define the learning objectives and define a roadmap with a study plan and exercises for the student to learn incrementally. Given constant feedback, the teacher can adjust the increments to the characteristics and progress of each student. This individualisation of teaching is the challenge that every teacher would like to be able to solve.

III. DevOps

The internet makes available many services, programs and applications to consumers and clients. With the growing needs of customers, these services need to be maintained and adapted by specialized teams. So, in addition to the development area that focuses on software design and development, there is an area that deals with the entire lifecycle of a product or service, its operation, improvement, and even its discontinuation and end of life [10]. To overcome competitors, it is necessary to put software in the market faster and faster, and to constantly adapt it to the customers' needs. This implies a simple and efficient process that connects the software development (Dev) and the operational (Ops) parts through workflow automation and rapid feedback.

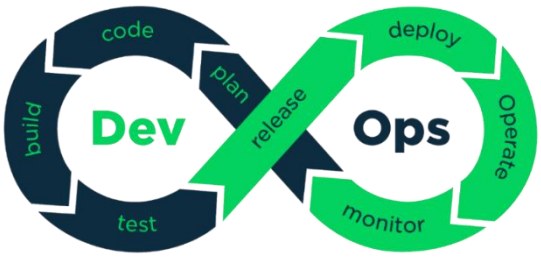


Fig. 1. The DevOps Cycle, source: https://www.hiclipart.com/free-transparent-background-png-clipart-pytym

The DevOps concept relies on principles and tools. Practices are defined and very well-tuned for automating tasks, collecting feedback, and conducting a very frequent delivery of products or service improvements, in an incremental and permanent loop [11]. Fig. 1 illustrates a typical DevOps cycle and the interconnection between its phases.

IV. DevOps and the Process of Teaching-Learning Computer Programming

A. *Applying DevOps principles to the teaching-learning of computer programming*

Although DevOps is a methodology specifically used in the software industry, where well-defined processes and tools assist the team, we argue that its principles can be used in other areas such as the process of teaching-learning programming.

When teaching programming, the teacher creates a plan that contains all the necessary elements of syntax, algorithm and structures to learn a programming language. This plan is divided into cycles that usually correspond to each topic of the programming language. Then, for each cycle, the teacher exposes the corresponding topics, provides learning materials and exercises, and tries to obtain output about the learning process, as well as the correct learning of the students. The aim is that the maximum number of students reach the level of competencies required for the curricular unit. However, feedback and adaptation of this method to each individual student is, in practice, only possible to reach with a small number of students.

The students, from their side, intend to learn as much as possible by studying and doing exercises (coding or others). Unfortunately, they don't usually give frequent feedback on their difficulties or needs (only through their assignments, such as solved exercises or tests), not allowing a rapid adaptation of the plan from the teacher.

As described in Section III, DevOps aims to make the product more adapted to the customers' needs, being essential to involve them and get their feedback, requirements, and priorities. In the process of learning to program, the client, i.e., the student, doesn't have the total vision of the learning plan or the necessary requirements, however, he knows his capacities as well as his difficulties. The teacher has visibility of the learning plan, however, not of the totality of the difficulties that are encountered by the students. This feedback is crucial to evaluate the results of the applied learning plans and improve them. However, it is not always obtained or analysed,

either by the enormous number of students or by their lack of study, due to low motivation, interest and attendance, or difficulty in solving exercises for the most diverse reasons.

Table I presents a comparison between what is done in the industry in each phase of the DevOps methodology and our adaptation of the process of teaching-learning of computer programming, supported by tools and automation. This process can consist of a set of iterations, as in an industry project, aligned and chained by topic, according to the plan defined by the teacher. The teacher starts by defining the goals and activities for that iteration (*Plan*), then the student plans his study (*Plan*), develops code and solves the provided exercises given by the teacher (*Code*). Then the student receives automatic feedback from his resolutions (*Build*) and checks if everything is correct (Test) before submitting them (*Release*). The Teacher then receives notifications about the submissions and may analyse each outcome (*Deploy*). At the end of the iteration, the teacher analyses the global results, provides feedback to the students (*Operate*) and adjusts the learning objectives for the next iteration for each student or group of students, according to their results (*Monitor*).

B. Platform and process to support the teaching-learning of computer programming with DevOps principles

Moodle is the Learning Management System (LMS) platform used at ISEC, allowing teachers to create courses to distribute documents and information to the students, as well as to create activities such as work submission, sending notifications, creating questionnaires, multiple-choice exercises, lessons, forums, calendars, score evaluation, or even allowing the exchange of messages between users. This platform, developed in PHP, is modular and allows the inclusion of additional functionalities through plugins, or the development of new modules. So, we have decided to use Moodle, and add or adapt the necessary plugins to create a course that could match the cycle and processes specified in Table I.

The objective would be to use the platform in the Programming courses, as an auxiliary tool to the classes. The platform was called LMSOps.

LMSOps was built using the following elements:

- Linux virtual machine with Ubuntu 20.04.1 LTS on the ISEC premises, for hosting the platform;
- Moodle version 3.9 (Build: 20200615);
- 'Tasks to be done' and 'Progress bar' plugins, visualization elements in the course that allow the students to check the path of exercises or tasks to be done until finishing the phase;
- 'Message my teacher' plugin, a visualization element of the course that allows the students to initiate direct contact with the teacher;
- ''Level' plugin, a visualization element of the course that assigns a level to each student according to the points he earns as he solves the exercises;
- 'Staff' tab, where the grades obtained in the exercises can be checked;
- 'Code Runner' plugin, a resource used to create the practical coding exercises (allows compilation of the code, tests to validate it and shows immediate feedback);
- 'Test/Exercise' resource for theoretical exercises with questionnaire type questions;
- Notifications for the teachers for each exercise resolution submitted by the students.

Fig. 2 illustrates the LMSOps platform interface.

TABLE IX. DEVOPS APPLIED TO TEACHING-LEARNING OF COMPUTER PROGRAMMING

	Industry DevOps	***Teaching -learning of computer programming***
lan	Requirements and feedback are collected from stakeholders and customers to create the product roadmap. The roadmap is then divided into tasks that are planned and prioritised in iterations, before the next one starts.	The teacher: Identifies needs. Draws up the study plan in topics and plans the next iteration. The student: Plans the study for the next topic based on the plan drawn up by the teacher.
Code	The previously planned tasks are coded and developed. Meetings are held to promote cooperation, inter-help and synchronisation of the status of the tasks taking place.	The student: Studies the elements provided by the teacher and solves the coding exercises, using the platform.
Build	After coding each task the code is integrated with the rest of the repository. This process is done after a manual review and approval by another team member. Automatic tests are then run over all the code with the objective of finding problems as soon as possible and informing the programmer so that he can solve them.	The student: Gets some automatic feedback from the tools installed in the platform (e.g. tests, static code analysis). May also ask the teacher for help or clarifications.
Test	The new code is placed in a testing environment. Acceptance tests, unit, security, exploratory and other tests are run.	The student: Checks if the result obtained in the exercise is as expected and meets the objective of the exercise. They obtain help from the teacher for any doubts they may have.
Release	The code is ready to be placed in the client's environment. This process can be manual or automatic depending on the control the team intends to have.	The student: Submits to the platform his work and data as completed, for evaluation. He can return to the Code phase until all the tasks defined by the teacher for this iteration are finished.
Deploy	In this phase, the code reaches the client's environment. It may still pass several client environments until it reaches the "Production" environment, depending on whether it is still distributed to other end-users. If necessary, new exploratory tests are made and deploy adjustments are made to try to avoid downtime.	The teacher: Analyses the reports generated by the tool about study and resolution times, exercise results and other outcomes. He may also manually analyse the exercises submitted.
Operate	In this phase, the operational parts of the new released version are guaranteed System metrics and feedback from the client and all product stakeholders are also collected.	The teacher: Analyses all students' progresses from a dashboard. Gives personalised feedback to each student, adjusting the next learning objectives.
Monitor	In this phase, the environment is supported and monitored based on pre-defined parameters in the feedback collected earlier.	The teacher: Based on the previous path of the students, defines new learning objectives for each student or group of students, by topic, and identifies study elements and defines tasks to be completed.

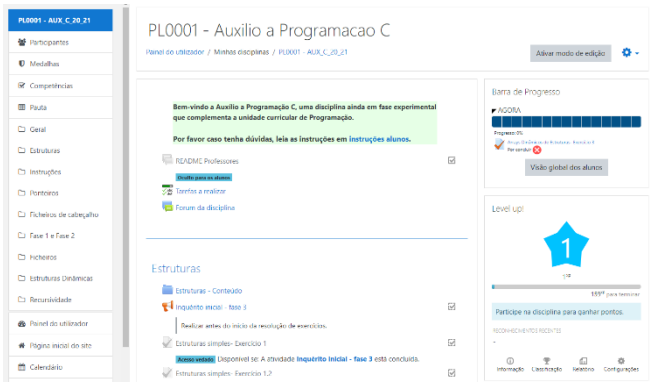


Fig. 2. LMSOps platform

V. TESTING THE PLATFORM

The methodology and platform were tested in the Programming course from the 1st year of the Bachelor in Informatics Engineering of ISEC.

The main goals defined for the experiment were the following:

- The target audience should be the students who were attending the course of Programming, therefore learning the C programming language;
- There should be a sample of about 30/40 students participating in the experiment;
- There experiment should have at least 2 phases, allowing to test improvements from one to the following phase;
- Distinct topics should be used in each phase;
- Feedback and data should be collected anonymously.

A. *Experimenting phase 1*

The first phase of the experiment with the LMSOps platform ran for 4 weeks and focused on the topic of Data Structures of the C programming language. Thirty accounts on LMSOps were made available to the Programming students who signed up for the experiment.

1) *Activities*

The tasks that the students had to fulfil featured:

a) 10 coding exercises, with tests to check their correct resolution and to give immediate feedback depending on the code submited and the assertion percentage on the resolution;

b) 2 surveys, one at the beginning with questions directed to the students' programming background and technical knowledge, and one at the end of the experiment (after the release phase when all exercises are complete) about the experience with the process and platform;

c) 1 theoretical exercise with multiple-choice questions about coding details.

2) *Analysis of the surveys*

17 students started the experiment (answered the 1st survey) and 9 finished it (the remaining 8 didn't submitted any exercise attempts). From their answers, we observed that:

a) 76% of the students that started the experiment already knew programming for more than 1 year. Only 41% said they enjoy programming and 18% don't even like it;

b) These students consider that their main difficulties are in understanding the problems (23%), in developing the algorithms (19%), in the language syntax (16%) and in coding (16%);

c) From the 9 students that finished the experiment, no one did all the exercises. However, all of them considered the platform intuitive and easy to follow, and the experience with Code Runner was very positive. 8 students considered their experience with this platform for learning Data Structures as very good or excellent;

d) The biggest issues reported by these students were the lack of clarity of some explanations, the automatic feedback, the misunderstanding of the problems they had to solve, and the access to the platform through a VPN (virtual private network).

3) *Analysis of the outcomes from the platform*

It was possible to extract the following outcomes from the participation of the students:

a) There were 42 attempts of resolution (Build phase). 30 of them were within the expected date;

b) All the submissions (Release phase) had a positive score (higher than 10, in a scale from 0 to 20);

c) The participation in the exercises decreased over time, i.e., the first coding exercises had more attempts (Build and Release phases) than the last ones. The theoretical exercise had though a higher participation;

d) All attempts submitted by the students have a positive average grade (higher than 10 in a scale form 0 to 20). Table II ilustrates the total attempts, finished attempts and average score in each exercise.

4) *Conclusions of Phase 1*

Although in general the students are very satisfied with this platform, some improvements are clearly identified as necessary, such as:

a) Recruit more volunteers;

b) Reduce the number of exercises so the students can finish the full phase;

c) Improve the quality of the exercises and the corresponding feedback;

d) Facilitate the contact with the teacher;

e) Improve the ability to access the platform;

f) Configure reminders and notification for the exercises and other interactions made in the platform.

B. *Experimenting phase 2*

Due to the lack of time to put the second experimentation phase of the LMSOps platform running during classes, it ran for 2 months during the normal and appeal exam periods. It focused on the topic of Simple Dynamic Structures of the C programming language.

Based on the experience, results, and conclusions of the first phase, some improvements were applied to the methodology and the platform, such as:

- The platform is now accessible through the internet, no longer requiring the use of ISEC's VPN;
- The number of exercises to complete is smaller, making it easier for the student to finish all the activities;

- The exercises and their contents, tests and feedback, have been revised and refined improving their quality;

TABLE II. PHASE 1 EXERCISES INFORMATION

Exercises	Total attempts	Finished attempts	Average 0-20
Simple structures -1	16	9	19.9
Simple structures -1.2	9	6	17.7
Simple Structures -2	10	5	20
Simple structures -2.2	4	4	18.3
Arrays of structures -3	8	5	19.8
Arrays of structures -3.2	5	3	12.3
Arrays of Structures -4	5	3	18.7
Arrays of Structures -4.2	2	1	19
Dynamic Arrays of Structures -5	2	1	18
Final Exercise	1	0	-
Theoretical Quiz	6	6	16.3

- All notifications, messages and exercise confirmations received through the platform are also sent by email to the email address of each account (teacher and student);
- Notifications for overdue exercises (according to the expected date of resolution configured in each exercise) to help students keep track of exercises.

This phase was made available to the 30 students who signed up for the first phase of the experiment.

1) Activities

a) 5 Code Runner exercises, with the corresponding tests to verify their correct resolution as well as appropriate feedback to each possible classification;

b) 1 final survey to be completed at the end of the experiment, directed to the learning experience in this second phase, comparing also with the experience of phase 1;

c) 1 theoretical exercise consisting of multiple-choice questions aimed at showing understanding of coding and compilation details.

2) Analysis of the outcomes from the platform and the survey

Four students started the exercises in this second phase. From the 5 coding exercises, 3 of the students didn't go further than the second. The only student that finished this phase did all the exercises with positive score in all the coding and exercises and theoretical exercise. Through these results, the following data was obtained:

a) The student that finished this phase skipped some steps, however, rated the platform, its the ease of use, clarity and feedback from the coding exercises with 8 or higher on a scale of 0 to 10;

b) This student highlights the importance of the coding exercises and considers this approach very positive for learning programming. However, he found the platform identical to the first phase noting none of the improvements made to its contents or structure.

c) The attempts submitted by the student have a positive average grade (higher than 10 in a scale form 0 to 20).

3) Conclusions of Phase 2

Although some improvements were made from the first to the second experiment, none was underlined by the students. Also due to the residual student participation, we have decided to extend the experiment to a third phase. The following improvement points were considered:

a) Enlarge the number of participants, opening the platform to students that did no participate in the previous phases;

b) Have the topics from the two previous phases available. Review the contents and reduce the extent of the exercises of the first phase;

c) Review and refine the instructions and steps present in the platform for using it.

C. Experimenting phase 3

The third phase of LMSOps experimentation took place for one and a half weeks during the special exams season with the contents of the first and second phases (Data Structures and Simple Dynamic Structures). New user accounts were made available for the students enrolled in this exam who were interested in training these topics. The dissemination of the platform and the distribution of the accounts was performed by the same teacher who accompanied the previous phases.

The main goal of this phase was to obtain greater participation, by reusing the previous materials. Anyway, all the contents of the platform were improved and revised according to all the feedback received previously.

1) Activities

The students had at their disposal:

a) 6 coding exercises on Data Structures and five exercises on Simple Dynamic Structures, with the corresponding tests and appropriate feedback;

b) 2 surveys, one at the beginning of this experiment with questions directed at the programming course, and the other at the end, regarding the learning experience and the platform;

c) 2 theoretical exercises, one on each of the topics.

2) Results and conclusions

Once more, only one student concluded this phase. We believe this was mainly because the students that participated in this exam could register up to the day before the exam date, therefore making their decision until the last moment.

The student who performed this phase presented similar results to those obtained from the second phase. He also highlighted the platform as intuitive and important regarding its purpose of studying and training programming.

D. General analysis and conclusions in the teacher's perpective

After the end of this last phase, we had a final meeting with the teacher responsible for the Programming curricular

unit, and who accompanied the learning process of the students through this platform, for a general analysis of the results and also to gather his opinion.

We (and especially the teacher), outlined the following strong points:

- This is an interactive tool that encourages the autonomous practice of programming, by offering exercises with increasing difficulty, and providing support material to help understand the subjects;
- It allows online compilation, with immediate validation and testing of the produced code. It enhances and accelerates the reward received by the students during self-study, as they immediately understand what they are doing and how it influences the outcome;
- Through the platform, the teacher understands how the exercises are progressing for each of the students. The students can ask for help and clarify doubts;
- The students can easily access the information and exercises available, being a tool very well organized and clear from the students' point of view;
- There is a lot of additional material available, complementing what is seen in the classroom context.

As weak points/possibilities for improvement, the following points were highlighted:

- The uptake in this experience was poor, but it was not a problem of the platform, or the learning strategy being tested. There was some alienation of the students, which can be partially justified by the fact that the experiments took place at a time when classes were exclusively held remotely (due to COVID-19). There was also a lack of a stronger link between the curricular unit and the tool, therefore the students regarded it as extra work, and didn't see any obvious advantage;
- The teacher's perspective needs improvement about the accomplishment of the tasks. It would be interesting to have a global view of the class status (although there is a global view of the students and some learning reports), with more detailed data;
- Customization to each student or group of students is still hard to obtain. The ideal route depends on the maturity/autonomy/degree of knowledge of the student. Some students may be unmotivated by tasks that are not very challenging, while others may feel they have a mountain too high to climb;
- Moodle has the advantage of being a simple and familiar resource for students but has a lack of flexibility to implement other kinds of strategies. Perhaps, students could feel more motivated if they found a different environment than usual.

VI. Conclusions

We believe that assisting the process of teaching and learning computer programming by drawing on the principles of industry methodology such as DevOps is possible and beneficial. Several approaches can be taken, not only in the concepts applied but also in the tools used. The approach followed throughout this project was essentially based on the adaptation of a standard industry DevOps process and the creation of a platform suited to teachers and students. Although the participation in the experiment felt short of what was expected leading to few and narrow conclusions, there are positive points and assumptions that we can take into consideration from this experimentation.

There was some enthusiasm from the students that love to program and that participated in the experiment, in the sense of feeling defied with the coding exercises. The use of theoretical exercises also attracted the participation of the students, challenging them with less common coding situations, helping them to see and understand different ways of programming and peculiar coding solutions. It is necessary to have a stronger link between the curricular unit/course and the platform, so more students can participate.

From the teacher's perspective, this platform has still a long way to go, facilitating the visualization of the results and adapting the learning objectives and exercises to each student or group of students. This tool is, for now, especially focused on the students' perspective.

As a final point, there is a huge room for improvement both in the process and in the functionalities that the platform offers. Anyway, we believe that applying DevOps principles to teaching and learning computer programming can be a strong added value for students to be able to practice programming in any programming subject, drawn with gamification and strong feedback, as well as for teachers, that benefit from the automation of many tasks and saving time for designing and adapting the students' goals and learning paths.

ACKNOWLEDGMENT

The authors would like to thank Professor Francisco Pereira and all the students who participated in this experience.

References

Sanja Mohorovicic and Vedran Strcic. (2011). "An Overview of Computer Programming Teaching Methods,". n.d.

Brown NCC and Wilson G. (2018). Ten quick tips for teaching programming. PLoS Comput Biol 14(4): e1006023. https://doi.org/10.1371/journal.pcbi.1006023.

Szlávi, Péter and Zsakó, László. (2003). Methods of teaching programming. Teaching mathematics and Computer Science. 1. 247-258. 10.5485/TMCS.2003.0023.

Mohorovicic, Sanja and Tijan, Edvard. (2010). New technologies in teaching university level programming. n.d.

ISEC. (2019). "SIGQ IPC". [Online]. Available: https://sigq.ipc.pt/system/files/RAC_201819_ISEC_9885_26.pdf.

ISEC. (2019). "SIGQ IPC". [Online]. Available: https://sigq.ipc.pt/system/files/RAC_201819_ISEC_9770_66.pdf.

Rashid, Tarik and El Farra, Rana. (2011). Assessing Teaching Methods in Programming A Case Study: Salahaddin University. The 4th International Scientific Conference of Salahaddin University-Erbil, October 18-20, 2011.

Papp-Varga, Zsuzsanna & Szlávi, Péter & Zsakó, László. (2008). ICT teaching methods–Programming languages. Annales Mathematicae et Informaticae. 35. 163-172.

A. Gomes, W. Ke, S. K. Lm, A. Siu, A. J. Mendes and M. J. Marcelino, "A teacher's view about introductory programming teaching and learning — Portuguese and Macanese perspectives," 2017 IEEE Frontiers in Education Conference (FIE), 2017, pp. 1-8, doi: 10.1109/FIE.2017.8190493.

C. J. J. P. a. N. R. M. Betsy Beyer. (2017). "Site Reliability Engineering," [Online]. Available at: https://sre.google/sre-book/table-of-contents/.

D. S. E. TEAM and “djangostars”. (2017). [Online]. Available: https://djangostars.com/blog/what-is-devops/.Mohorovi, Sanja and Vedran Str. “An Overview of Computer Programming Teaching Methods.” (2011).