LISBON SCHOOL OF ECONOMICS & MANAGEMENT
UNIVERSIDADE DE LISBOA

MESTRADO

MÉTODOS QUANTITATIVOS PARA A DECISÃO ECONÓMICA E EMPRESARIAL

TRABALHO FINAL DE MESTRADO

PROJETO

APLICAÇÃO MÓVEL COMO SOLUÇÃO PARA A APRENDIZAGEM DE PROGRAMAÇÃO

RAFAEL GONÇALO MATOS OLIVEIRA

LISBON
SCHOOL OF
ECONOMICS &
MANAGEMENT
UNIVERSIDADE DE LISBOA

OUTUBRO - 2019

LISBON SCHOOL OF ECONOMICS & MANAGEMENT
UNIVERSIDADE DE LISBOA

MESTRADO
MÉTODOS QUANTITATIVOS PARA A DECISÃO ECONÓMICA E EMPRESARIAL

TRABALHO FINAL DE MESTRADO
PROJETO

APLICAÇÃO MÓVEL COMO SOLUÇÃO PARA A APRENDIZAGEM DE PROGRAMAÇÃO

RAFAEL GONÇALO MATOS OLIVEIRA

ORIENTAÇÃO:
CARLOS COSTA

LISBON
SCHOOL OF
ECONOMICS &
MANAGEMENT
UNIVERSIDADE DE LISBOA

OUTUBRO - 2019

GLOSSÁRIO

HTML – Hypertext Markup Language.

MIT – Massachusetts Institute of Technology.

SWOT – Strengths, Weaknesses, Opportunities and Threats.

UC – Unidade Curricular.

URL – Uniform Resource Locator.

RESUMO

Saber programar é uma competência considerada cada vez mais relevante. Porém, aprender a programar é difícil, exige esforço e motivação. Neste contexto, importante identificar quais as soluções para contribuir para melhorar o desempenho de quem quer aprender a programar. Para responder a esta questão começou por se realizar a revisão de literatura. Enumera os principais fatores que influenciam a aprendizagem e o ensino de programação e ainda propostas para solucionar os mesmos. Analisadas as propostas de solução, é proposto como solução uma aplicação móvel. Ainda da revisão da literatura foi possível derivar um modelo concetual. Sendo ainda possível identificar as principais funcionalidades. Assim, são detalhadas as particularidades da aplicação bem como a sua implementação. Por fim é produzida uma avaliação preliminar onde são identificados os pontos fortes e fracos da aplicação.

PALAVRAS CHAVE: ensino e aprendizagem de programação; estilos de aprendizagem; Python; robôs virtuais.

ABSTRACT

Knowing how to program is an increasingly relevant skill. However, programming is difficult, requires effort and motivation. In this context, it is important to identify what are the solutions to help improve the performance of those who want to learn to program. To answer this question, we start by conducting a literature review. This allows us to list the main factors that influence the learning and teaching of programming and still apply to solve them. Analysing the literature, we propose a solution: a mobile application. The literature review also allows us to derive a conceptual model and identifying the main features. Then we implemented a prototype. Finally, we performed a preliminary assessment, which identifies the strengths and weaknesses of the application.

KEYWORDS: programming learning and teaching; learning methods; Python; virtual robots.

ÍNDICE

Índice de Figuras

Índice de Tabelas

AGRADECIMENTOS

Em primeiro lugar, agradeço ao professor Carlos Costa. O seu apoio e persistência foram fundamentais para o bom rumo deste projeto.

Em segundo, agradecer à Mafalda Sousa Santos e ao Pedro Carapau por terem integrado a amostra de pessoas que avaliou esta a aplicação. As suas sugestões e palavras foram bastante importantes.

Quero também agradecer à minha namorada, Cristina Filipciuc, por me ter apoiado incondicionalmente nos momentos menos bons e por também ter contribuído para a avaliação desta aplicação.

Por último, agradecer aos meus pais. Sem o seu apoio e, acima de tudo, sem a educação que me transmitiram, nada disto seria possível.

1. INTRODUÇÃO

Nos tempos correntes é cada vez maior a necessidade de pessoas qualificadas na área da programação. Tal deve-se maioritariamente aos novos desafios da era digital bem como à evolução tecnológica de grande parte das áreas de negócio. No entanto existe uma falta de programadores competentes e qualificados para abranger os diversos projetos que vão surgindo nas tecnologias de informação (Pereira, Costa e Aparício, 2017). Este facto está diretamente relacionado com as dificuldades de aprendizagem e ensino de programação por parte dos estudantes e professores.

Da parte dos estudantes os principais problemas apontados são a falta de motivação demonstrada, o pouco tempo despendido na aprendizagem, a falta de ginástica mental entre outros. Por sua vez, os professores apresentam como alguns dos potenciais problemas a prevalência de problemas teóricos sobre os problemas práticos e o ensino de conceitos dinâmicos através de materiais estáticos (como apresentações por slides) (Pereira, et al., 2017, Ferreira, Costa, Aparício e Aparício, 2017 e Gomes e Mendes, 2007). Identificadas estas adversidades é necessário estudar novas abordagens de ensino e aprendizagem que tragam mais vantagens e, acima de tudo, resultados a todos os envolvidos. Como tal, será proposta uma solução de ensino de programação, orientada para os estudantes, através de uma aplicação móvel.

Neste contexto, o trabalho de investigação aqui apresentado parte de uma questão relevante: como ultrapassar a dificuldade de aprendizagem de programação.

Propõe-se como objetivo contribuir para a resolução deste problema utilizando aplicações móveis.

A abordagem metodológica está dividida em quatro partes: revisão da literatura existente, proposta conceptual, implementação e avaliação preliminar. A revisão da literatura subdivide-se em dificuldades na aprendizagem e ensino de programação e propostas de soluções. A proposta conceptual apresenta uma solução para ultrapassar algumas das dificuldades apontadas na revisão da literatura. A implementação descreve o processo de elaboração da proposta conceptual e a avaliação preliminar incorpora um conjunto de opiniões e pareceres sobre a proposta apresentada.

2. Revisão da Literatura

2.1. Dificuldades na aprendizagem e ensino de programação

Existem vários fatores que podem influenciar o ensino e aprendizagem de uma linguagem de programação. Dividir-se-á esta secção em duas subsecções: dificuldades apresentadas pelos estudantes e dificuldades apresentadas pelos professores.

2.1.1 Dificuldades percecionadas pelos estudantes

Os principais problemas apontados são os métodos de estudo, a falta de motivação e as capacidades dos estudantes (Ferreira, et al., 2017 e Gomes e Mendes, 2007).

Geralmente os estudantes acham difícil aprender uma nova linguagem de programação, uma vez que são confrontados com termos característicos de programação e existe a necessidade de visualizar os processos envolvidos. Como resultado disto os estudantes tentam memorizar os conceitos e processos ao invés de os compreender (Pereira, et al., 2017). Este método não é o mais adequado para a aprendizagem de uma linguagem de programação. Programar requer um método de estudo diferente. O estudo da programação deve basear-se em exemplos práticos e exercícios intensivos, com o objetivo de resolver o maior e mais diverso número de exercícios (Gomes e Mendes, 2007).

Num inquérito desenvolvido por Piteira e Costa (2013), estudantes e professores avaliaram diversos materiais de estudo pelo seu grau de utilidade na aprendizagem de programação (Figura 1). Os materiais avaliados foram: livros de programação (M1), notas das aulas (M2), exemplos de código (M3), imagens de estruturas de programação (M4), visualizações interativas (M5), tutoriais (M6), vídeos disponíveis no Youtube (M7), páginas de Twitter relacionadas com programação (M8), páginas de Facebook relacionadas com programação (M9), cópias de acetatos (M10), conteúdos disponíveis na página da UC (M11). Os alunos classificaram os exemplos de código, os tutoriais, vídeos no Youtube e os conteúdos disponíveis na página da UC como sendo os mais benéficos, contrariamente às visualizações interativas e às páginas de programação nas redes sociais. As avaliações dos professores foram em linha com as dos estudantes, excetuando os livros de programação, as notas das aulas, os exemplos de código e as imagens de estruturas de

programação, que foram consideradas menos úteis comparativamente com as avaliações dos estudantes (Figura 1).

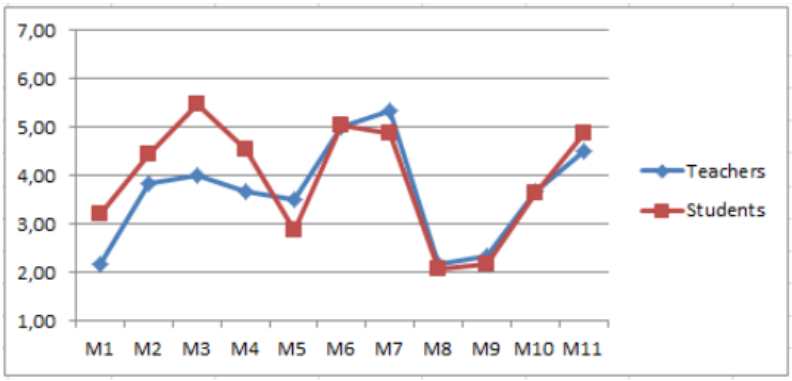

FIGURA 1 –Dificuldades percebidas (Piteira e Costa, 2013).

As capacidades e as atitudes dos estudantes também são apontadas como um problema recorrente na aprendizagem de programação. Esta falta de capacidade dos estudantes está muito associada à falta de aptidão para a resolução de problemas. Frequentemente os estudantes não compreendem com clareza qual é o problema em questão. Tal deve-se às fracas capacidades de interpretação dos alunos ou à ansiedade para começar a escrever o código sem ler corretamente os enunciados (Gomes e Mendes, 2007). Outra das razões costuma ser a pouca capacidade para relacionar conceitos e processos, o que está correlacionado com o facto de os estudantes interiorizarem contextos específicos sem entenderem os conceitos gerais (Pereira, et al., 2017 e Gomes e Mendes, 2007). A falta de motivação é talvez o maior problema dos alunos na aprendizagem de uma linguagem de programação. O facto de as Unidades Curriculares de programação serem conotadas como difíceis faz com que os alunos entrem nas salas de aula com uma ideia negativa sobre os conteúdos, causando desmotivação (Gomes e Mendes, 2007).

A natureza da programação também é algo que influencia a aprendizagem. As linguagens de programação têm uma sintaxe muito complexa que complica a aprendizagem aos iniciantes em programação. Os maiores problemas surgem mesmo nas fases iniciais de aprendizagem, quando é esperado que os aprendizes percebam e apliquem conceitos abstratos (como estruturas de controlo) que são difíceis de compreender e não têm uma conexão direta com uma representação do mundo real (Piteira e Costa, 2013 e Gomes e Mendes, 2007).

2.1.2. Dificuldades percecionadas pelos professores

No que respeita aos métodos de ensino o principal problema é a utilização de materiais estáticos para o ensino de conceitos dinâmicos. Este método de ensino através de quadros negros e livros gera um grande número de desistências e uma consequente perda de interesse por parte dos alunos (Pereira, et al., 2017 e Gomes e Mendes, 2007).

Outra grande contrariedade é o tempo limitado existente para a aprendizagem de uma linguagem de programação. Os planos curriculares têm tempos pré-determinados, o que causa uma maior pressão sobre os estudantes, visto que cada indivíduo tem o seu próprio ritmo de aprendizagem (Ferreira, et al., 2017 e Gomes e Mendes, 2007)

Os próprios professores têm algumas lacunas no ensino de programação. O ensino por parte de um professor espelha, frequentemente, o seu método de aprendizagem, o que, nem sempre, é benéfico para os estudantes na medida em que cada aluno aprende de maneira diferente e métodos distintos podem ser benéficos para alunos distintos (Gomes e Mendes, 2007).

Gomes e Mendes (2007) também apontam como obstáculo ao ensino o facto de os professores privilegiarem o ensino de uma linguagem em particular e as suas particularidades em vez de promoverem a resolução de problemas usando essa mesma linguagem. Esta abordagem beneficiaria o desenvolvimento das capacidades de programação.

2.2. Soluções para ultrapassar as dificuldades na aprendizagem de programação

Aparício e Costa (2018) apresentaram um conjunto de possíveis soluções para ultrapassar as dificuldades na aprendizagem de programação (Tabela 1).

SOLUTIONS	SUPPORT	MOTIVATION
Gamification		X
Use of physical robots,		X
Use of virtual robots,		X
Graphic languages	X	
Simplified languages	X	
Game Design		X
Web-based programming learning environment	X	
Program visualization tool	X	
Instructional technology	X	
Visualization tool	X	

TABELA 1 – Soluções para ultrapassar dificuldades (Aparício e Costa, 2018).

Algumas das soluções apresentadas baseiam-se em soluções tangíveis (e.g. Aparicio et. al., 2019). Outras traduzem-se em robots virtuais (exemplo, Aparicio e Costa, 2018, Aparicio, Aparicio e Costa, 2018, Costa, Aparicio e Cordeiro, 2012). O uso de gamificação, a utilização de linguagens de programação simples e fornecimento de *feedback* instantâneo são outras soluções propostas (Pereira, et al., 2017, Piteira e Costa, 2017 e Aparício e Costa, 2018).

2.2.1 Robôs Virtuais

Papadakis e Orfanakis (2018) defendem que as dificuldades dos estudantes na aprendizagem de programação podem ser ultrapassadas transformando um curso introdutório de programação numa experiência fácil, divertida e interativa, o que resulta num aumento do interesse e uma melhoria dos resultados na aprendizagem de programação.

A utilização de ferramentas de visualização como robots virtuais pode alavancar a aprendizagem de modelos abstratos e as suas aplicações em programas, o que resulta numa nova perspetiva sobre os conhecimentos adquiridos e pode melhorar o comportamento e motivação sobre uma linguagem de programação (Costa, et al., 2012). Os robots virtuais apresentam um ambiente controlado, o que conciliado com uma linguagem de uso geral e de fácil aprendizagem promove o maior foco na aprendizagem da programação (Aparício e Costa, 2018). Vihavainen, Airaksinen e Watson (2014)

elaboraram um estudo em que introduziram diferentes metodologias no ensino de um determinado curso de programação e concluíram que a utilização de ferramentas de programação virtuais aumentou a taxa de sucesso em cerca de 17,3%.

2.2.2 Gamificação

A gamificação é um método emergente que começou a ganhar notoriedade em 2010. Foi criado em 2002 e, segundo Deterding, Dixon, Khaled e Nacke (2011), é o processo de incorporar elementos de jogo em contexto de não jogo, com o objetivo de melhorar a experiência e aumentar os níveis de motivação e participação dos utilizadores (Pereira, et al., 2017 e Piteira e Costa, 2017). Estes elementos de jogo vão desde medalhas e utilização de pontos até à atribuição de níveis, o que potencia o envolvimento do aluno na aprendizagem ao mesmo tempo que fornece *feedback* (Piteira e Costa, 2017).

Os videojogos são um assunto do quotidiano das novas gerações. Têm como finalidade entreter os utilizadores e devem cativá-los e motivá-los a manterem-se numa atividade intensa durante longos períodos de tempo, porém não trazem uma utilidade direta pura e dura (Pereira, et al., 2017). Se introduzirmos no ensino de programação, por meio de gamificação, o tipo de sensações que um videojogo pode provocar, o ensino tornar-se-á mais apelativo e motivador (Deterding, et al., 2011 e Pereira, et al., 2017). Este método está em crescimento exponencial pois existe a convicção que melhora os processos de aprendizagem (Pereira, et al., 2017) e, consequentemente, os resultados: a aplicação de técnicas de gamificação em contexto de ensino de programação melhora os resultados obtidos pelos estudantes em cerca de 11% (Vihavainen, et al., 2017).

2.2.3 Feedback

Segundo Piteira e Costa (2013) os alunos consideram materiais dinâmicos (eg. código, tutoriais e vídeos) mais vantajosos do que materiais estáticos (eg. livros e slides). Referem também, tal como Ferreira e colegas (2017), que o ensino de uma linguagem de programação deve ser focado na aplicação de conhecimentos ao invés de se cingir à armazenagem dos mesmos. Como tal, e para que a evolução dos estudantes possa ter um efeito tangível, alguns autores defendem que o fornecimento de *feedback* pode beneficiar a aprendizagem de uma linguagem de programação pois está diretamente relacionado com a aquisição de conhecimentos, visto que possibilita aos estudantes a aprendizagem através dos seus erros e proezas (Pereira, et al., 2017 e Papadakis e Orfanakis, 2018).

2.3. Síntese

Após analisar-se as dificuldades e soluções propostas, considerou-se que os aspetos especialmente relevantes num curso de programação para iniciantes são a utilização de linguagens de programação com sintaxes menos complexas (Gomes e Mendes, 2007), a aplicação de uma interface virtual de modo a tornar a experiência de aprendizagem fácil e interativa (Papadakis e Orfanakis, 2018) e ainda a disponibilização de feedback, de forma a que os estudantes possam aprender com os erros (Pereira, et al., 2017 e Papadakis e Orfanakis, 2018).

3. PROPOSTA CONCEPTUAL

Apresentadas as principais dificuldades na aprendizagem de programação e algumas propostas para suprimir essas dificuldades, este relatório de projeto tem como principal objetivo conceber um método de ensino de programação que vá ao encontro das soluções apresentadas. Como tal, serão definidas duas questões às quais se pretende dar resposta neste capítulo: Como se pode ajudar um aluno a aprender a programar mais facilmente? Qual a linguagem de programação a utilizar nesse processo?

Após analisar as diferentes propostas de soluções entendeu-se que, para otimizar os resultados no processo de aprendizagem de programação, dever-se-ia desenvolver uma aplicação móvel que incluísse uma interface, um curso de programação para iniciantes e que forneça *feedback* instantâneo (Figura 2)

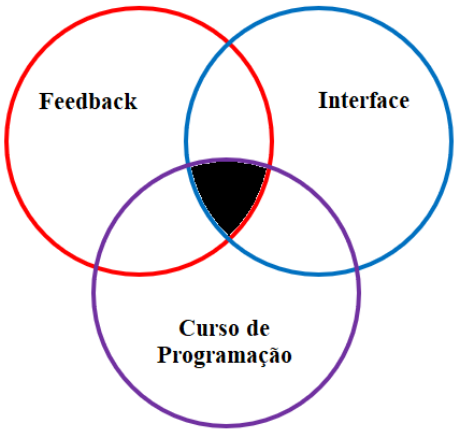

FIGURA 2 – Solução Proposta.

Ferreira e colegas (2017) defendem que um programador aprendiz tem de estudar os conceitos e paradigmas de uma linguagem para compreender e escrever código mais complexo. Como tal a aplicação móvel deverá estar dividida por aulas. Cada aula corresponderá a um conjunto de temas e conceitos, que devem ser ordenadas progressivamente pelo seu grau de dificuldade e complexidade (Pereira, et al., 2017). Assim a estrutura do curso de programação deve englobar os conceitos basilares de uma determinada linguagem de programação, isto é, variáveis, operadores, estruturas de controlo, objetos e funções.

Segundo Piteira e Costa (2013) os estudantes de programação preferem fazer exercícios práticos em oposição a explicações teóricas. Por conseguinte, cada aula teórica

deverá ter um momento prático, ou seja, a aplicação deverá ser desenvolvida de modo a implementar uma vertente prática, onde os utilizadores possam exercitar os conceitos assimilados em cada aula. Esta vertente será criada através de uma interface dividida em duas partes: a primeira parte será constituída por uma janela de comandos onde os utilizadores podem testar código para resolver um conjunto de exercícios propostos para cada tema. A segunda parte corresponde ao *output* do código criado pelos utilizadores, isto é, o *feedback* devolvido pela interface. O *output* permitirá aos aprendizes desenvolverem as suas capacidades, uma vez que lhes possibilita a aprendizagem através dos seus erros (Papadakis e Orfanakis, 2018).

Definido o método a utilizar resta encontrar a linguagem de programação adequada para integrar a aplicação móvel. Aqui um aspeto a ter em linha de conta será a usabilidade (Costa, Silva, & Aparício, 2007).

Para aprendizes de programação, a primeira linguagem a ser assimilada deve ser uma linguagem simples e que dê um conhecimento básico e a experiência necessária para a resolução de problemas (Avouris, 2018).

A linguagem a utilizar deve, para além de ser concisa e clara de forma a facilitar a aprendizagem dos utilizadores, ser uma ferramenta que seja aplicada no mundo laboral. Deste modo, os alunos terão uma perceção da utilidade do curso, o que pode aumentar o seu grau de satisfação (Ferreira, et al., 2017).

O *Python* é uma linguagem que cumpre todos os requisitos anteriores. O código foca-se na legibilidade e coerência, tendo entre um terço a um quinto do tamanho do código equivalente em C++ ou Java. Para além disso, tem uma vasta coleção de funcionalidades pré-criadas conhecidas como Bibliotecas. Estas bibliotecas suportam uma variedade de tarefas de programação, desde a correspondência de padrões de texto até *network scripting* (Lutz, 2013).

O *Python* é assim uma linguagem útil para realizar tarefas do mundo real, isto é, o tipo de encargos que os programadores fazem dia após dia (Lutz, 2013).

Para além de todos os fatores acima referidos, o *Python* é uma linguagem que, desde que foi criada em 1989 e, especialmente desde finais do século vinte, tem sofrido um aumento exponencial no número de utilizadores, sendo neste momento a quarta

linguagem de programação mais usada do mundo e, em 2018, foi a linguagem mais pesquisada no Google (Figura 3).

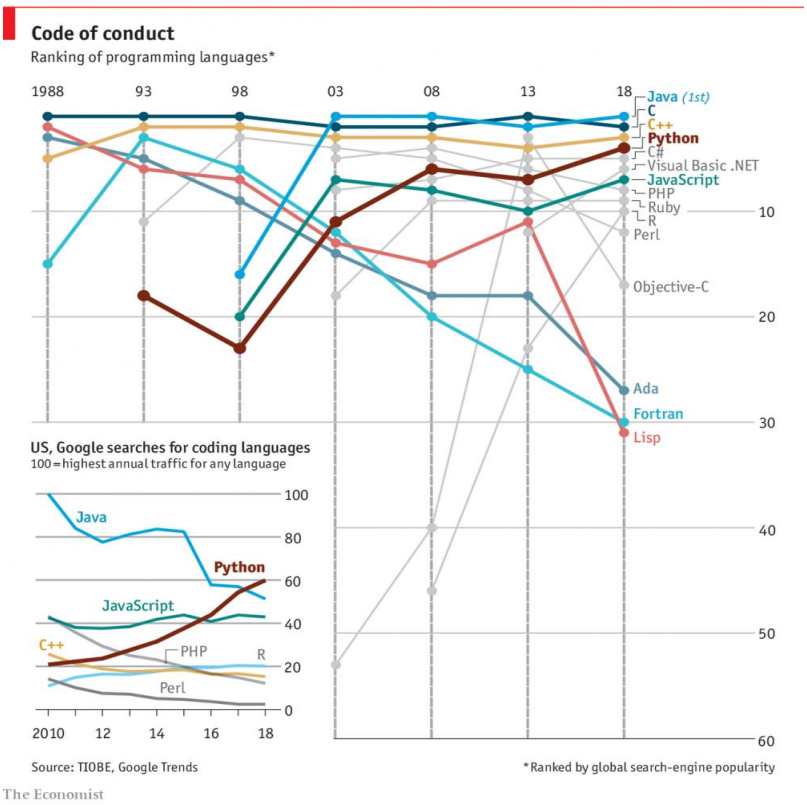

FIGURA 3 – "*Python is becoming the world's most popular coding language*", 2018.

Apresentados todos os pressupostos, será desenvolvida uma aplicação móvel para a aprendizagem de *Python*. Esta aplicação terá uma vertente teórica dividida por aulas. Cada aula corresponderá a um conjunto de temas e conceitos e o grau de dificuldade aumentará progressivamente. Existirá uma componente prática com propostas de exercícios, onde os utilizadores poderão aplicar os conhecimentos apreendidos em cada lição. Esta componente prática será constituída por uma interface dividida em parcelas: uma janela de comandos onde os utilizadores podem testar código e o *output* desse mesmo código, isto é, o *feedback* devolvido pela interface.

4. IMPLEMENTAÇÃO

Para executar a proposta conceptual apresentada decidiu-se utilizar o *app inventor* do MIT. O *app inventor* permite utilizar um *Web Viewer* que se articula com um URL. Assim, para implementar a aplicação, decidiu-se utilizar HTML.

A próxima figura mostra a interface inicial. Nesta interface o utilizador tem o primeiro contacto com a estrutura do curso de programação e pode escolher a aula que quer assistir (Figura 4).

FIGURA 4 – Interface inicial da aplicação móvel.

A interface de cada aula é composta pelos conteúdos teóricos divididos por slides e por uma janela de comandos. Nesta janela de comandos o utilizador pode testar os conceitos assimilados em cada aula, através da resolução de exercícios propostos. Na janela de comandos são apresentadas algumas dicas para a resolução desses mesmos exercícios. Após introduzir o código correspondente a cada exercício, o utilizador pode carregar num botão e a interface devolve feedback, isto é, o output do código introduzido (Figura 5)

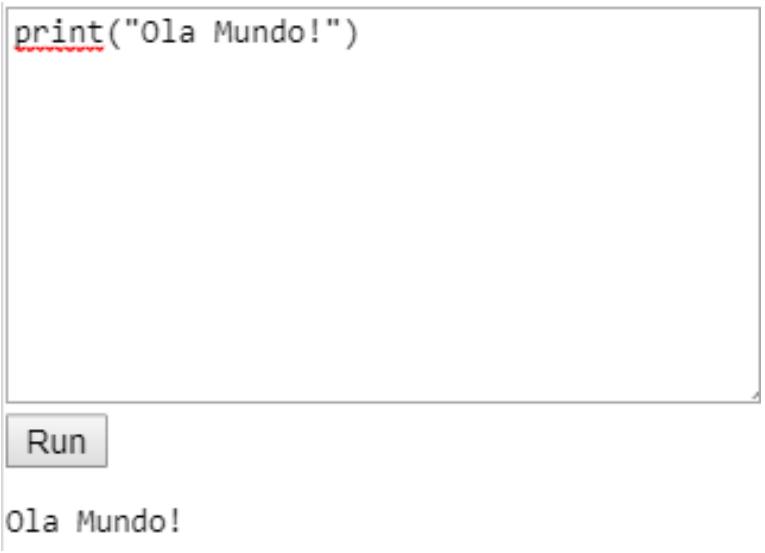

FIGURA 5 – Janela de comandos da aplicação móvel.

A componente teórica aborda conceitos e temas introdutórios na aprendizagem de *Python*. Estes conceitos estão ordenados por grau de dificuldade e por uma sequência lógica de aprendizagem, ou seja, o conteúdo teórico de uma aula implica o conhecimento dos conteúdos lecionados nas aulas anteriores. Posto isto os temas abordados são: variáveis; operadores; estruturas de controlo; listas e tuplos; dicionários; conjuntos; funções.

4.1.Variáveis

Neste segmento os utilizadores aprendem o conceito e os tipos de variáveis existentes. Na vertente prática os estudantes exploram o uso e atribuição de variáveis e têm ainda um primeiro contacto com simples operações aritméticas e com *strings* (Figura 6).

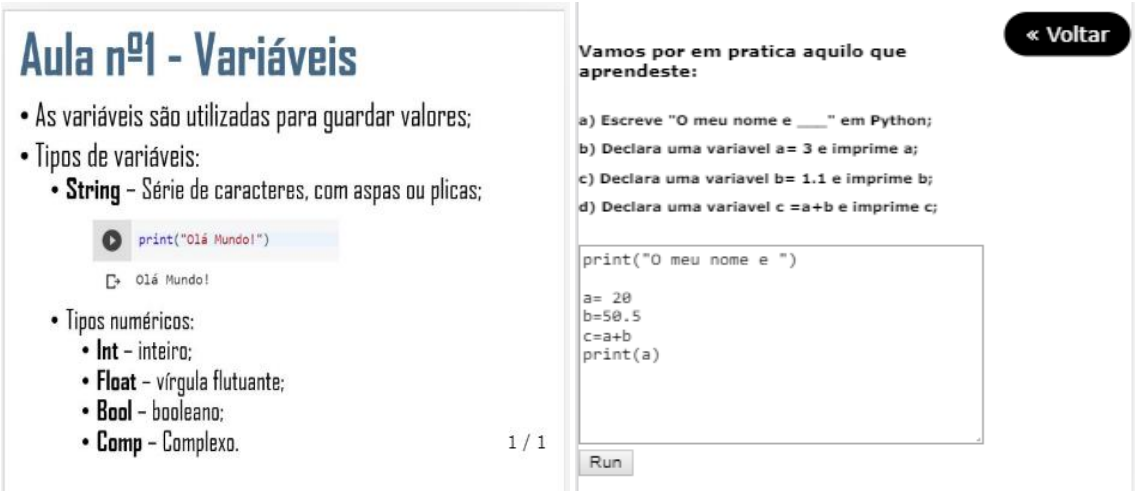

FIGURA 6 – Variáveis.

4.2. Operadores

A segunda aula serve para apresentar aos utilizadores os tipos de operadores existentes (Figura 7)

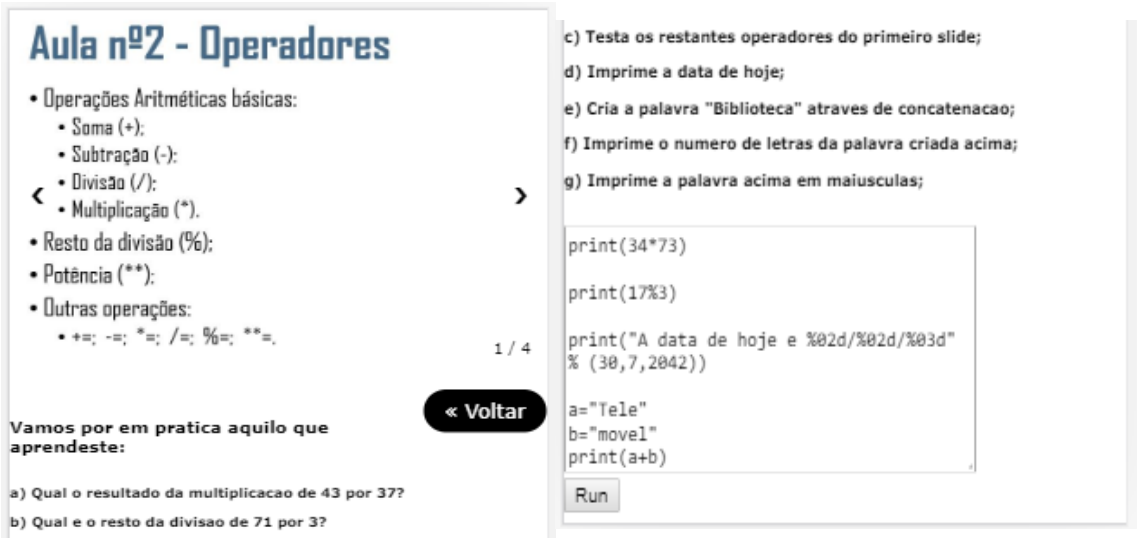

FIGURA 7 – Operadores.

Aqui podem familiarizar-se com operadores aritméticos, de comparação, lógicos e ainda operadores com *strings*. Na vertente prática disponibiliza-se um conjunto de exercícios onde os utilizadores podem testar todos os conceitos assimilados.

4.3.Estruturas de Controlo

Na terceira aula são apresentadas as estruturas de controlo. São abordadas as estruturas de condição (IF e IF-ELSE) e ainda as estruturas de repetição (WHILE). A componente prática engloba os conceitos anteriores. O utilizador pode testar as estruturas através de condições com operadores e variáveis, sendo que cada exercício é progressivamente mais difícil (Figura 8).

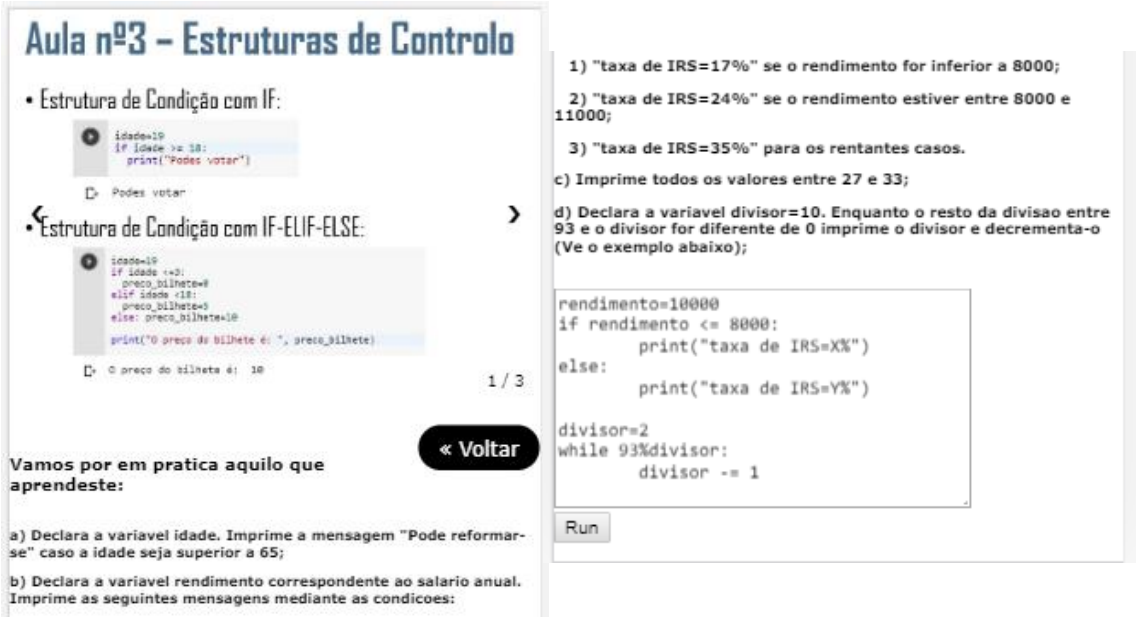

FIGURA 8 – Estruturas de Controlo.

4.4. Listas e Tuplos

No quarto segmento são apresentados os conceitos de listas e tuplos (Figura 9).

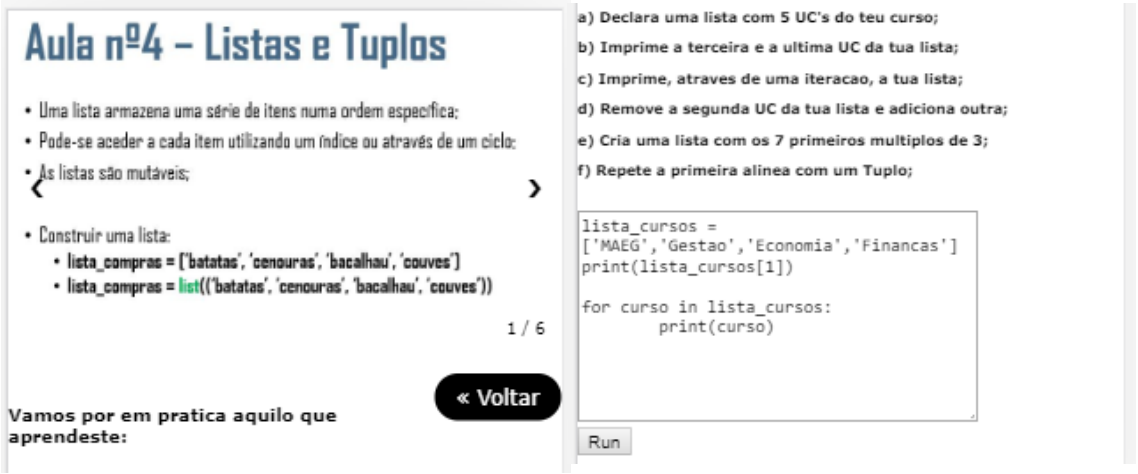

FIGURA 9 – Listas e Tuplos.

Os utilizadores aprendem alguns métodos de operar com listas e, na componente prática, são convidados a formalizar alguns desses métodos, sendo que existe uma complexidade superior nos exercícios propostos comparativamente com as aulas anteriores.

4.5. Dicionários

O conceito de dicionário é introduzido na quinta aula. Os dicionários são conceitos bastante semelhantes a listas e tuplos, no entanto, para que a duração da aula não fosse desproporcional comparativamente com as restantes estes conceitos foram divididos em dois segmentos (Figura 10).

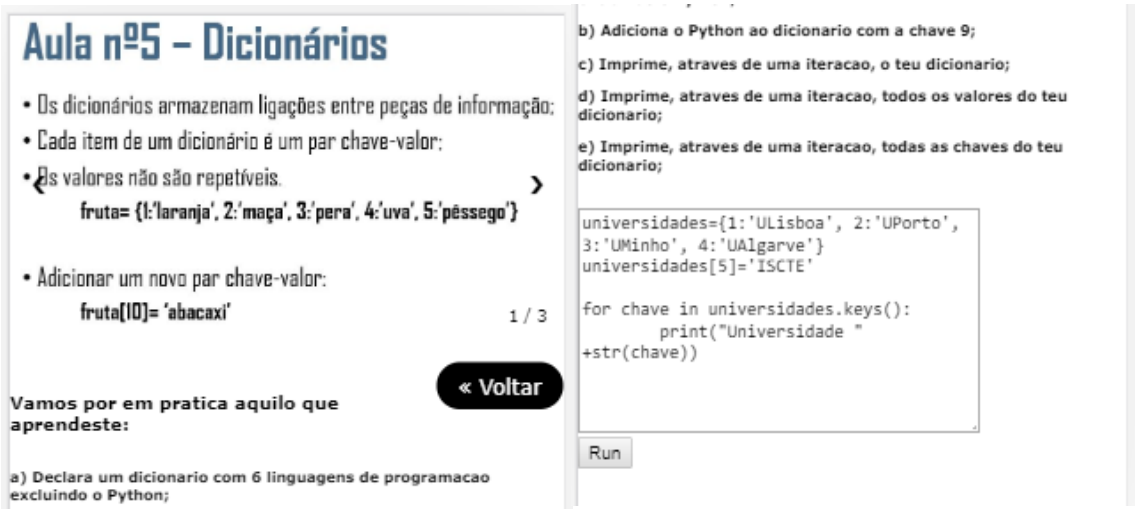

FIGURA 10 – Dicionários.

4.6. Conjuntos

A aula número seis é a respeito de conjuntos. Conjuntos são estruturas específicas do *Python*, que são utilizadas para representar coleções não ordenadas de elementos únicos. Neste segmento os utilizadores aprendem a funcionar com estes conceitos, utilizando operadores de união, interseção e diferença (Figura 11).

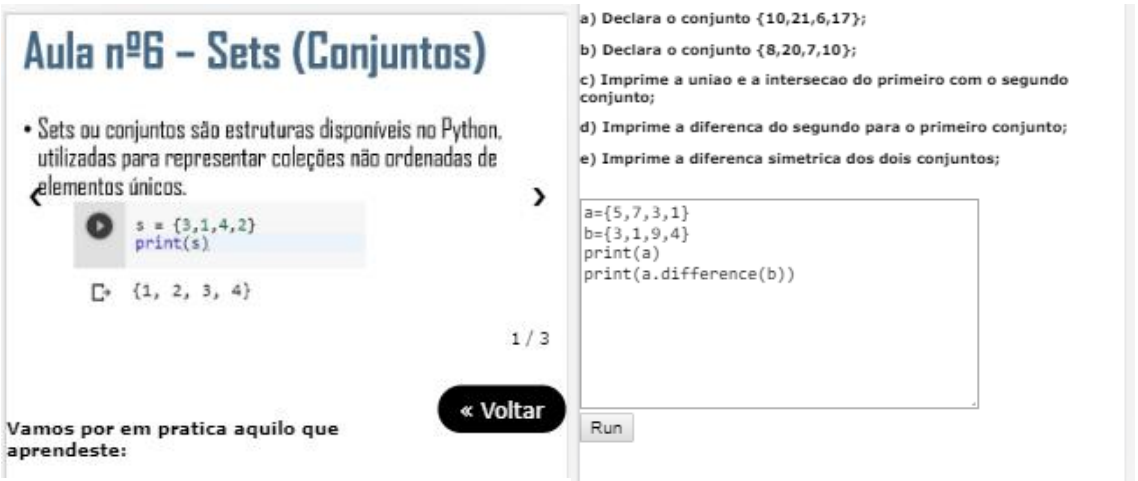

FIGURA 11 – Conjuntos.

4.7. Funções

Na última aula é desenvolvido o tema mais complexo do curso. Numa primeira fase é explicado o conceito e invocação de uma função e, mais à frente, expõem-se alguns exemplos de funções (Figura 12).

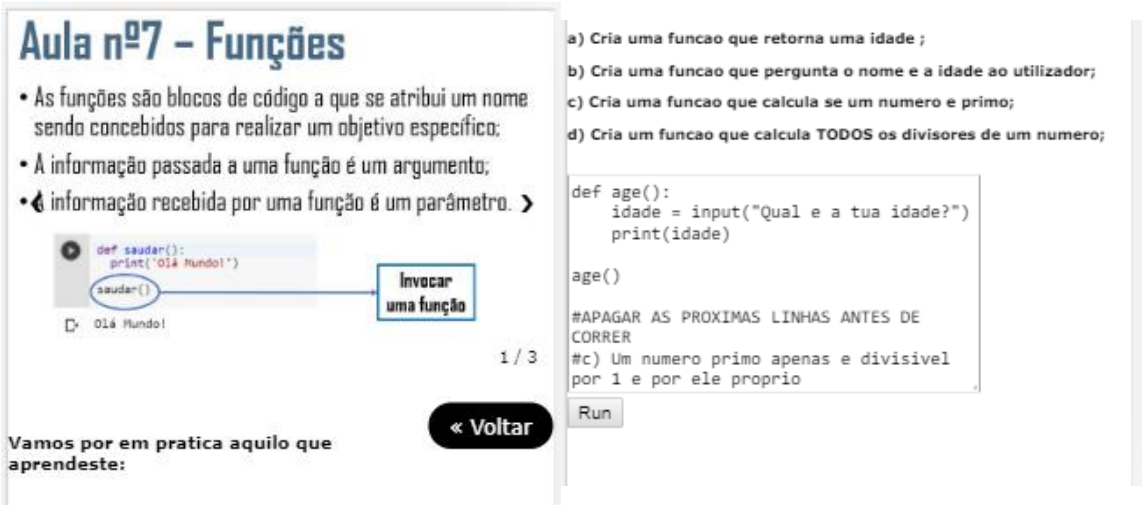

FIGURA 12 – Funções.

Este é o tema mais abrangente de todo o curso, pois permite aos utilizadores colocarem em prática todos os conceitos assimilados nas aulas anteriores.

5. AVALIAÇÃO PRELIMINAR

Após a implementação da aplicação móvel, foram selecionados três utilizadores (com conhecimentos básicos de programação, mas sem contacto anterior com Python) para testarem a mesma. A cada utilizador foi pedido que navegassem na aplicação e realizassem todos os exercícios. Logo após o teste, os utilizadores avaliaram a aplicação em três vertentes: Componente Teórica, Componente Prática e Desenho da Aplicação. Os utilizadores apontaram as vantagens, desvantagens e ainda sugestões de melhoramento para cada uma das vertentes.

5.1.Componente Teórica

Na tabela 1 foram identificadas as principais apreciações dos utilizadores.

Vantagens	- Conteúdo explícito e direto - Exemplos fáceis e demonstrativos - Aulas muito abrangentes
Desvantagens	- Um pouco simplista - Os slides poderiam estar melhor apresentados
Sugestões	- Acrescentar exemplos de casos reais da aplicabilidade do Python após cada aula ou no final do curso - Acrescentar um exemplo mais complexo no final de cada aula

TABELA 2 – Avaliação preliminar da componente teórica da aplicação.

Os utilizadores identificaram o conteúdo como sendo bastante explícito e abrangente. Os exemplos teóricos também foram apontados como um benefício, pois facilitam a aprendizagem.

As principais desvantagens mencionadas foram o facto de o conteúdo ser um pouco elementar e ainda a fraca apresentação dos slides.

Os utilizadores sugeriram ainda acrescentar exemplos mais complexos no final de cada aula e exemplos de casos reais da aplicabilidade do *Python*, de modo a que os utilizadores tenham uma melhor perceção da utilidade da linguagem de programação.

5.2.Componente Prática

A componente prática foi igualmente avaliada pelos utilizadores (Tabela 2)

Vantagens	- Forte componente prática com vasta diversidade de exercícios, que permite ao utilizador meter em prática os conceitos estudados - Exercícios de fácil compreensão que não induzem os utilizadores em erro
Desvantagens	- Nalguns exercícios estão a ser testados vários conceitos ao mesmo tempo, o que cria alguma confusão ao utilizador - Não apresenta feedback personalizado, o que seria útil para identificar os erros no código
Sugestões	- As dicas de solução poderiam aparecer numa página oculta, pois facilitam em demasia a resolução dos exercícios

TABELA 3 – Avaliação preliminar da componente prática da aplicação.

A diversidade de exercícios e a sua complexidade foram considerados como vantagens na utilização da ferramenta, uma vez que permitem aos utilizadores meter em prática os conceitos absorvidos e não os induzem em erro.

O facto de vários conceitos serem testados ao mesmo tempo foi considerado uma desvantagem, uma vez que pode confundir o utilizador. No entanto, a principal desvantagem foi a ausência de feedback personalizado, isto é, o facto da ferramenta não apontar os erros específicos do código.

Sugeriram ainda esconder as dicas de resolução dos exercícios, pois, por vezes, podem facilitar em demasia a resolução dos mesmos.

5.3. Desenho da Aplicação

O desenho da aplicação foi apontado como sendo bastante intuitivo, organizado e de fácil utilização. A interface também foi destacada como uma vantagem, uma vez que o a componente teórica ser seguida pela componente prática facilita a aprendizagem do utilizador, pois permite meter em prática instantaneamente os conceitos de cada aula.

As principais desvantagens mencionadas foram relacionadas com a componente visual. O desenho da aplicação foi indicado como pouco apelativo visualmente,

nomeadamente a falta de uniformização do tamanho das letras e a falta de acentos nas palavras. As setas para mudar de slide foram igualmente classificadas como desvantagens, uma vez que a cor das mesmas não é a mais apelativa, uma vez que se confunde com o fundo dos slides.

De forma a aperfeiçoar a aplicação, os utilizadores sugeriram mudar a cor das setas de mudança de slide e tornar o design mais apelativo visualmente. Sugeriram ainda criar uma janela inicial, onde existiria uma pequena introdução sobre a aplicação. Por fim recomendaram a criação de um *assessment* final para que os utilizadores tenham a oportunidade de aplicar todos os conhecimentos estudados nas aulas (Tabela 3).

Vantagens	- Intuitivo - Muito clean e bem organizada - User friendly - A componente teórica é seguida pela componente prática, o que facilita a aprendizagem
Desvantagens	- Pouco apelativo visualmente - Tamanho da letra não está uniformizado - As palavras não têm acentos - As setas para mudar de slide são difíceis de encontrar
Sugestões	- Tornar as setas dos slides mais visíveis - Personalização do design - Introduzir uma janela introdutória antes de seguir para as aulas - Exame final onde os utilizadores apliquem os conhecimentos gerais assimilados

TABELA 4 – Avaliação preliminar do desenho da aplicação.

5.4. Avaliação Geral

Após analisarem-se as avaliações dos utilizadores, criou-se uma análise SWOT com as conclusões essenciais da aplicação móvel (Tabela 4).

As forças e fraquezas identificadas estão em linha com as vantagens e desvantagens apresentadas pelos utilizadores.

O conteúdo explícito, os exercícios de fácil compreensão, a forte componente prática e o fácil manuseamento da aplicação foram considerados os pontos fortes da ferramenta.

As fraquezas detetadas foram a ausência de feedback personalizado, o facto de serem testados vários conceitos simultaneamente e a excessiva simplicidade da aplicação.

As oportunidades apresentadas são as introdução de exemplos reais da aplicabilidade do *Python* para que os utilizadores tenham uma real perceção onde e como poderão aplicar os conhecimentos assimilados, o desenvolvimento do design da aplicação tornando-a mais apelativa para o utilizador, criação de uma janela inicial com uma pequena introdução sobre a aplicação e os seus objetivos e ainda a inclusão de um exame final para que os utilizadores possam testar todos os conhecimentos aprendidos. Ao contrário dos exercícios de cada aula, este exame não tem acesso à componente teórica, ou seja, seria como um exame final sem consulta.

Por fim, uma das ameaças identificadas foi o possível surgimento de outras aplicações com o mesmo conceito, mas mais apelativas visualmente e melhor construídas. A outra ameaça foi o aparecimento de uma linguagem de programação que coloque o *Python* em desuso.

Forças	**Oportunidades**
- Conteúdo explícito e direto - Exemplos e exercícios de fácil compreensão - Forte componente prática - User friendly	- Introduzir exemplos reais da aplicabilidade do Python - Personalizar o design da aplicação - Criar uma janela introdutória - Incluir um exame final
Fraquezas	**Ameaças**
- Simplista - Demasiados conceitos a ser testados ao mesmo tempo - Não apresenta feedback personalizado - Pouco apelativo visualmente	- Surgimento de aplicações com o mesmo conceito mas mais apelativas - Criação/aparecimento de uma linguagem de programação que coloque o Python em desuso

TABELA 5 – Análise SWOT à aplicação móvel.

Concluindo, os utilizadores avaliaram positivamente a proposta conceptual, uma vez que as principais vantagens enumeradas estão relacionadas com a componente prática e a sua utilidade na aprendizagem de programação, enquanto que as fraquezas estão maioritariamente associadas ao design e a sua simplicidade.

6. CONCLUSÃO

Com a evolução constante do mundo digital surgiu uma necessidade crescente de pessoas qualificadas na área da programação. No entanto, o número de pessoas qualificadas não aumentou ao mesmo ritmo que essa necessidade. Tal pode ser explicado pelas dificuldades inerentes à aprendizagem e ensino da programação.

As principais dificuldades identificadas na aprendizagem de programação foram os métodos de estudo, a falta de capacidade e atitudes dos estudantes e ainda a natureza das linguagens de programação. A nível do ensino as dificuldades descritas foram o ensino de conceitos dinâmicos através de materiais estáticos e ainda a o ensino das particularidades de cada linguagem ao invés de métodos de resolução de problemas.

Para ultrapassar as dificuldades criou-se uma aplicação móvel para a aprendizagem de *Python*, onde os utilizadores têm um conjunto de aulas teóricas seguido de propostas de exercícios práticos.

A proposta conceptual foi avaliada por três utilizadores que apontaram como principais vantagens o conteúdo explícito e direto, os exemplos apresentados, a estrutura da aplicação e a forte componente prática.

Para trabalhos futuros sugiro a revisão do conteúdo teórico e prático: introduzir conteúdos mais complexos e reformular os exercícios práticos em conformidade com a revisão teórica. Recomendo também o melhoramento do design da aplicação, de modo a ser mais apelativa visualmente e mais moderna. Por fim, sugiro ainda a introdução de um exame final de avaliação incluindo componentes de gamificação. Este exame daria a oportunidade aos utilizadores de meterem em prática todos os conceitos aprendidos durante o curso e a incorporação de elementos de gamificação aumentaria os níveis de motivação e envolvimento dos utilizadores no ensino da programação.

REFERÊNCIAS BIBLIOGRÁFICAS

Aparicio, J. T., Aparicio, M., & Costa, C. J. (2018). A Virtual Robot to support Programming Learning. In *ISR 2018; 50th International Symposium on Robotics* (pp. 1-4). VDE.

Aparicio, J. T., & Costa, C. J. (2018). A virtual robot solution to support programming learning an open source approach. *2018 13th Iberian Conference on Information Systems and Technologies (CISTI)*, 1–6. https://doi.org/10.23919/CISTI.2018.8399263

Aparicio, J. T., Pereira, S., Aparicio, M., & Costa, C. J. (2019). Learning Programming Using Educational Robotics. In *2019 14th Iberian Conference on Information Systems and Technologies (CISTI)* (pp. 1-6). IEEE. https://doi.org/10.23919/CISTI.2019.8760709

Avouris, N. (2018). Introduction to computing: A survey of courses in Greek higher education institutions. *Proceedings of the 22nd Pan-Hellenic Conference on Informatics - PCI '18*, 64–69. https://doi.org/10.1145/3291533.3291549

Costa, C. J., Aparicio, M., & Cordeiro, C. (2012). A solution to support student learning of programming. *Proceedings of the Workshop on Open Source and Design of Communication - OSDOC '12*, 25–29. https://doi.org/10.1145/2316936.2316942

Costa, C. J., Silva, J., & Aparício, M. (2007). Evaluating web usability using small display devices. In *Proceedings of the 25th annual ACM international conference on Design of communication* (pp. 263-268). ACM. https://doi.org/10.1145/1297144.1297202

Deterding, S., Dixon, D., Khaled, R., & Nacke, L. (2011). *From game design elements to*

gamefulness: Defining «gamification». In Proceedings of the 15th International Academic MindTrek Conference: Envisioning Future Media Environments (MindTrek '11). ACM, New York, NY, USA, 9-15. DOI: https://doi.org/10.1145/2181037.2181040

Ferreira, F., Costa, C. J., Aparicio, M., & Aparicio, S. (2017). Aprendizagem na programação: Um modelo de continuidade de aprendizagem de programação. *12th Iberian Conference on Information Systems and Technologies, CISTI 2017* https://doi.org/10.23919/CISTI.2017.7975815

Gomes, A., & Mendes, A. J. (2007). Learning to program – difficulties and solution. *International Conference on Engineering Education – ICEE 2007.*

Lutz, M. (2013). *Learning Python: Powerful Object-Oriented Programming*. O'Reilly Media, Inc.

Papadakis, S., & Orfanakis, V. (2018). Comparing novice programing environments for use in secondary education: App Inventor for Android vs. Alice. *International Journal of Technology Enhanced Learning*, *10*(1/2), 44. https://doi.org/10.1504/IJTEL.2018.10008587

Pereira, R., Costa, C. J., & Aparicio, J. T. (2017). Gamification to support programming learning. *2017 12th Iberian Conference on Information Systems and Technologies (CISTI)*, 1–6. https://doi.org/10.23919/CISTI.2017.7975788

Piteira, M., & Costa, C. (2013). Learning computer programming: Study of difficulties in learning programming. *Proceedings of the 2013 International Conference on Information Systems and Design of Communication - ISDOC '13*, 75. https://doi.org/10.1145/2503859.2503871

Piteira, M., & Costa, C. J. (2017). Gamification: Conceptual framework to online courses of learning computer programming. *2017 12th Iberian Conference on Information Systems and Technologies (CISTI)*, 1–7. https://doi.org/10.23919/CISTI.2017.7975695

Python is becoming the world's most popular coding language. (2018, July 26). The Economist. Retrieved from https://www.economist.com/graphic-detail/2018/07/26/python-is-becoming-the-worlds-most-popular-coding-language

Vihavainen, A., Airaksinen, J., & Watson, C. (2014). A systematic review of approaches for teaching introductory programming and their influence on success. *Proceedings of the Tenth Annual Conference on International Computing Education Research - ICER '14*, 19–26. https://doi.org/10.1145/2632320.2632349