UNIVERSIDADE DE LISBOA

INSTITUTO DE EDUCAÇÃO

Ler bem, escrever melhor: o papel da leitura de código na estruturação do Pensamento Computacional

Curso Profissional de Técnico de Gestão e Programação de Sistemas Informáticos

Observação e Planificação da Intervenção no Módulo 4 de Programação e Sistemas de Informação

na ESCOLA SECUNDÁRIA DE SÃO JOÃO DA TALHA

Agrupamento de Escolas de São João da Talha, Loures

Joana Paulo Pardal, 22678

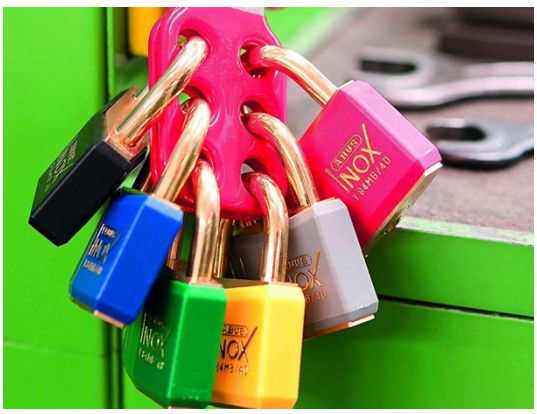
Orientadores: *Prof. Dr. João Manuel Nunes Piedade*

Prof. Dr. José Carlos Medeiros de Campos

INICIAÇÃO À PRÁTICA PROFISSIONAL III

MESTRADO EM ENSINO DA INFORMÁTICA

Janeiro de 2020



"Teaching should be such that
what is offered is perceived as a valuable gift
and not as hard duty.
Never regard study as a duty
but as the enviable opportunity to learn
to know the liberating influence of beauty
in the realm of the spirit
for your own personal joy
and to the profit of the community
to which your later work belongs."
— Albert Einstein

RESUMO

Neste relatório descreve-se a experiência feita na Escola Secundária de São João da Talha, onde se observaram três aulas da disciplina de Programação e Sistemas de Informação do curso profissional de Técnico de Gestão e Programação de Sistemas Informáticos, com vista à preparação da intervenção pedagógica a realizar no 3.º período, que ocorre durante o 2.º semestre.

Apresenta-se a planificação e a calendarização de 12 de Maio a 9 de Junho de 2020. Durante esse período lecionarei, com a supervisão do professor cooperante, a primeira metade do módulo 4 desta disciplina, relativo a "Estruturas de Dados Estáticas".

Pretende-se ainda apresentar a investigação que se fará durante a intervenção e que se refere à problemática relativa ao impacto das estratégias utilizadas no desenvolvimento do pensamento computacional dos alunos. Em particular abordar-se-ão as seguintes questões de investigação:

Q_1: Qual o impacto das atividades desligadas na compreensão conceptual dos algoritmos?

Q_2: Qual o impacto da leitura de (bom) código na implementação dos algoritmos relacionados?

Q_3: Como é que a representação pictórica do funcionamento de um programa em memória melhora a compreensão dos algoritmos?

Conclui-se este relatório com uma reflexão sobre o as opções tomadas e as elações que se retiram para a intervenção e para o trabalho docente futuro.

PALAVRAS CHAVE:

Aprendizagem baseada em Problemas;

Estruturas de Dados Estáticas;

Introdução à Programação;

Atividades Desligadas;

Pensamento Computacional;

Escape Room Criptográfico.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

1. INTRODUÇÃO

Este relatório descreve a preparação da intervenção pedagógica a realizar-se no 3.º período do ano letivo 2019/20 na ***Escola Secundária de São João da Talha***, Loures, com os alunos do 10.º ano, turma E, do curso profissional de ***Técnico de Gestão e Programação de Sistemas Informáticos***, na disciplina de ***Programação e Sistemas de Informação***, da responsabilidade do professor cooperante ***José António Cruz***.

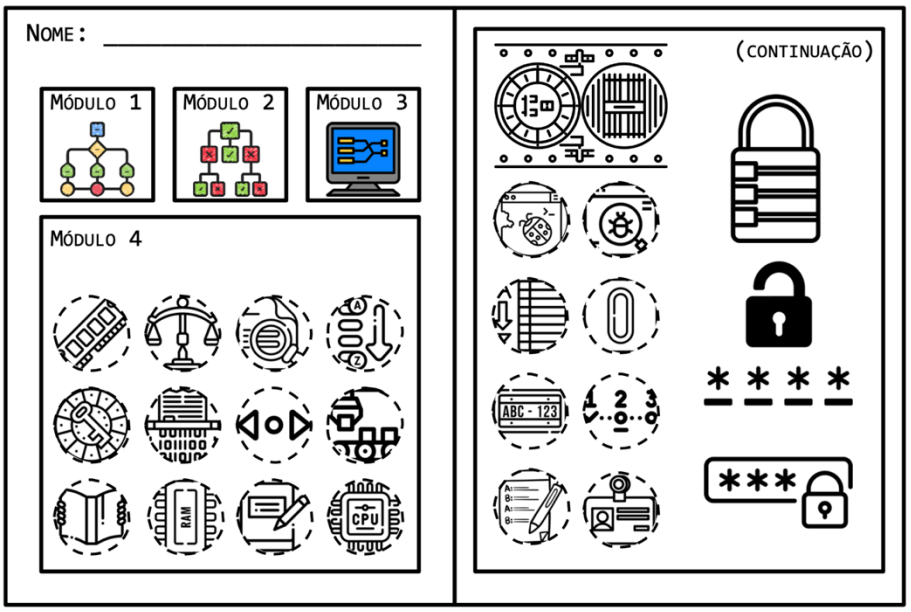
O trabalho descrito neste relatório integra-se na disciplina de ***Iniciação à Prática Profissional III***. Os materiais pedagógicos produzidos são do foro da disciplina de ***Didática da Informática III***, e a intervenção pedagógica descrita será realizada no âmbito da disciplina de ***Iniciação à Prática Profissional IV*** e a sua execução será descrita em detalhe no ***Relatório de Prática Supervisionada*** cujo plano pode ser consultado no anexo **Error! Reference source not found.**. Todas as disciplinas integram o curso de ***Mestrado em Ensino da Informática***, e são relativas ao ano letivo 2019/20.

A intervenção foi planeada para ocupar os 16 tempos letivos correspondentes à primeira metade do módulo 4 da disciplina: ***Estruturas de Dados Estáticas***. Pretende explorar várias pistas pedagógicas e didáticas relativas à aprendizagem inicial da programação, mas incluindo temas menos habituais, como a criptografia, como base das atividades. O uso de atividades desligadas, a leitura de código e a representação esquemática da memória durante a execução de um programa são utilizadas como método para a melhor compreensão dos algoritmos de pesquisa e ordenação.

Registam-se também as observações da disciplina que foram feitas durante os módulos 1 e 2, e que ocorreram em datas espaçadas de acordo com o calendário das aulas de *Didática da Informática III e IV* (que coincidiam) e com o calendário do professor cooperante que se ausentou do país várias vezes para participar em projetos internacionais e formações no estrangeiro.

Depois desta breve introdução, o relatório apresenta o contexto escolar da escola onde a intervenção vai decorrer tendo como base o projeto educativo da escola, o seu regulamento interno e os restantes documentos oficiais disponíveis no *website* da escola. Segue-se um breve enquadramento curricular da disciplina de ***Programação e Sistemas de Informação***, incluindo as relações inerentes aos temas a abordar nas várias disciplinas técnicas do curso.

Na secção 4, descreve-se a planificação da intervenção Pedagógica, e na secção 5, apresenta-se a componente de investigação que se aprofundará durante a intervenção sobre o impacto das metodologias e estratégias adotadas na aprendizagem dos alunos sob a forma de capacidades de pensamento computacional. O documento termina com um balanço reflexivo e as conclusões, na secção 6. No final apresentam-se as referências bibliográficas e os anexos que permitem compreender melhor aquilo que se propõe fazer na intervenção pedagógica.



2. CONTEXTO ESCOLAR

A escola localiza-se no município de Loures, na região de Lisboa e Vale do Tejo, que está integrado na Grande Área Metropolitana de Lisboa e tem como concelhos limítrofes: Lisboa, Vila Franca de Xira, Arruda dos Vinhos, Mafra, Sintra e Odivelas.

Figura 2.1: Localização do concelho de Loures na Área Metropolitana de Lisboa.

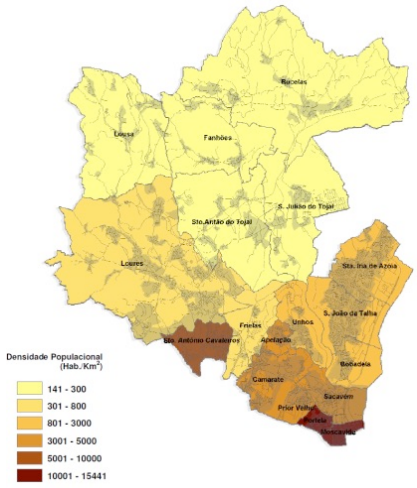


Figura 2.2: Densidade populacional por freguesia no concelho de Loures em 2001.

De acordo com a "Carta Educativa do Município de Loures" (2006), em 2004/05, 7% dos alunos do 2.º e 3.º ciclos têm necessidades específicas de educação. Destes, 48% tem dificuldades de aprendizagem e 16% tem distúrbios emocionais. A taxa média de retenção no concelho no 2.º e 3.º ciclos é de 16% e a taxa média de abandono escolar é de 2%.

2.1. CONTEXTO SOCIAL

O Concelho de Loures conta com uma vasta rede escolar com 20 escolas.

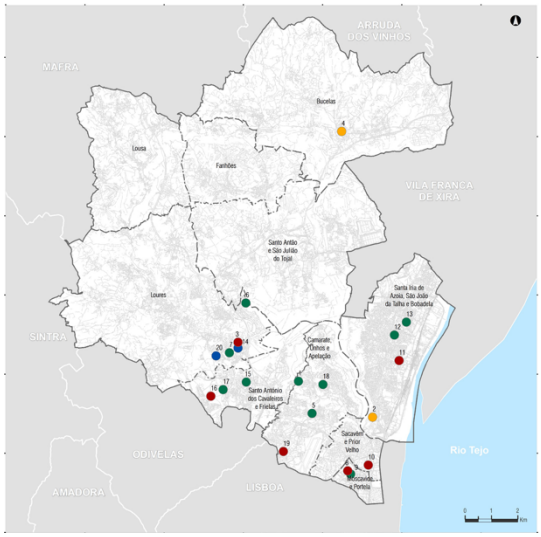


Figura 2.3: Escolas da rede escolar do concelho: a Secundária de São João da Talha é a #11.

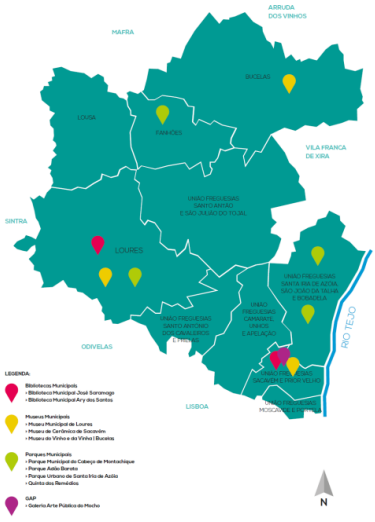


Figura 2.4: Bibliotecas Museus e Parques Municipais de Loures.

2.1.1. AGRUPAMENTO DE ESCOLAS DE SÃO JOÃO DA TALHA

O Agrupamento de Escolas de São João da Talha situa-se em São João da Talha, concelho de Loures e oferece todos os ciclos escolares. A Escola Secundária de São João da Talha faz parte do Agrupamento de Escolas de São João da Talha que integra, além desta, a Escola Básica de S. João da Talha, a Escola EB1/JI n.º 1 e a n.º 2 de São João da Talha, a Escola EB1/JI de Vale de Figueira, e a Escola EB1 n.º 4 de São João da Talha.

2.1.2. ESCOLA SECUNDÁRIA DE SÃO JOÃO DA TALHA

A Escola Secundária recebe alunos do 3.º ciclo e do ensino secundário. Tem como missão "*Formar cidadãos com uma sólida formação pessoal, social e científica, que desenvolvam as competências necessárias para um bom desempenho profissional e pessoal, com autonomia e espírito crítico, visando a sua integração numa sociedade em constante mudança.*"

A escola oferece o 3.º ciclo, Ensino Secundário Científico-Humanístico e Profissional. É uma escola dinâmica com participação em vários projetos como GEN10S da Google com 7 turmas do 5.º e 6.º anos da Escola Básica; ERASMUS+, *Teaching and Learning Through Motion Pictures – Me and EU*; *Beyond Learning with robots*, Blend*it*, EML@b, e UAC.

Tem uma "Sala do Futuro" com alguns dos *kits* de *drones*, robôs e *kits* de eletrónica incluindo *Arduino*, equipamentos interativos (quadro interativo), equipamento de videoconferência, filmagens e produção vídeo. Além disso, depois de um investimento de dois anos na promoção da Programação e Robótica, a escola tem a disciplina de ***Introdução à Programação e Robótica*** para todos os alunos dos 5. ° e 7. ° anos como oferta curricular de escola. No entanto, os alunos que já estão no Ensino Secundário não tiveram acesso a esta oferta quando frequentaram aqueles anos letivos. Há Clube de Robótica e Clube de Teatro.

A escola tem 38 salas de aulas, 14 salas laboratoriais, 2 balneários e conta com serviços de PBX, Reprografia, Papelaria, Bar e Refeitório.

No triénio de 2016-18 tinha 410 alunos distribuídos por 27 turmas, dos quais, 274 alunos no ensino secundário e 136 alunos no ensino básico (3.º ciclo).

Em 2018/19, de acordo com os *rankings* publicados anualmente, os alunos desta escola obtiveram uma média de 9.13 nos exames nacionais e uma média de 12.17 na nota final das disciplinas. Obteve 22,6% da média nacional; e 23,0% de percentagem de percursos diretos de sucesso na escola no ensino secundário; e 27% da média nacional; e 26% de percentagem de percursos diretos de sucesso na escola no ensino básico.

A escola atribui Bolsas de Mérito a alunos do Ensino Básico e Secundário que obtenham classificação média anual igual ou superior a 4 (EB) e 14 (ES).

Tem também Quadro de Honra e Excelência com categorias de Aproveitamento e Representatividade onde figuram alunos das várias escolas do agrupamento, dos diversos ciclos.

A nível secundário, oferece vários cursos profissionais há vários anos, incluindo, na área de Informática, o de Técnico de Gestão e Programação de Sistemas Informáticos (§ Anexos D, E e F mostram as ofertas curriculares dos últimos anos letivos).

Pretende ter um computador por aluno nos laboratórios de informática, aulas de apoio a todas as disciplinas, rede e acesso à *Internet* estáveis e acesso privativo em fibra ótica. É utilizada a plataforma de Gestão de Aprendizagens ***Moodle*** que dinamiza a partilha de recursos *online* e permite a entrega de trabalhos à distância.

A escola tem uma boa rede de contactos com empresas locais onde coloca os alunos a estagiar. As empresas informam as escolas das suas necessidades em termos de linguagens de programação e outros requisitos que ajudam a futura absorção pelo mercado de trabalho local.

A escola tem protocolos de cooperação com uma série de entidades para melhorar a vida da comunidade escolar, incluindo a Junta de Freguesia, os Agrupamentos de Escolas de Santa Iria de Azóia, e Bobadela, o Técnico Lisboa e a SCIENCE4YOU.

Figura 2.5: Vista aérea da Escola Secundária de São João da Talha.

Figura 2.6: Entrada do Pavilhão F, onde se localiza a sala F-48 onde decorrerá a intervenção.

A escola tem 8 pavilhões (A a F e o de Educação Física, EF). As aulas da intervenção decorreram na sala F-48 que se localiza no pavilhão F, onde estão localizadas as várias salas de Informática. No pavilhão A encontram-se os serviços de secretaria, a reprografia, a sala de professores e a direção da escola. No pavilhão C encontra-se a papelaria, o bar da escola dentro da sala de convívio e o refeitório. Nos pavilhões B e D há salas de aula. No pavilhão E encontram-se as salas de aulas dedicadas a disciplinas de Eletricidade. No pavilhão F localizam-se os laboratórios de Informática e os de Biologia e Geologia. No pavilhão de Educação Física são os balneários e o ginásio coberto, ao lado dos campos desportivos e da pista de atletismo.

2.1.3. A TURMA 10.º E

A turma tem 24 alunos de dois cursos profissionais: Técnico de Auxiliar de Saúde (10) e Técnico de Gestão e Programação de Sistemas de Informáticos (14). A constituição da turma oscilou bastante no início do 1.º período, sobretudo por indefinição dos alunos que mudaram de escola ou de curso (alguns de Informática para Saúde).

A turma era, inicialmente, composta por 26 alunos, dos quais 14 frequentavam o Curso de Informática e 12 o de Saúde. Daqueles, 2 mudaram para Saúde, e 2 parecem ter desistido uma vez que não têm avaliação em nenhum módulo inicial de nenhuma das disciplinas. As idades dos 10 alunos variam entre 14 e 18 anos, sendo a média de 17 anos (um aluno tem 14, a idade expectável neste ano de escolaridade, dois alunos têm 16, cinco têm 17, e dois têm 18). O que significa que, tirando o mais novo, todos têm, pelo menos, uma retenção no seu percurso académico. Apenas um aluno foi retido no Ensino Secundário.

No ano letivo anterior, os alunos não estavam todos na mesma turma, nem sequer nesta escola: no 9.º ano frequentavam diversas turmas e alguns vieram transferidos de outras escolas.

Estas informações foram partilhadas pelo professor cooperante. Uma vez que (ainda) não se realizou nenhum teste diagnóstico, nem se aplicou nenhum inquérito sobre os alunos, estes dados (ainda) não foram aprofundados. Antes da intervenção, no início do módulo 3, será aplicado um inquérito que possa sistematizar esta informação, juntamente com os gostos dos alunos e outras informações que ajudem a escolher temas para as atividades a desenvolver.

2.1.3.1. HORÁRIO DA TURMA

A turma tem aulas diariamente das 8:15 às 18:10, exceto às sextas-feiras que tem a manhã livre, começando às 13:20; e às quartas-feiras terminam mais cedo, às 16:05. De segunda a quinta o almoço é das 13:10 às 14:15. À sexta terá que ocorrer antes das 13:20.

Esta carga horária com apenas uma manhã livre pretende habituar os alunos ao horário completo (tipicamente de 35 a 40 horas semanais) que terão quanto entrarem no mundo laboral. No entanto, tantas horas na escola dificultam o trabalho individual de estudo e consolidação, nomeadamente, a possibilidade de trabalhos de casa, dado o cansaço com que os alunos terminam o dia. Podem ser indicadas tarefas para o fim-de-semana, mas não tem sido costume entre os professores da turma.

As aulas decorrem em várias salas de aula. Nas disciplinas técnicas a turma divide-se de acordo com os cursos: metade da turma tem as disciplinas de saúde (omitidas por simplicidade), a outra metade tem as disciplinas de informática.

As aulas das disciplinas técnicas de Informática decorrem no pavilhão de Informática, F. As aulas de Programação e Sistemas de Informação, em particular, decorrem na sala F-48 às terças à tarde e às quintas de manhã. Isto configura um conflito com o meu horário profissional pelo que, no Conselho de Turma, foi aprovado trocar as aulas de quinta de manhã para sexta (que está livre) pontualmente na medida das necessidades.

Tempos	Segunda	Sala	Terça	Sala	Quarta	Sala	Quinta	Sala	Sexta	Sala
08:15 09:05	Área de Integração	B 16	Redes de Computadores	F 43	Inglês	D 37	Redes de Computadores	F 43		
09:15 10:05							**Programação 1 (Introdução)**	F 48		
10:20 11:10	Programação 2 (SI+BD)	F 43	Inglês	B 25	Português	G 56				
11:20 12:10			Português	D 30	Matemática	G 58	Português	D 30		
12:20 13:10										
13:20 14:10									Física e Química	G 55
14:15 15:05	Matemática	D 33	Sistemas Operativos	F 44	Arquitetura de Computadores	F 45	Programação 2 (SI+BD)	F 43	Arquitetura de Computadores	F 45
15:15 16:05									Sistemas Operativos	F 45
16:15 17:05	Física e Química	B 23	**Programação 1 (Introdução)**	F 48			Educação Física	E EF 1	Tecnologias de Informação e Comunicação	F 47
17:20 18:10										

Figura 2.7: Horário da Turma 10.º E da Escola Secundária de São da Talha no ano letivo de 2019/20.

Os alunos têm 11 disciplinas diferentes, o que os dispersa bastante. A estas disciplinas, previstas no próprio plano de curso, acresce a existência simultânea de dois módulos de programação que estão divididos por dois professores: Programação 1 e 2. A primeira dedica-se aos módulos introdutórios (1 a 4), a segunda dedica-se aos módulos de base de dados (12 a 15).

2.1.3.2. RESULTADOS DOS PRIMEIROS MÓDULOS

Nos primeiros módulos, terminados no final do 1.º período, os alunos tiveram uma média de 14 valores na média das disciplinas. Tendo tido uma média de 13,7 a Português no módulo de "*Textos de carácter autobiográfico*"; 13,6 a Matemática no módulo de "*Geometria*"; 14,7 a Inglês no módulo "*Eu e o Mundo Profissional*".; 11,8 a Física e Química, no módulo "*Estrutura Atómica, Tabela Periódica. Ligações Químicas.*"

Nas disciplinas de Área de Integração, Educação Física, Tecnologias de Informação e Comunicação (TIC), e Redes de Comunicação ainda não foram concluídos nenhuns módulos por ainda não ter havido tempo letivo suficiente pelo que, ainda não foram atribuídas classificações.

Nas disciplinas técnicas de Arquitetura de Computadores no módulo "*Sistemas Digitais*" a média da turma foi de 16,3; no de "*Sistema Operativo Cliente*", de Sistemas Operativos, foi 15,8.

Na disciplina de Programação e Sistemas de Informação (PSI), os alunos completaram dois módulos iniciais de dois dos temas principiais da disciplina: "*Introdução à Programação e Algoritmia*" com média de 14,4 e "*Introdução aos Sistemas de Informação*" com 12,8.

Apesar destes módulos serem introdutórios, os resultados foram bastante melhores no de Algoritmia. Tal prende-se, provavelmente, com a maior necessidade de maturidade para compreender estes temas mais abrangentes e mais relacionados com o mundo empresarial do qual têm pouca experiência.

Note-se, também, que as médias são feitas sobre notas que, no mínimo, são de 10, uma vez que a falta de aprovação num módulo não é possível e este fica com a indicação de "não avaliado" e não com a nota, de facto, obtida na avaliação, enquanto o aluno não repete a avaliação e não obtém aprovação. Houve dois alunos sem avaliação e sem nenhum módulo terminado. Sendo os dois maiores de idade, este cenário pode configurar abandono escolar. Houve um aluno com três módulos (ainda) sem avaliação, incluindo PSI M12. Houve três alunos com um módulo sem avaliação, sendo que um é repetente e também maior de idade.

Nesta disciplina onde decorrerá a intervenção, a média das provas escritas foi baixa (6,1) com uma nota máxima de 11 valores e mínima de 1; e três alunos sem nota.

Já nas 20 fichas de trabalho, resultado da persistência, empenho diário e atenção em sala de aula, obtiveram notas bastante melhores: uma média de 14,1 com cinco alunos a obter a classificação máxima, um 19, um 15, um 7 (do aluno repetente) e quatro alunos sem nota.

Também nos 3 mini projetos os resultados foram bons: uma média 17,3 com sete alunos a atingir a classificação máxima, dois 13, um 7 (de um dos alunos que não entregou nenhuma ficha de trabalho) e dois alunos sem nota.

Estes resultados, juntamente com as componentes de "*Literacia Tecnológica*" (média: 15,2) e "*Valores Sociais e de Cidadania*" (média: 18,7) evidenciam o empenho dos alunos e a sua capacidade de trabalho quando guiados, apesar das dificuldades em articular e demonstrar o conhecimento em ambientes mais formais como provas escritas.

2.2. AULAS DE PROGRAMAÇÃO E SISTEMAS DE INFORMAÇÃO

Durante as aulas observadas, os alunos trabalharam num ambiente descontraído, enquanto ouviam música nos seus telemóveis. Em geral cumpriram o proposto apesar de nem todos os alunos mostrarem gosto na realização das tarefas. Se ouvir música enquanto trabalham pode ser motivador, na prática também serve de distração uma vez que é preciso escolher a "próxima" música e, já que estão no *YouTube*, acabam por ouvir relatos desportivos (repetidos) ou programas de *YouTubers* o que, tendo conteúdos falados, acaba por distrai-los. Além disso, a participação simultânea em tarefas diferentes (*multitasking*) tem conhecido impacto negativo no desempenho global das tarefas (Watson, Terry, & Doolittle, 2012).

As aulas decorrem em ambiente descontraído, mas controlado. Demoram um pouco a voltar dos intervalos e entram sempre um pouco agitados, mas não há problemas de maior com o seu comportamento. A exceção é o aluno mais novo, que se mostra desatento e desafiador. Este aluno, apesar de ter as competências necessárias, nem sempre se encontra na disposição de trabalhar o que atrasa o avanço da turma. Esta sua atitude reflete-se nas classificações baixas que obteve nos primeiros módulos de todas as disciplinas (média de 11 quando a média da turma foi de 14).

Outro aluno é particularmente inseguro e recusa-se a entregar os elementos de avaliação formais (testes escritos, por exemplo) quando teme que a classificação não seja tão alta quanto o desejado. No final do período tinha três disciplinas sem classificação atribuída, incluindo a de PSI, por não ter carregado os exercícios do teste para a plataforma de avaliação.

Outro aluno tem um irmão a estudar informática na universidade pelo que tem, além de muito gosto e interesse, algum apoio em casa quando tem dificuldades.

Como vimos, no primeiro módulo da disciplina de PSI, a maior parte dos alunos obteve classificação positiva. Os alunos que não tiveram foram: o aluno mais inseguro que não entregou o teste (mas que, entretanto, já teve aprovação); e os alunos que têm faltado às aulas e que também não tiveram avaliação nas restantes disciplinas e que podem corresponder a abandonado escolar.

2.2.1. PROFESSOR COOPERANTE

O professor cooperante, José António Cruz, é licenciado em Matemática Aplicada pelo Instituto Superior Técnico, Universidade Técnica de Lisboa; Mestre de Ensino de Informática pelo Instituto de Educação da Universidade de Lisboa. Participa em vários projetos da escola, incluindo o Clube de Robótica e ERASMUS+. Participa também em *workshops* internacionais como *ESA e-Technology Lab* e e*Twinning*.

Figura 2.8: Professor José António Cruz, professor cooperante.

É professor desde 1994 pelo que tem bastante experiência de ensino.

2.2.2. SALA DE AULA

A sala de aula F-48 está bem equipada. Tem 30 postos de trabalho individuais em pares o que permite o trabalho individual de cada aluno sem impedir a colaboração entre colegas. De facto, muitas são as trocas de ideias e as ajudas dadas entre pares mas não chega a constituir *Pair Programming* uma vez os alunos não trabalho, de facto, em equipa, dividindo tarefas (um que escreve e outro que analisa o que é escrito, criticamente, procurando erros ou sugerindo métodos alternativos para resolver os problemas encontrados.

As máquinas estão todas funcionais e têm ligação à *Internet* estável, tanto que os alunos ouvem música *online* enquanto trabalham.

Ao lado de cada mesa há uma mesa livre com outros dois lugares onde os alunos se podem sentar para trabalhar fora do computador, individualmente, a pares ou em grupos.

Figura 2.9: Sala F-48 da Escola Secundária de São João da Talha.

A sala é muito comprida o que dificulta a existência de aulas de caráter expositivo, mas poderia, facilmente, ser organizada de forma a criar espaços mais demarcados, por exemplo, colocando os postos de trabalho na parte de trás da sala, e as mesas nas filas da frente, onde se podem apresentar as partes mais teóricas sem que a turma esteja tão dispersa.

Tal como está, a configuração mostra a preocupação da escola em ter um computador por aluno e espaço para trabalhar no papel ou em projetos de grupo. Aliás, na planta original da escola, o espaço desta sala correspondia a duas salas mais pequenas nas quais não haveria espaço para colocar tantos computadores para o trabalho individual.

Nesta disciplina, esta disposição acaba por não ter muito impacto porque os alunos, sendo poucos, cabem todos na metade posterior da sala. Pela observação de aulas feita, a questão a ter em atenção é, no momento do ingresso na sala, garantir que os alunos se sentam na zona esperada, distribuindo-se de forma adequada. Isto porque alguns alunos tentam ir para a parte anterior da sala para "estarem à vontade a trabalhar" o que dificultaria a comunicação durante a aula.

2.2.3. RECURSOS DIDÁTICOS DISPONÍVEIS NA ESCOLA

- videoprojector e tela;
- quadro interativo (*smart board*);
- impressora a cores;
- quadro branco;
- quadro de giz;
- 30 PCs multimédia, acesso à Internet, rato e teclado;
- 30 monitores com colunas de som;
- 6 robôs mBot;
- 10 microprocessadores micro:bit;
- Arduinos.

Além disso, os alunos dispõem de telemóveis que trazem consigo e que utilizam para ouvir música ou ver vídeos na Internet (YouTube), o que poderia ser utilizado como recurso didático promovendo o visionamento de vídeos relativos aos temas abordados, se não em casa, como forma de preparar a aula seguinte, na sala, dedicando mais atenção aos conteúdos.

2.2.4. TECNOLOGIAS E APLICAÇÕES

Os computadores têm o sistema operativo *Windows 7* instalado e o *Office* da *Microsoft*.

Têm acesso à Internet pelo que poderão utilizar ambientes de programação *online*.

Os alunos utilizam o Moodle para entregar as fichas de trabalho e mini projetos ao professor.

A área de rede também é utilizada com alguma frequência.

Na disciplina em que decorre a intervenção, utiliza-se o Portugol e o Visual Studio 2012 Express, este último por requerer menos recursos informáticos do que a versão mais recente da aplicação.

Se necessário, é possível instalar software nas máquinas.

3. ENQUADRAMENTO CURRICULAR

A intervenção será realizada no âmbito da disciplina de **PSI** (***Programação e Sistemas de Informação)***, mais concretamente no módulo 4 do curso, que se refere a Estruturas de Dados Estáticas, cadeias de caracteres (*Strings*), vetores (*arrays* unidimensionais) e matrizes (*arrays* multidimensionais).

3.1. CURSO PROFISSIONAL DE TÉCNICO DE GESTÃO E PROGRAMAÇÃO DE SISTEMAS DE INFORMAÇÃO (TGPSI)

O curso profissional de Técnico de Gestão e Programação de Sistemas de Informação é criado pela Portaria n.º 916/2005 de 26 de Setembro seguindo os princípios orientadores estabelecidos no Decreto-Lei n.º 74/2004 de 26 de Março. Nela se estabelece este curso se enquadra na "família profissional de informática" e se integra na "área de educação e formação de Ciências Informáticas (418)".

Determina também que o "Perfil de desempenho à saída do curso" é tal que "O técnico de gestão e programação de sistemas informáticos é o profissional qualificado apto a realizar, de forma autónoma ou integrado numa equipa, actividades de concepção, especificação, projecto, implementação, avaliação, suporte e manutenção de sistemas informáticos e de tecnologias de processamento e transmissão de dados e informações." E identifica as principais atividades a serem desempenhadas por estes técnicos onde se incluem: "Efectuar a análise de sistemas de informação"; "Conceber algoritmos através da divisão dos problemas em componentes"; e "Desenvolver, distribuir, instalar e efectuar a manutenção de aplicações informáticas, utilizando ambientes e linguagens de programação procedimentais e visuais".

Os alunos aprendem por isso, não só a gerir sistemas de informação, com instalação e manutenção de sistemas operativos, mas também a programá-los, com a aprendizagem de vários paradigmas: imperativo, com objetos, bases de dados e páginas *web*.

A portaria de 2005 estabelece ainda o plano de estudos do curso, indicando número de horas que se deve dedicar a cada disciplina. Esta informação é complementada pelos programas das disciplinas que elencam os módulos a lecionar, os seus objetivos de aprendizagem, a duração de referência, os conteúdos e o seu âmbito, a bibliografia complementar e os recursos a mobilizar.

As várias disciplinas são oferecidas em módulos que têm aprovação individual e podem, se necessário, ser repetidos. Esta divisão dificulta a avaliação global do aluno, uma vez que cada módulo didático é pequeno e estrito. Mais, não havendo precedências, é possível ter aprovação num módulo sem ter tido aprovação nos módulos anteriores.

O Decreto-Lei 55/2018 de 6 de Julho determina que a Formação Técnica destes cursos seja reduzida de 1600 horas para 1000 a 1300; e que a Formação em Contexto de Trabalho passe de 420 horas para 600 a 840. O total de hora do curso pode ser alargado de 3100 até às 3440 horas.

O curso permite o prosseguimento de estudos no ensino superior, com a realização do exame necessário ao ingresso, tal como indicado pela universidade onde se queira ingressar.

Componentes de formação		**Total de horas** (a) (ciclo de formação)		
Sócio-Cultural:	Cidadania e Desenvolvimento (i)			
Português (b)		320		
Língua Estrangeira I ou II (c)		220		
Área de Integração		220		
Tecnologias de Informação e Comunicação (d)		100		
Educação Física		140		
Subtotal		*1 000*		
Científica:				
Matemática		300		
Física e Química		200		
Subtotal		*500*	*(máx)*	
Técnica / Tecnológica (j):				*(variável)*
Sistemas Operativos		144		140
Arquitectura de Computadores		152		140
Redes de Comunicação		252		210
Programação e Sistemas de Informação		632		610
Subtotal		*1600*	(f)	1000 a 1300
Formação em Contexto de Trabalho		420	(e)	600 a 840
Educação Moral e Religiosa (g)			*(g)*	
Total de horas do curso		***3100***	***(h)***	***a 3440***

(a) Carga horária global não compartimentada pelos três anos do ciclo de formação, a gerir pela escola, de acordo com o estabelecido na Portaria n. 550-C/2004, de 21 de Maio, e demais regulamentação aplicável. No âmbito da sua autonomia pedagógica, acautelando o equilíbrio da carga anual de forma a otimizar a gestão modular, a formação em contexto de trabalho e o seu projeto de flexibilidade com o Decreto-Lei 55/2018.

(b) Disciplina sujeita a avaliação sumativa externa, nos termos previstos no artigo 11.º do Decreto-Lei n.º 74/2004 de 26 de Março, conjugado com os artigos 26.º, 27.º e 30.º a 33.º da Portaria n.º 550-C/2004, de 21 de Maio.

(c) O aluno deverá dar continuidade a uma das línguas estrangeiras estudadas no ensino básico. Com o Decreto-Lei 55/2018, o aluno escolhe uma língua estrangeira. Se tiver estudado apenas uma língua estrangeira no ensino básico, iniciará obrigatoriamente uma segunda língua no ensino secundário

(d) Com o Decreto-Lei 55/2018, a escola opta pelo desenvolvimento da disciplina de Tecnologias de Informação e Comunicação ou por uma Oferta de Escola, de frequência obrigatória, gerindo a carga horária em função da necessidade de reforço das aprendizagens.

(e) Aumenta para 600 a 840 horas com o Decreto-Lei 55/2018.

(f) Reduz-se para 1000 a 1300 horas com o Decreto-Lei 55/2018.

(g) Com o Decreto-Lei 55/2018, é disciplina de oferta obrigatória e de frequência facultativa, com uma carga horária anual nunca inferior a 54 horas nos três anos do ciclo de formação.

(h) Com o Decreto-Lei 55/2018, a carga horária total da formação varia entre um mínimo de 3100 horas e um máximo de 3440 horas. De modo a não ultrapassar a carga horária máxima do total da formação, deve ajustar -se a carga horária da formação em contexto de trabalho em função da carga horária das UFCD da componente tecnológica.

(i) Componente desenvolvida com o contributo de disciplinas e componentes de formação com o Decreto-Lei 55/2018.

(j) Unidades de formação de curta duração desenvolvidas de acordo com os respetivos referenciais de formação constantes do CNQ, observando as orientações da Agência Nacional para a Qualificação e o Ensino Profissional, I. P., designadamente nos cursos enquadrados em regime provisório no CNQ, para os quais se mantêm as três a quatro disciplinas definidas nos planos de estudo publicados nas portarias de criação de cada curso, devendo ser aplicados os respetivos programas em vigor, com o Decreto-Lei 55/2018

Tabela 3.1: Matriz curricular do ciclo de formação do curso profissional de Técnico de Gestão e Programação de Sistemas de Informação com a indicação das alterações introduzidas pelo Decreto-Lei 55/2018.

A avaliação passa a incluir aspetos mais gerais, como a participação e o interesse, pela avaliação de atitudes e valores. Passa a existir "Cidadania e Desenvolvimento".

Na prática, a planificação das aulas é feita assumindo que os módulos são lecionados numa sequência didática pré-definida, sem nenhum registo específico de atividades de recuperação. De certa maneira, espera-se que a aprendizagem desenvolvida nos módulos seguintes permita recuperar as lacunas que impediram o sucesso no(s) módulo(s) anterior(es). Esta prática, baseada na diferenciação pedagógica, acaba por funcionar, mas um aluno que beneficia-se da repetição efetiva de módulos não tem acesso a essa modalidade que exigiria mais recursos humanos e físicos.

3.2. DISCIPLINAS SOCIOCULTURAIS

O currículo das disciplinas socioculturais é, como se disse, modular. ***Português***, ***Língua Estrangeira*** (***Inglês***), ***Área de Integração***, ***Tecnologias de Informação e Comunicação*** (TIC) e ***Educação Física*** orientam-se para o desenvolvimento de competências gerais: saber, saber fazer, saber ser, e saber aprender. Os vários programas das disciplinas têm um caráter fortemente marcado pela perspetiva da vida profissional tendo, inclusivamente, módulos especificamente dedicados à temática: Inglês, por exemplo, tem os módulos "*Eu e o Mundo Profissional*" (M1); e em Área de Integração abordam "O Mundo do Trabalho" (Área 2, Unidade Temática 6).

Dado o número total de horas previstas para o ciclo de formação, algumas disciplinas, nas diferentes implementações nas diferentes escolas, têm Inglês e Área de Integração e Física e Química apenas no 10.º e 11.º; ou TIC apenas no 10.º ano.

Português e Inglês, melhoram a comunicação na língua materna (da maioria dos alunos) e na língua estrangeira mais utilizada nas comunicações e negociações empresariais; que também é a língua em que se podem consultar mais documentos técnicos de programação.

3.3. DISCIPLINAS CIENTÍFICAS

As disciplinas gerais científicas são ***Matemática*** e ***Física e Química***, ambas modulares e com a missão de estabelecer as bases científicas das restantes disciplinas. Beneficiam de ser articuladas com as outras disciplinas explicitando essa dependência conceptual. No entanto, são dadas de forma autónoma e em conjunto com os alunos de outro curso profissional, dificultando as possíveis articulações. Alguns exemplos de particular interesse seriam: na Matemática, estudar os módulos de *Estatística* utilizando Folhas de Cálculo ou o de *Estatística Computacional*; na Física e Química, os módulos de *Circuitos Elétricos*, *Corrente Alternada*, *Equilíbrio de Oxidação-Redução, Eletroquímica*, *Ligações Metálicas, Materiais Cerâmicos e Compósitos*.

3.4. DISCIPLINAS TECNOLÓGICAS

O currículo do curso inclui três disciplinas para além daquela em que ocorrerá a intervenção. Estas disciplinas são de mais baixo nível, mas são igualmente importantes para perceber o funcionamento interno dos computadores. Fazemos uma breve descrição e avaliação da proposta curricular, sublinhando desde já a sobreposição de alguns termos que, abordados de diferentes pontos de vista, podem ser reforçados, mas também confundidos. Além disso, dado o reduzido número de horas destas disciplinas, seria preferível abordar os diferentes pontos de vista como discussão e enriquecimento, mais do que como repetição ou métodos alternativos para uma mesma aplicação. A atualidade dos temas nem sempre é adequada, mas está prevista a adaptação dos temas ao desenvolvimento tecnológico existente.

3.4.1. DISCIPLINA DE SISTEMAS OPERATIVOS

A disciplina de *Sistemas Operativos* (Pinheiro, 2005) foca-se sobretudo na componente de gestão dos sistemas de informação do curso, uma vez que estes são executados naqueles ambientes e têm que ser entendidos para se programar adequadamente nos diferentes sistemas existentes.

O programa inclui o funcionamento interno dos sistemas operativos (M1), os serviços disponíveis nos sistemas operativos (M3) e o seu uso no desenvolvimento de aplicações (M2). A instalação e manutenção de sistemas operativos, utilitários e gestores de dispositivo (*drivers*) é abordada como forma de enriquecer os ambientes de desenvolvimento ou de execução dos sistemas de informação. Aborda a gestão paralela de processos e atividades (multitarefa) e a programação concorrente, onde esses processos sincronizam. Termina com os sistemas *Open Source* (M4) e um modo opcional escolhido entre configurações avançadas e arquitetura.

Fica a faltar um breve estudo de evolução histórica dos sistemas operativos, a par da evolução dos processadores e das arquiteturas computacionais, na forma como lidam com um número crescente de recursos de processamento (*multicores*) mas que depois exigem melhor sincronização. Este tema é, aliás, abordado na bibliografia recomendada (Alves Marques & Guedes, 1998) (Tanenbaum, 1987). Falta também as noções elementares de sistemas de processamento distribuído, necessários quando a quantidade de dados a processar é demasiado grande para uma só máquina (base do *Big Data*; e da Inteligência Artificial com aprendizagem sobre essa quantidade de dados). Também a segurança deveria ser abordada mais cedo, uma vez que muitos dos conceitos são aplicados com mais eficiência se os sistemas os tiverem em consideração desde o início. Se forem "remendos" posteriores ficam muito mais vulneráveis a ataques (Shuaibu & Ibrahim, 2017). Deve ser ensinada a forma correta de proteger os utilizadores e a informação com os diferentes mecanismos de autenticação e autorização (Stallings & Brown,

2015). A leitura de algumas porções de código (bem escrito), realmente utilizado nos sistemas operativos, também poderia ajudar a esclarecer as dúvidas mais comuns. A modificação simples de um gestor de dispositivo (*device driver*) poderia ajudar a consolidar essa aprendizagem.

Alguns destes conceitos podem ser abordados nas disciplinas opcionais, mas não deveriam estar sujeitos à escolha dos alunos dada a sua importância para os sistemas desenvolvidos atualmente e a probabilidade dos alunos os virem a utilizar na sua vida profissional.

3.4.2. DISCIPLINA DE REDES DE COMUNICAÇÃO

Na disciplina de *Redes de Comunicação* (Carvalho, 2005) os alunos aprendem o funcionamento da transmissão de dados e a utilização de redes de comunicação de dados. Aprendem a instalar, a configurar sistemas de comunicação e a fazer diagnóstico de falhas e erros nas infraestruturas, incluindo a montagem e teste de cabos de rede; as camadas de rede dos modelos OSI e Internet (TCP/IP); e a resolução de nomes com DNS (M1). Ficam a conhecer os tipos de cabo existentes (STP, UTP, coaxial, fibra ótica, 10/100/1000 Base Tx) e as topologias de redes (*bus*, *star*, *(dual) ring*, *tree*, *mesh*) bem como os seus elementos constituintes (*hubs*, *bridges*, *switches*, *routers*), comunicação síncrona e assíncrona, e compressão de dados (M2 e M3).

O primeiro módulo opcional permite o enriquecimento das páginas *web* dinâmicas (M5) com acesso a bases de dados que forneçam conteúdos dinâmicos. O segundo enriquece a disciplina de sistemas operativos com arquitetura de redes. Se for bem articulada, permite a interligação das duas disciplinas no nível conceptual adequado. Caso contrário, poderá ser apenas a repetição dos conceitos já abordados, mas agora num nível mais baixo de *hardware*, abaixo do nível da interface de programação com *sockets*. O terceiro ensina a caracterizar, instalar e configurar serviços de DHCP, DNS e encaminhamento, e servidores de páginas *web* (IIS e Apache). Este módulo complementa os módulos 4 e 5 ao abordar os serviços necessários para disponibilizar páginas *web,* mas, mais uma vez, tem que ser bem articulado com a disciplina de sistemas operativos para não haver simples repetição de conteúdos. Finalmente, o quarto módulo opcional aborda a instalação, configuração e gestão de servidores de e-mail (POP3 / SMTP / IMAP / NTP / SSL) e as arquiteturas cliente-servidor que já são tratadas em módulos da disciplina de sistemas operativos. É um tema menos articulado com os restantes, mas muito necessário nos ambientes empresariais.

Ficam a faltar noções sobre os sistemas operativos mais comuns: os dos dispositivos móveis, sobretudo de Android. Também seria interessante trabalhar com Raspberry Pi no âmbito da Internet das Coisas (Norris, 2015) onde a infraestrutura de rede adquire especial importância como esqueleto de todo o sistema. Ainda os dispositivos vestíveis (*wearables*) e a sua utilização em aplicações de saúde (Metcalf, Milliard, & Gomez, 2016).

3.4.3. DISCIPLINA DE ARQUITETURA DE COMPUTADORES

A disciplina de Arquitetura de Computadores pretende que o aluno adquira conhecimentos sobre a estrutura e organização de computadores e microprocessadores (Rodrigues, 2005).

Começa ao mais baixo nível, com os sistemas de numeração binário, decimal, octal e hexadecimal e a conversão entre sistemas; continua com as operações lógicas elementares (*OR*, *AND*, *NOT*) e a sua implementação em circuitos lógicos. Prossegue com as restantes portas lógicas (NAND, NOR, XOR, XNOR). Termina com a Álgebra de Boole, os teoremas de DeMorgan e a representação de portas em esquemas, incluindo os símbolos lógicos IEEE/ANSI.

No segundo módulo aborda o conhecimento e reconhecimento dos componentes de um computador, sua instalação, configuração, manutenção e substituição. Aborda também os periféricos habituais de um computador. O contínuo desenvolvimento tecnológico destes componentes exige a atualização sistemática destas listas, tanto mais que os alunos poderão ser responsáveis pela aquisição, instalação, e manutenção destes equipamentos na sua vida profissional futura. O terceiro módulo obrigatório dedica-se à obtenção (e correção) de avarias ao nível do *software* e do *hardware*. Apresenta os problemas típicos e frequentes (identificação) dos componentes periféricos estudados no módulo anterior, e os códigos de *beeps* e as configurações de *setup*, *post* e arranque. Termina com os procedimentos de substituição de componentes.

Dada a quantidade de dados que hoje se armazena na nuvem (*Cloud*), com serviços como *DropBox*, *OneDrive* ou *Google Drive*, fica a faltar um módulo dedicado a este tipo de serviços, sua comparação e métodos de salvaguarda (*backup*) de dados assim armazenados, bem como a correção de conflitos de sincronização de dados ou a gestão da quota de espaço de armazenamento.

O primeiro módulo opcional aborda a arquitetura de microcomputadores e o desenvolvimento de sistemas com microprocessadores e microcontroladores, e a interligação de dispositivos com interface em "*bus*". Não está previsto, mas dada a acessibilidade financeira de *Raspberry Pis*, *Arduinos*, ou *micro:bits*, ou *LittleBits*, seria interessante que este módulo os utilizasse, permitindo a programação simples de pequenos serviços. A maior limitação será sempre o número reduzido de horas deste módulo, mas, havendo interesse, poderia ser complementada com pequenos *workshops*. O segundo dedica-se à programação de microprocessadores através da aprendizagem de linguagem máquina *ASSEMBLY*. O terceiro propõe a instalação e configuração de redes locais. Este é um tema mais da área da disciplina de Redes de Comunicação, mas é abordado do ponto de vista da arquitetura de computadores, dos componentes necessários à sua instalação. Mais uma vez, deve ter em conta aquilo que já foi aprendido, de forma a complementar e reforçar as aprendizagens. O quarto dedica-se à manufatura de circuitos impressos.

Mais uma vez, a utilização de componentes em *breadboards* poderia ser suficiente, aproveitando melhor o número reduzido de horas (24 horas, 29 tempos de 50 minutos, 14 aulas de 100 minutos). A soldadura é um processo interessante, mas provavelmente fora de tópico para o perfil destes alunos. A ser, num contexto específico, seria boa prática fazê-lo com o módulo OP1 que serviria de base e permitiria a construção de um circuito em *breadboards* deixando para este módulo apenas as questões técnicas de impressão de circuitos.

3.5. DISCIPLINA DE PROGRAMAÇÃO E SISTEMAS DE INFORMAÇÃO

A disciplina em que decorrerá a intervenção é aquela a que o plano curricular dedica mais horas, mais de metade no caso, e depois do Decreto de Lei 55/2018, com cerca de 600 horas de um total de 1100 (versus as 134 de Arquitetura de Computadores, as 144 de Sistemas Operativos e as 234 de Redes de Comunicação, cf. Tabela 3.1). É, reconhecidamente, a disciplina mais importante do curso, tanto que o próprio programa (DGFV, 2005) recomenda o desdobramento de turmas para rentabilizar o tempo a ela dedicado.

Esta disciplina deveria, em abono da clareza pedagógica, corresponder a três disciplinas diferentes: Introdução à Programação, Programação com Objetos e Introdução às Bases de Dados. A primeira dedicada aos conceitos iniciais da programação, tais como algoritmos, variáveis, ciclos, condicionais (primeiros 7 módulos); a segunda dedicada ao conceito de objeto e à sua programação (módulos 9, 10 e 11); e a terceira dedicada a bases de dados e sua utilização (módulos 12 a 15).

Tal como as outras, a disciplina está dividida em módulos e módulos tão pequenos que acabam por não fazer muito sentido. A possibilidade de dividir módulos em períodos já não é muito adequada; a divisão por anos letivos é ainda pior, dificultando a mobilidade dos alunos que, a terem necessidade de mudar de escola, podem perder o reconhecimento do trabalho desenvolvido no ano anterior.

A disciplina propõe os conteúdos técnicos habituais em cursos iniciais de programação (Schulte & Bennedsen, 2006) e propõe competências típicas do século XXI, a saber, a aprendizagem ao longo da vida, a adaptação a novas situações, resolução de problemas, pensamento crítico, e trabalho em equipa. Estas competências foram reforçadas com o Perfil do Aluno à saída do Ensino Obrigatório (Direção Geral da Educação, 2017).

A disciplina inclui tarefas de nível superior de abstração que, na prática, não são implementadas de forma formal, explícita: analisar as linguagens de programação existentes; e estudar a semântica das linguagens de programação. Pelo observado nas várias planificações, em várias escolas, os alunos utilizam várias linguagens de programação, mas não sobem a "escada" da abstração para as analisar formalmente, nem para as comparar (Victor, 2011).

As técnicas de implementação de interpretadores, a noção de *Scanner* e de *Parser*, e o funcionamento elementar de compiladores também não são, tipicamente, abordados de forma explícita, ou seja, apesar de referido no programa não é planeado tempo letivo para o seu estudo.

Fala-se apenas dos paradigmas imperativo e orientado a objetos. O paradigma funcional é referido, mas, talvez dada a sua complexidade, é reduzido à noção de recursão e a casos muito simples. No entanto, a sua aplicação mais sistemática permitiria uma melhor compreensão de ciclo de vida das variáveis e do seu âmbito (*scope*). Aliás, Bruce, Danyluk, e Murtagh (2005) defendem que a sua aprendizagem deveria anteceder a aprendizagem de vetores (*arrays*).

Tipicamente, no 10.º ano são lecionados os primeiros 7 módulos, dedicados à introdução à algoritmia (M1), mecanismos de controlo de execução (M2), programação estruturada (M3), estruturas de dados estáticas (M4), compostas (M5) e dinâmicas (M6), e tratamento de ficheiros (M7). No final deste ano, por isso, os alunos conseguem fazer programas de consola (linha de comando) com alguma complexidade, interação simples com o utilizador e leitura e escrita de ficheiros em formatos simples. Durante esse ano, os alunos têm, pelo menos, 14 anos, completando 15 até ao fim do ano civil, dependendo do percurso académico e das retenções que tenham tido.

Em algumas escolas, conforme a carga horária atribuída à disciplina, há tempo para lecionar o módulo 8 ainda neste primeiro ano. Este dedica-se a "conceitos avançados de programação" e aborda "um novo paradigma", os ambientes gráficos e a programação por eventos e filas (*queues*). Estes temas já são abordados, em parte, na disciplina de Sistemas Operativos e, neste contexto, no fecho de um ano, ou na abertura do segundo, configuram-se um desvio pedagógico, com pouco valor, e já desatualizado. Sendo o objetivo abordar paradigmas recentes, seria mais interessante abordar o desenvolvimento de aplicações móveis (Papadakis & Orfanakis, 2018). Os conceitos técnicos são semelhantes, mas o valor para o mercado de trabalho será, provavelmente, maior.

No segundo ano do curso, tipicamente, os alunos têm os módulos seguintes de Programação Orientada a Objetos (M9, M10 e M11), seguidos dos módulos de Sistemas de Informação (M12) e de Modelação (M13), Manipulação (M14) e Definição (M15) de Dados.

O módulo 16 dedica-se a “Projeto de Software”, mas não só do ponto de vista teórico, da gestão. Este “supermódulo” – assim descrito no próprio programa da disciplina – tem como duração de referência 74 horas, sendo, assim, o maior módulo de todos, com mais do dobro das horas de todos os outros. Nele, espera-se a aplicação de tudo aquilo que se aprendeu durante o curso para implementar um projeto. Começando pelo levantamento dos requisitos segundo a técnica de Gestão de Projetos escolhida (à partida, de forma completa se usarem o modelo *Cascata*; ou à medida de cada *sprint*, se usarem *SCRUM* ou outra técnica ágil de gestão de projetos).

Podem utilizar-se técnicas de ensino inovadoras, baseadas em jogos, para ensinar SCRUM: faz-se um projeto de uma cidade fazendo o planeamento e gestão com as técnicas próprias do método que queremos aprender (Krivitsky, 2017) (Carmo, Pardal, & Venâncio, 2013).

A disciplina termina onde, talvez com proveito pedagógico e prático, poderia ter começado, com técnicas de Gestão de Projetos e seu faseamento (desenho, implementação, documentação e apresentação de projetos e produtos). Mas ficam a faltar as técnicas ágeis de gestão de projetos que são muito utilizadas em pequenas e médias empresas, e, sobretudo, em *Startups*, onde se quer testar o produto e a sua viabilidade tão cedo quanto possível. Se lecionadas no início, associadas às práticas de gestão de pessoal típicas dessa metodologia (como o *Pomodoro*, por exemplo), poderiam contribuir para melhorar a gestão de tempo dos alunos, para dar visibilidade ao trabalho desenvolvido e para a autorregulação das aprendizagens. A escolha do módulo final da disciplina prende-se, provavelmente, com a necessidade de utilizar essas técnicas na "Formação em Contexto de Trabalho" preparando a "Prova de Aptidão Profissional" com que os alunos completam o seu curso. Ainda assim, antecipá-las facilitaria a sua utilização neste contexto.

Os alunos têm ainda três módulos opcionais de 30 horas, escolhidos pela escola de um elenco de sete possibilidades. O primeiro trata de métodos de acesso remoto a bases de dados bem como das arquiteturas de suporte à sua utilização. O segundo explora métodos e ferramentas de deteção e tratamento de erros (*debug*). Mas deveria também falar em prevenção de erros ensinando técnicas de organização de código, refactorização, simplificação e generalização. O terceiro módulo apresenta metodologias de análise e desenvolvimento de sistemas com ferramentas CASE.

O quarto introduz os alunos aos conceitos de organização e gestão de empresas, dando a conhecer as funções tipicamente existentes, as tarefas administrativas necessárias ao seu funcionamento, e os circuitos de informação interna. Os alunos ficam assim a conhecer a nomenclatura utilizada em ambientes profissionais e é explicitado o papel das tecnologias como suporte ao trabalho. Fica a faltar uma breve análise crítica histórica das vantagens e dos problemas trazidos pela adoção de sistemas de informação, possivelmente com recurso ao estudo de casos notáveis (Laudon & Laudon, 2019).

O quinto módulo dota os alunos de conhecimentos avançados na utilização de ferramentas de desenvolvimento de páginas web. Alguns tópicos podem já ter sido abordados noutros módulos, pelo que é necessário ajustar adequadamente a planificação para que seja útil aos alunos.

O sexto atravessa o espectro dos conceitos de programação e dedica-se a ferramentas de animação gráfica para criar animações tridimensionais. Foi introduzido para ensinar a linguagem *Adobe Flash* muito em voga na altura, mas que está a ser progressivamente abandonada, não sendo utilizada em dispositivos móveis (Adobe, 2020). Atualmente, o melhor será utilizar HTML5.

Este módulo é inesperado na sequência pedagógica, mas poderia ser utilizado para introduzir os conceitos de programação com objetos com programas como o Alice (Cooper, Dann, & Pausch, 2000): uma linguagem gráfica de manipulação de objetos virtuais, que pode ser uma boa escolha como primeira linguagem (Mullins, Whitfield, & Conlon, 2009), uma vez que a metáfora é clara.

Finalmente, o sétimo e último módulo opcional, trabalha o tratamento de imagens e as ferramentas disponíveis para o fazer. Este módulo trata dos mesmos tópicos que os módulos *Aquisição e Tratamento de Imagem Estática* e *Aquisição e Tratamento de Imagem Vectorial*. Sendo uma competência transversal pode, com benefício, ser ensinada a todos os alunos da turma, em conjunto.

Concluindo, o programa desta disciplina é extenso e bastante completo. Ainda assim, deixa de fora alguns tópicos interessantes como os mecanismos de controlo de versões, de revisão de código por pares e de sincronização de trabalho de equipa. Estes tópicos podiam ser abordados em contextos de *workshops*, transversais a todas as disciplinas.

De facto, estes temas, bem como a utilização de editores de texto simples para correções pontuais são sistematicamente deixados de fora em cursos introdutórios. Aliás, recentemente, o MIT criou uma disciplina nova, *The Missing Semester of your CS Education* (Athalye, Gjengset, & Ortiz, 2020), onde ensina ferramentas de consola, escrita de programas de *script* de consola (bash, zsh), editores leves (como o Vim, NotePad, NotePad++, GNU Emacs), ferramentas de processamento de texto (grep, sed, sort, uniq, head, tail, awk), controlo de versões (CVS, SVN, Git), técnicas de deteção e resolução de erros (debugger, strace, time, perf), teste de software (*test suite: unit test, integration test, regression test, mocking*) escrita de artigos científicos (LaTeX, BibTeX, plot, TikZ), segurança, privacidade e criptografia.

Além das quatro disciplinas técnicas, os alunos têm **Formação em Contexto de Trabalho**. Tipicamente, as 600 a 840 horas são divididas entre o 2.º e 3.º anos do curso: no final do 11.º ano é realizado um pequeno estágio visando um primeiro contacto com o mundo do trabalho; no 12.º ano é realizando um segundo estágio, bastante maior, onde é desenvolvido o projeto de caráter profissional, interessante, exequível e pluridisciplinar conducente à realização da Prova de Aptidão Profissional, com que o curso termina. No âmbito desta disciplina é produzido um relatório que, tipicamente, tem uma série de entregas que registam e acompanham o desenvolvimento do projeto. Há ainda uma apresentação final do projeto e do relatório, com defesa do trabalho perante um júri constituído pelo Diretor Pedagógico, que preside; o Coordenador do Departamento; o Diretor de Curso e o Diretor de Turma; um dos professores orientadores do projeto; um representante de uma associação profissional ou empresarial e um de uma associação sindical e uma personalidade de reconhecido mérito na área da formação profissional.

3.5.1. MAPAS DE CONCEITOS

A disciplina agora apresentada inclui, como já vimos, três grandes tópicos: Programação e Algoritmia; Programação com Objetos; Sistemas de Informação e Bases de Dados (cf. Figura 3.1).

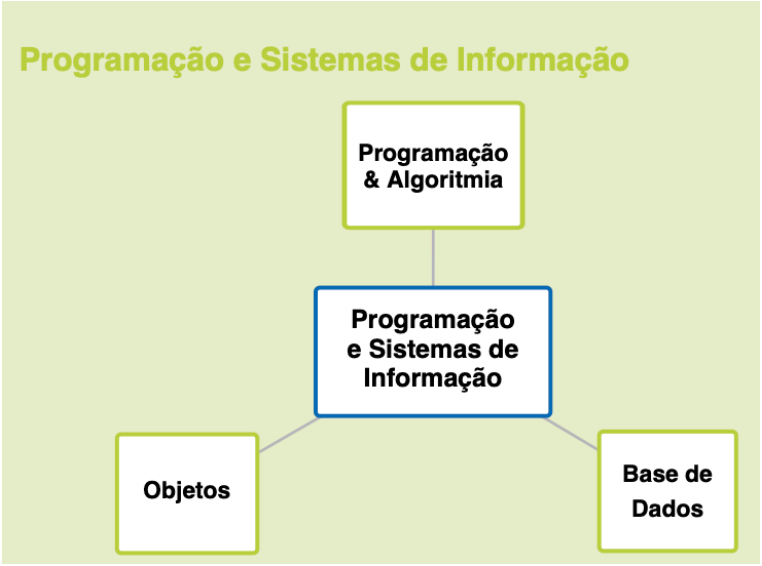


Figura 3.1: Componentes principais da disciplina de Programação e Sistemas de Informação

Os módulos relativos à Programação relacionam-se entre si, incluindo aquele em que decorrerá a intervenção (cf. Figura 3.2). Estas dependências devem ser respeitadas de modo a enriquecer o módulo 4.

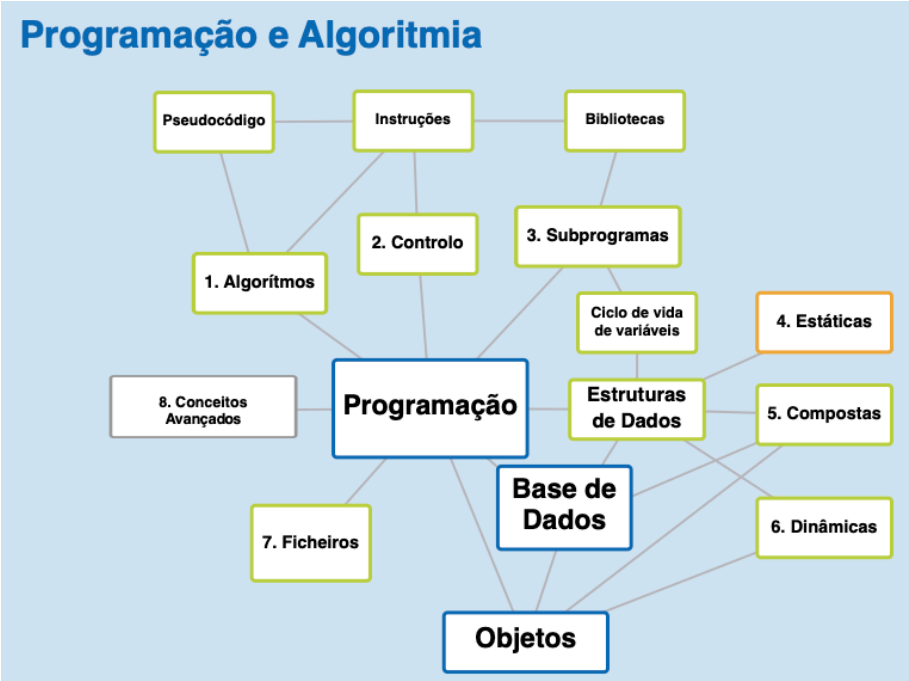


Figura 3.2: Conceitos relacionados com os módulos de Programação, onde se inclui o da intervenção.

Figura 3.3: Conceitos principais do módulo de Estruturas de Dados Estáticas, em que decorrerá a intervenção.

A Figura 3.3 detalha os conceitos relativos ao módulo em que decorrerá a intervenção: Estruturas de Dados Estáticos. Inclui *Strings*, cadeias de caracteres e vetores de tipos simples. Referem-se também as operações principais que se realizam com e sobre estas estruturas e que são referidas no programa da disciplina: declaração, inicialização, manipulação; inserção, pesquisa, remoção; e a ordenação.

Há ainda a referir algumas tarefas típicas: escrita de todos os valores; escrita de todos os valores maiores ou menores que um dado valor; procura do maior e menor elemento; procura do elemento mais comum (moda); cálculo da média dos valores de um vetor; contagem dos valores superiores ou inferiores à média; procura de subsequências; união e interseção, soma e subtração de valores; rotação de elementos.

Este tópico poderá ser enriquecido mais tarde, na aprendizagem de Programação com Objetos, analisando as interfaces disponibilizadas pelas linguagens utilizadas, o que dará uma ideia mais clara, também, de quais as melhores estruturas para cada tipo de problema.

A compreensão destes algoritmos implica também a comparação da eficiência dos diferentes métodos e a sua análise crítica ponto em evidência o melhor, o pior e o caso médio. Sendo esta tarefa de um nível conceptual elevado pode ser necessário simplificá-la considerando fatores mais concretos como o tempo de execução ou número de visitas a cada posição do vetor durante a realização da tarefa. Aprender a medir os tempos de execução é, em si, uma boa aprendizagem.

3.5.2. DIFICULDADES ASSOCIADAS ÀS TEMÁTICAS

As principais dificuldades associadas à aprendizagem da Programação, de forma geral, incluem a complexidade associada à análise *top-down*, a falta de bons hábitos de estruturação, a má interpretação de especificações e a falta de teste e *debug* (Ulloa, 1980).

Aprofundando o tema, temos o estudo de Dale (2006) que inqueriu cerca de 350 professores num formulário *online* sobre ***o tópico mais difícil de ensinar nos cursos introdutórios***. As respostas foram segmentadas em quatro categorias:

① resolução de problemas (desenho e implementação de algoritmos);
② temas gerais de programação (ciclos, parâmetros, vetores e ficheiros);
③ conceitos orientados a objetos (desenho de classes, polimorfismo e herança); e
④ maturidade dos alunos, ou falta dela (falta de trabalho, não pensam no que vão codificar, absentismo, capacidade de estudo, e má gestão de tempo).

De facto, programar é muito mais do que, simplesmente, copiar instruções ou pedaços de código de folhas de apoio, por isso a primeira dificuldade com que os alunos se deparam é a de compreender aquilo que estão a escrever (ou a copiar) e qual o papel de cada instrução na construção do programa final (Bosse & Gerosa, 2016). Aqui um grande entrave à compreensão por parte dos alunos é a sua recusa em ler com atenção os materiais que lhes são fornecidos (fator referido nos inquéritos de Dale). Assim, aumentar a quantidade de texto, detalhar as explicações ou incluir exemplos diferentes tem efeito reduzido no desempenho final. Encontrar o nível de detalhe mais adequado é, em si, uma tarefa difícil e alvo de investigação (Bolkan & Goodboy, 2019). O desenho dos materiais pedagógicos deve procurar encontrar esse equilíbrio.

No que se refere à aprendizagem das primeiras estruturas de dados, normalmente as listas ou vetores, Vrachnos & Jimoyiannis (2017) indicam que os equívocos mais comuns se prendem com a compreensão do conceito de variável que nem sempre é bem apreendido anteriormente.

Espantados com um estudo que refere que, mesmo no Ensino Superior, alguns alunos chegam ao fim da cadeira introdutória de programação sem saber programar (McCracken, et al., 2001), Danielsiek, Wolfgang, & Vahrenhold (2012) analisaram os equívocos que estão na base desses resultados, focando a área dos Algoritmos e Estruturas de Dados e concluíram que muitas vezes a própria nomenclatura é estranha aos alunos o que dificulta a comunicação. A utilização de cartões com as definições dos termos melhorou os resultados.

Este resultado não é surpreendente porque, dependendo do estágio de desenvolvimento cognitivo dos alunos, é possível que aquilo que lhes tentamos ensinar ainda não lhes seja acessível no formato proposto. Se os termos utilizados lhes forem estranhos, os resultados serão ainda mais desanimadores. E se assim é na universidade, pior será com os alunos mais novos.

No seu Guia para o Ensino de Ciência da Computação, Hazzan, Lapidot, & Ragonis (2014), exemplificam o método de planificação do ensino com o tema dos *vetores unidimensionais*. Tal como neste relatório, identificam várias **dificuldades** na sua aprendizagem:

- confusão entre a posição no vetor e o conteúdo da célula nessa posição;
- não compreender que além de construir a estrutura do vetor também é preciso construir as estruturas ou classes que vão ocupar cada célula;
- papel desempenhado pela célula nos vários algoritmos.

Identificam também os **equívocos** mais comuns:

- assumir a necessidade de visitar todas as células de um vetor;
- pensar que imprimir o vetor é imprimir o conteúdo das suas células.

Incluem ainda um conjunto de **erros** frequentes:

- índice ultrapassa o tamanho limite do vetor (*out of bounds*);
- perda de algum valor durante a iteração porque se escreve em cima ou por não visitar alguma célula;
- buracos no meio dos vetores (em vez dos valores ocuparem posições consecutivas).

Sugerem também algumas soluções a implementar no ensino da temática:

- utilização de jogos educativos visuais;
- isolamento das funções repetidas em funções que descrevam o que é feito em cada passo da iteração sobre a célula atual do vetor;
- escolha da ordem de apresentação das estruturas de dados;
- escolha do primeiro algoritmo de ordenação a ensinar;
- utilização da história da tecnologia para introduzir novos algoritmos, identificando a sua razão de ser.

Outra dificuldade que é preciso referir é a linguística: as linguagens de programação usam uma espécie de inglês estruturado. No entanto, os alunos não usam essa língua com a devida proficiência o que os obriga a mais uma operação cognitiva quando pensam naquilo que querem codificar e o traduzem para a linguagem de programação. Para minimizar esta dificuldade, os alunos começaram por utilizar o Portugol (Manso, Oliveira, & Marques, 2009), uma linguagem de programação que utiliza o português estruturado permitindo a escrita de código na língua nativa dos alunos. O IDE tem também um editor gráfico de fluxogramas. Este tipo de ferramentas pretende tornar o (primeiro) ambiente de programação mais amigável (Dasgupta & Hill, 2017) mas depois é necessário fazer a transição para a linguagem de programação tradicional (Baldwin & Macredie, 1999)que vai ser usada nos restantes módulos (Weintrop & Wilensky, 2019).

3.5.3. CASOS NOTÁVEIS DE ENSINO DESTAS TEMÁTICAS

"Sabe-se muito pouco quando só se sabe o que é indispensável" Max von Laue, Prémio Nobel (1914) Físico Matemático alemão

Para bem ensinar qualquer temática, um professor deve munir-se de conhecimentos daquilo que vai ensinar, mais aprofundados e mais abrangentes do que lhe parece que os alunos vão aprender (Ball, Hill, & Bass, 2005). Essa maturidade sobre os temas permite-lhe estabelecer as ligações conceptuais que enriquecem as suas aulas e lhe permitem entrar nas discussões temáticas que possam surgir e que podem levar os alunos um pouco mais longe do que o inicialmente previsto. Permite também guiar os alunos que queiram ir mais longe. Assim, para preparar as aulas, foram consultados vários livros temáticos e observadas aulas *online* sobre os tópicos a abordar.

Dos vários cursos encontrados, o mais interessante foi o curso inicial de programação, CS 50, da Universidade de Harvard (Malan, 2019). Este MOOC (do inglês *Massive Open Online Course*) tem objetivos bem definidos, e o currículo parecido com o dos primeiros módulos da disciplina. Como se dirige a todos os alunos, de todos os cursos (e não aos futuros informáticos, como é o caso do CS 101), os temas são explicados partindo sempre da intuição dos alunos e de exemplos da vida real onde os conceitos tenham aplicação. Também a sequência didática escolhida (Malan, 2010) e a narrativa utilizada para ligar os temas é muito parecida com aquilo que se tinha inicialmente planeado, pelo que, serviu de confirmação do trabalho que se estava já a desenvolver. O visionamento das gravações destas aulas (MacWilliam, Aquino, & Malan, 2013) contribuiu para rever os conceitos e deu algumas pistas para tornar a articulação ainda mais elegante.

A primeira aula, a aula 0, aborda os temas iniciais: resolução de problemas; entradas e saídas; representação da informação; unário, binário e decimal; ASCII e Unicode; RGB; algoritmos simples; tempos de execução e complexidade; e pseudocódigo. Termina com a apresentação da linguagem de programação por blocos SCRATCH e o trabalho de casa é fazer pequenos programas nesse ambiente de programação, aplicando tudo o que foi aprendido durante as aulas (Papancea, Spacco, & Hovemeyer, 2013).

A segunda aula, faz a transição para a linguagem C, apresentando o Linux, a interface de linha de comando, o compilador e a execução na linha de comando. Fala ainda de funções, argumentos, valores devolvidos, variáveis, expressões booleanas, condições e ciclos. Termina com os tipos de dados, o *overflow* dos inteiros e a imprecisão dos números de vírgula flutuante.

Na terceira aula, trata das estruturas de dados estáticas, compostas e dinâmicas, explicando o processo de compilação do C: pré-processamento, compilação, *assembling* e depuração. Fala também de *arrays* e de *Strings* utilizando a criptografia e o binário como exemplo.

A quarta aula fala de algoritmos de procura e de ordenação; da notação assintótica e da recursão. A quinta aula explica o funcionamento da memória: ponteiros, *segmentation faults,* alocação dinâmica de memória, *stack* e *heap*; estruturas de dados, ficheiros e imagens.

A sexta aula ainda utiliza o C e termina o assunto das estruturas de dados com as listas ligadas e duplamente ligadas, árvores, árvores binárias, *hash tables*, *tries*, *stacks* e *queues*.

Finalmente, a sétima aula fala de Python, o oitavo de SQL e Bases de Dados.

Para explicar a necessidade de vetores, utiliza o registo das notas de uma turma de 10 alunos. A primeira solução passa por usar 10 variáveis *nota1*, *nota2*, ..., *nota10*. Depois pede para calcular a média: tarefa simples ainda que demore a escrever todos os termos da soma (as dez variáveis). De seguida, pede para encontrar o aluno com melhor nota. A solução que decorre da solução encontrada é comparar as 10 variáveis até encontrar. Mas... se esta é uma boa solução, será que dá para estender até aos 100 alunos? E aos 1000? Queremos uma solução que funcione para os vários casos: uma boa solução resolve o problema de forma independente da quantidade de alunos.

Chegamos, assim, aos *arrays*. De seguida apresenta os métodos de inicialização e manipulação, mostrando o que se passa na memória do computador à medida que cada linha de código é executada. Então, mas se em vez de notas de alunos, tiver letras? Como faço? Segue-se a explicação das *strings* como cadeias de carateres que terminam com um símbolo especial, o '`\0`' também chamado de NULL. Para exemplificar os algoritmos de 'navegação' no vetor faz as contagens de letras, e aplica a cifra de César para mostrar como a representação das letras em ASCII pode ser útil para resolver problemas: convertendo o *char* para inteiro e somando o deslocamento pedido pela configuração da cifra.

Na aula dedicada à procura e ordenação, começa por ordenar objetos do dia-a-dia: livros. Podemos usar como critério a ordem alfabética do título do livro, ou do 1.º autor; ou o ano da 1.º edição, ou o ano daquela edição. Nos vários exercícios que faz, na nova versão do curso (2019/20), explicita o critério de comparação utilizado na pesquisa ou na ordenação. Também faz uma pesquisa binário numa lista telefónica, rasgando-a em partes à medida que avança na pesquisa. Isto ilustra a redução do espaço de procura, mas também mostra como não é preciso percorrer todas as posições de um vetor para ter sucesso na tarefa a que nos propomos.

Estas atividades desligadas permitem a compreensão intuitiva e em termos conhecidos, reduzindo a complexidade e o nível cognitivo. A leitura e o comentário do código que vai sendo escrito também é uma estratégia pedagógica extremamente interessante.

Este curso utiliza um IDE *online* (ide.cs50.io) onde os alunos podem escrever e copilar o seu código, podendo assim fazer todos os exercícios propostos *online*.

Outra cadeira muito interessante é *Programming for the Puzzled* do MIT (Devadas, 2017).

3.5.4. ESTRATÉGIAS E METODOLOGIAS DE ENSINO E APRENDIZAGEM

Dadas todas as condicionantes apresentadas, qual será, então, a melhor forma de ensinar estes conceitos de forma a que os alunos os aprendam da melhor maneira possível?

3.5.4.1. REFORÇO DAS TEMÁTICAS ANTERIORES

Aquilo que os alunos aprenderam nos módulos anteriores pode e deve ser utilizado em todos os módulos seguintes, sempre que possível. Se as variáveis, os ciclos e os condicionais são inerentemente necessários, a recursão e os subprogramas organizados em bibliotecas têm que ser explicitamente pedidos. Uma estratégia será que o código implementado é agrupado e organizado numa biblioteca própria que, no desafio final, é utilizada para resolver os enigmas de forma mais eficiente e rápida.

3.5.4.2. PREPARAÇÃO DAS TEMÁTICAS SEGUINTES

Para manter a narrativa utilizada consistente, é necessário destacar alguns dos problemas com que nos vamos deparando de forma a despertar nos alunos a curiosidade sobre a forma de resolver cada caso de uma forma mais elegante.

Exemplos: o terminador da cadeia de caracteres (NULL) desempenha na memória o mesmo papel que o *End-of-File* (EOF) terá nos ficheiros; utilizar a mesma posição de vários vetores de *Strings* para guardar informação 'alinhada' faz intuir que deverá haver uma solução mais elegante, levando às estruturas e aos objetos; o tempo necessário para rodar um vetor ou para ordenar o seus elementos fará desejar que haja uma estrutura mais eficiente; também as cópias que são necessárias quando intersectamos ou unimos dois vetores fará emergir a necessidade de armazenar o tamanho do vetor; finalmente, guardar o índice atual para se poder continuar iterações já começadas.

Este arco pedagógico não só prepara as aulas seguintes, como introduz momentos onde é possível recuperar equívocos ou erros que ainda se cometam, inadvertidamente.

3.5.4.3. CENÁRIO DE APRENDIZAGEM

O desenho dos cenários de aprendizagem deve ser alvo de cuidados de forma a que seja interessante e motivador ao mesmo tempo que introduz os alunos às temáticas que se quer abordar (Matos, 2010). Foi desenvolvido um cenário de aprendizagem para a intervenção que se apresenta no anexo A (*vide* Figura 9.2) onde se descreve o conceito da atividade a desenvolver com os alunos.

É sabida a influência do estilo dos professores (Bona, 2017) no ensino e aprendizagem. Dada a extensão da intervenção, adaptar-se-ão algumas estratégias 'fora da caixa' que serão alvo de avaliação, medindo o impacto que tiverem na aprendizagem realizada pelos alunos.

3.5.4.4. APRENDIZAGEM BASEADA EM PROJETOS

O programa da disciplina sugere que se utilize a aprendizagem baseada em projetos onde uma questão orientadora guia o processo de investigação para que, aplicando as competências do século XXI, os alunos encontrem uma solução e desenvolvam o projeto (*vide* Figura 3.4).

Figura 3.4: Aprendizagem baseada em Projetos, adaptado de Buck Institute of Education por (Pedro, Matos, Piedade, & Dorotea, 2017)

No entanto, a intervenção tem uma duração limitada, não há nenhum projeto de médio prazo a decorrer no âmbito da disciplina e os alunos ainda estão no início do seu percurso de aprendizagem. Estes fatores sugerem que esta não seja uma boa abordagem.

3.5.4.5. APRENDIZAGEM BASEADA EM PROBLEMAS

Neste tipo de aprendizagem, os alunos devem explorar os seus conhecimentos para tentar resolver um dado problema. Depois de identificarem os conhecimentos necessários para aprenderem a resolver o problema, os alunos investigam sobre o problema e procuram soluções. Se forem encontradas várias possibilidades para resolver o problema, têm que ser avaliadas para que se possa escolher uma e, finalmente, solucionar o problema. Termina com a apresentação dos resultados alcançados. Nesta metodologia, o aluno é responsável pela sua própria aprendizagem. Utilizaremos a aprendizagem baseada em problemas (*vide* Figura 3.5), fazendo corresponder cada problema resolvido a um carimbo no passaporte de aprendizagem (*vide* 3.5.4.9).

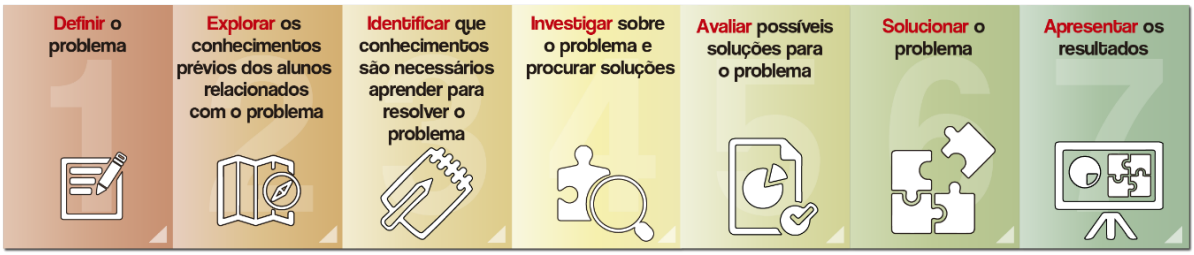


Figura 3.5: Processo de Aprendizagem baseada em Problemas (Pedro, Matos, Piedade, & Dorotea, 2017)

3.5.4.6. CONSTRUÇÃO DO CONHECIMENTO EM ESPIRAL

Muitas vezes, a melhor forma de construir o conhecimento sobre os conceitos abordados é em espiral, ou seja, voltando a eles com níveis de profundidade maiores de forma a que os alunos os revisitem com novos conhecimentos e os possam compreendam melhor (Hazzan, Lapidot, & Ragonis, 2015). Esta metodologia estabelece pontos de recuperação ou aprofundamento, conforme o desempenho anterior do aluno.

3.5.4.7. AULA INVERTIDA (*FLIPPED CLASSROOM*)

Outra estratégia que poderia funcionar é a da aula invertida (Lima, 2017) (*vide* Figura 3.6) onde os alunos são convidados a visionar algum vídeo introdutório em casa, de forma autónoma, nos seus dispositivos móveis, deixando o tempo de aula para resolver problemas propostos e esclarecer dúvidas que surjam.

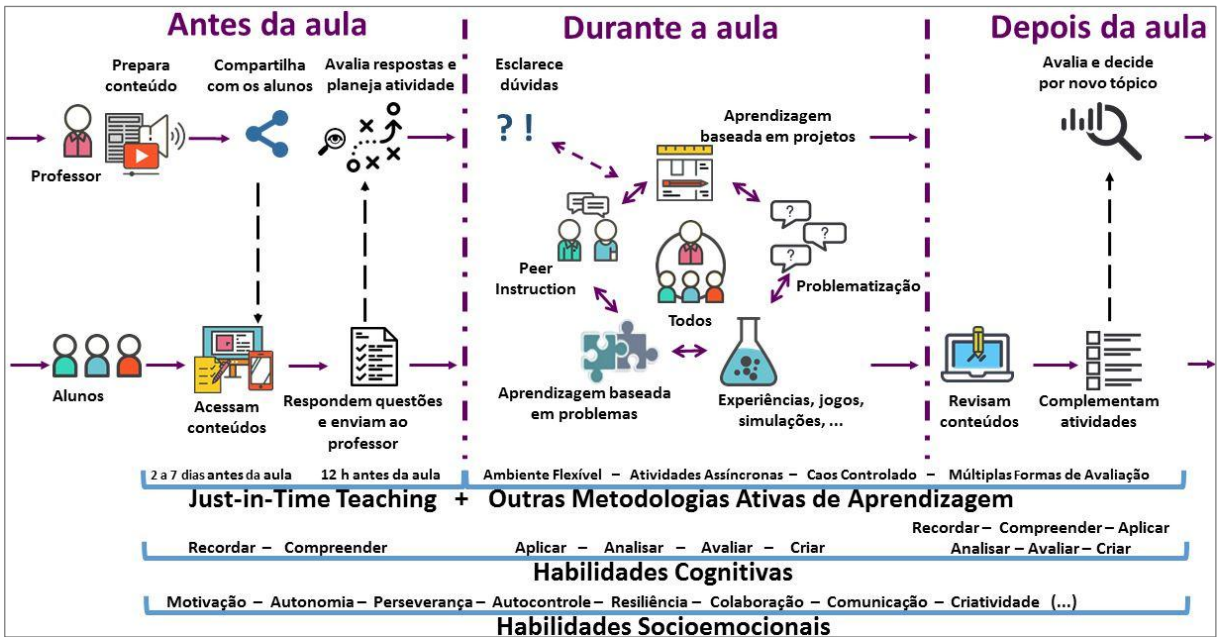


Figura 3.6: Integração do modelo de Sala de Aula Invertida com metodologias ativas de aprendizagem (Schmitz, 2016)

Aqui a dificuldade será garantir que todos os alunos fazem uma preparação semelhante em casa, uma vez que, como vimos (*vide* Figura 2.7), têm horário completo na escola, entrando diariamente às 8:15, para saírem às 18:10.

3.5.4.8. PROGRAMAÇÃO EM PARES (*PAIR PROGRAMMING*)

Havendo recursos suficientes, os alunos poderão trabalhar em pares, em *pair programming* (NCW&IT, 2009), cujos benefícios (Williams, Wiebe, Yang, Ferzli, & Miller, 2002) ultrapassam as desvantagens (Cockburn & Williams, 2001) sobretudo em fases iniciais de introdução à programação (Bryant, Romero, & Du Boulay, 2008).

Esta técnica prevê que um aluno seja o **condutor** e tente resolver os problemas, ao mesmo tempo que o copiloto, ou **navegador**, disputa as escolhas feitas e sugere alternativas (Williams, McCrickard, Layman, & Hussein, 2008). Esta técnica, ilustrada na Figura 3.8, melhora a retenção da aprendizagem e a confiança dos alunos, e melhora a qualidade dos produtos desenvolvidos (McDowell, Werner, Bullock, & Fernald, 2006).

3.5.4.9. PASSAPORTE DE APRENDIZAGEM

Preconizando-se a aprendizagem por problemas, será importante encontrar uma forma de registar o trabalho desenvolvido por cada aluno, e as soluções encontradas. Um passaporte de aprendizagem é um registo mantido por cada aluno onde se registam os progressos feitos pelo aluno. Serve para estabelecer os objetivos, guiar o aluno na sua persecução e motivar o aluno enquanto os tenta alcançar (Lai, Yang, Liang, & Chan, 2005). Este tipo de registo também permite explicitar, desde logo, qual o percurso pedagógico a realizar, contextualizando melhor o aluno que, dessa forma, pode autorregular a sua aprendizagem (Loksa, et al., 2016).

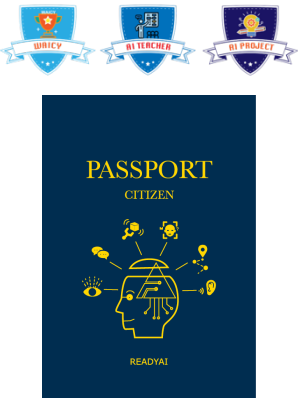


Figura 3.7: Distintivos e Passaportes de Aprendizagem de Inteligência Artificial da ReadyAI.

Se houver necessidade de algum aluno recuperar algum módulo, podemos explicitar essa diferenciação pedagógica incluindo mais selos e tarefas no passaporte desse aluno.

3.5.4.10. ENSINO DE LÍNGUAS ESTRANGEIRAS

Robertson & Lee (1995) investigaram quais os métodos pedagógicos utilizados no ensino de línguas estrangeiras que poderiam ser aplicados com sucesso na programação. Há, de facto, vários, desde logo a necessidade de exercitar todos os sentidos: por exemplo, o percurso didático do Inglês dos cursos profissionais prevê a sequência de passos: "Ouvir, Ler, Falar e Escrever" organizando a aprendizagem de acordo com a carga cognitiva de cada um. No ensino da programação também se deveria recorrer esses vários passos: ouvir as instruções de um dado algoritmo e executá-las pode ser algo tão simples como seguir uma receita falada; ler código bem escrito, com um desenho elegante, limpo; falar para explicar os passos que são necessários e como poderão ser dados; e só então escrever o código que funcionará melhor e com menos problemas. No entanto, tipicamente, saltamos as primeiras etapas e utilizamos metodologias mais experimentais, seguindo os métodos científicos, em vez dos matemáticos ou gramaticais, mais formais, a necessitar de mais tempo, mas mais certeiros nos resultados obtidos (Milková, 2015).

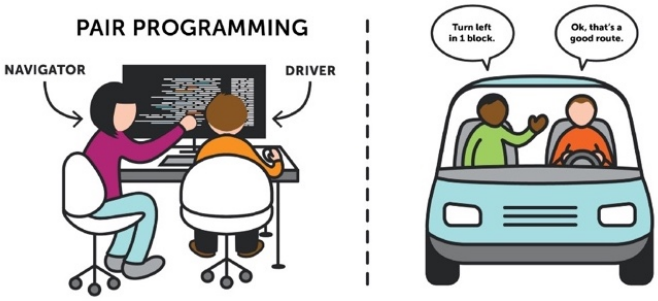


Figura 3.8: *Pair Programming* para Resolução de Problemas [Fonte: medium.com/@tomspencer_uk]

3.5.4.11. LER ANTES DE ESCREVER

Ko e a sua equipa tem-se dedicado à compreensão da influência de ler código e escrever o estado da memória daquilo que se lê (*tracing*) para conseguir escrever melhor código (Xie, et al., 2019). Procuram também perceber qual a diferença entre os alunos aprenderem apenas as *templates* de escrita de código, e conseguirem chegar ao nível semântico do código, criando assim quatro quadrantes leitura – escrita; *templates* – semântica (*vide* Figura 3.9).

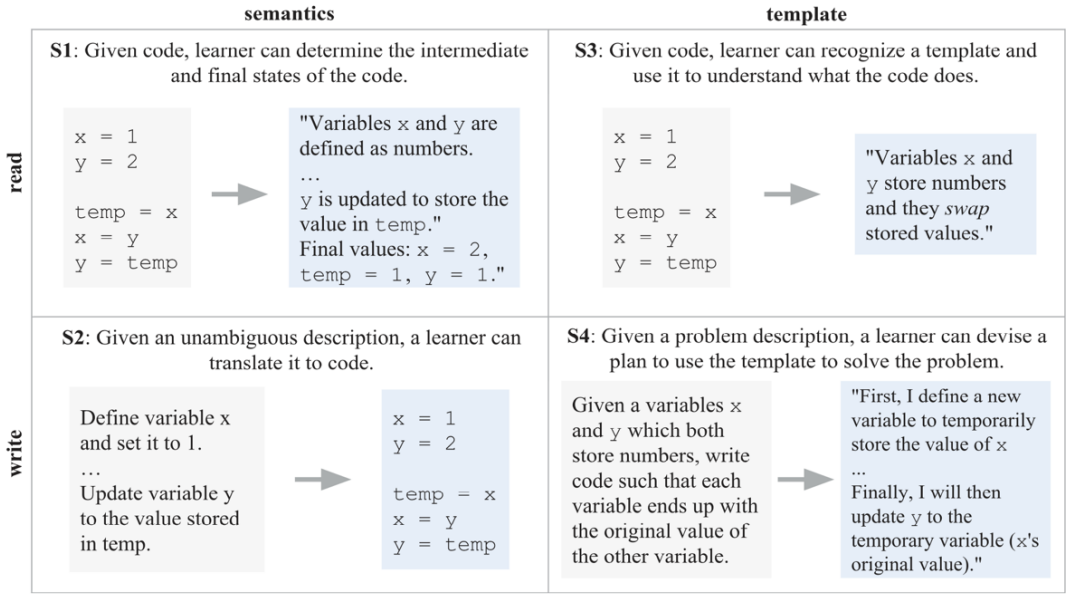


Figura 3.9: Conhecimento demonstrado em relação a cada um dos quatro quadrantes de programação (Xie, et al., 2019)

Esta teoria procura os elementos constitutivos da linguagem da programação. Tal como no método português da Cartilha Maternal de João de Deus (Deus, 1876) que parte dos sons de cada letra, para os juntar em ditongos, para os juntar em palavras começando pelos que têm menos valores possíveis e acabando na letra cuja leitura implica a aprendizagem de várias regras; e depois forma frases, cada vez mais complexas, até que o aluno saiba ler qualquer texto a um bom ritmo. Seria interessante encontrar estes elementos também nas linguagens de programação (e isso faremos nos próximos pontos).

3.5.4.12. REPRESENTAÇÃO PICTÓRICA DA MEMÓRIA: *TRACING*

Na metodologia de ensino da Matemática *GreatMath,* baseada no método de Singapura (Bennett, 2014), desenvolvida pela APECEF no Colégio de São Tomás e no Colégio de São José – Ramalhão, a representação pictórica dos exercícios, a chamada representação de barras, é o que permite que mesmo os alunos com mais dificuldades consigam recuperar e fazer o seu percurso académico com sucesso. Uma avaliação empírica deste método encontra alunos proficientes que utilizam métodos equivalentes de esquemas de forma natural; enquanto que os alunos com menor grau de eficiência se debatem com a própria compreensão daquilo que está a acontecer.

De forma equivalente, na programação, alunos proficientes são capazes de fazer representações esquemáticas daquilo que se passa na memória do computador à medida que um programa é executado, mas os alunos menos proficientes têm dificuldade até em descrever oralmente o que possa estar a acontecer. Assim, pareceu pertinente fazer um paralelo as duas estratégias, ditas boas práticas. De facto, há já um corpo de trabalho que aponta nessa entre direção.

Xie, Nelson, & Ko (2018) estudaram as tabelas de memória desenhadas por programadores aprendizes (*vide* Figura 3.10).

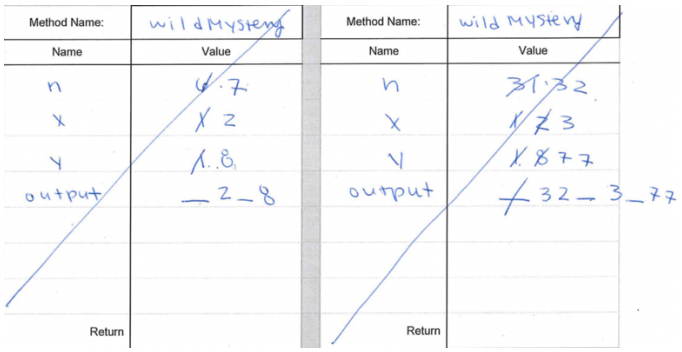


Figura 3.10: Duas tabelas de Memória para duas chamadas ao mesmo método (Xie, Nelson, & Ko, 2018)

Teague & Lister (2014) fizeram um estudo semelhante com um programador novato a quem pediram que explicitasse o seu raciocínio "pensando alto". Esta estratégia, também utilizada no estudo anterior, permite acompanhar os conceitos e que os programadores se apoiam enquanto resolvem um dado problema, os seus equívocos, e, claro, os seus erros.

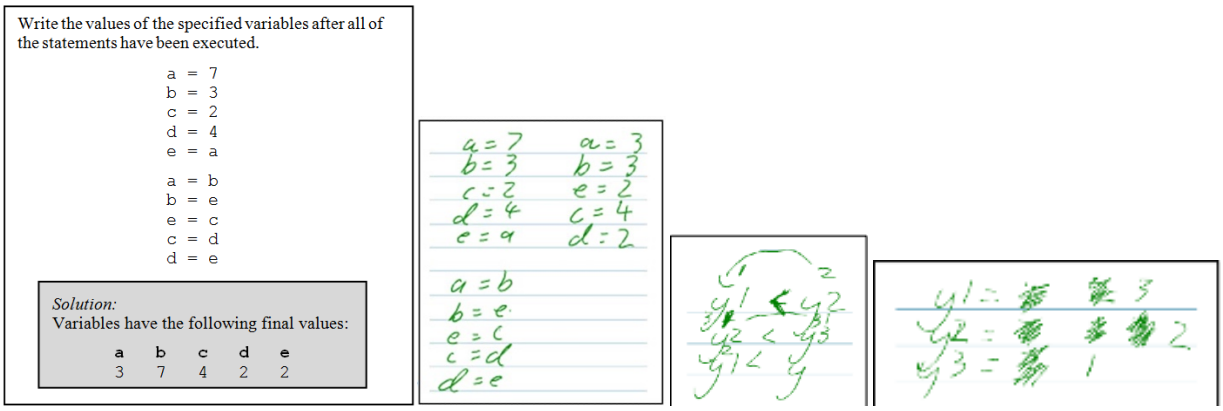


Figura 3.11:Evolução do registo do conteúdo das variáveis de um programa ao longo de várias semanas (Teague & Lister, 2014)

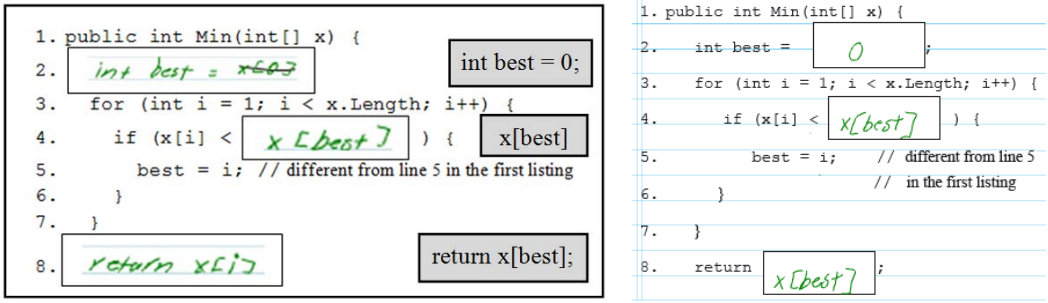


Figura 3.12: Evolução da compreensão de um *array* quando foi ensinado, e três semestres mais tarde (Teague & Lister, 2014)

3.5.4.13. PENSAMENTO COMPUTACIONAL

A avaliação que se faz da compreensão dos alunos é feita, normalmente, com base na escrita de código. No entanto, como vimos em 3.5.4.11 um aluno pode já ter competências de leitura de código mas ainda não ser proficiente na escrita. Tal como no estudo das línguas estrangeiras, mencionado em 3.5.4.10, a preocupação com sintaxe e a gramática deve vir depois do o aluno já ter adquirido algum vocabulário (ouvindo e lendo) e já se sentir confiante para arriscar falar, ou mesmo escrever. Também a representação pictórica é uma ajuda para aceder ao nível de compreensão do aluno, especialmente se acompanhada da explicitação do raciocínio (*vide* 3.5.4.12). Assim, para aceder aos níveis conceptuais atingidos pelos alunos podem usar-se descrições dos problemas e pedir a sua resolução esquemática, algo que é comparável ao longo do percurso inicial do aluno, uma vez que não depende da aquisição gramatical feita.

Poderíamos utilizar desafios matemáticos tradicionais, por exemplo de (Knop, 2019), mas essa estratégia traria um problema que é recorrente no ensino inicial da programação que é o recurso a conceitos matemáticos, eles próprios complexos, para ensinar os novos conceitos. Esta dependência faz depender o desenvolvimento do pensamento computacional, do pensamento matemático já adquirido (Knuth, 1985). Com a emergência do conceito de Pensamento Computacional (Wing, 2006), é possível medir novas competências enriquecendo as comparações. A acompanhar esta intuição está o desenvolvimento de competições de Pensamento Computacional (*vide* www.bebras.uk e bebras.dcc.fc.up.pt) , a par das já tradicionais competições de Programação (*vide* miup19.tecnico.ulisboa.pt e swerc.eu). Assim, usaremos vários cartões da competição Bebras inglesa (Dagiene, Futschek, Koivisto, & Stupurienė, 2017), com problemas de pensamento computacional para avaliar o desempenho e progresso dos alunos medindo o desenvolvimento através dessas tarefas (Lockwood, 2018) (Chiazzese, 2019).

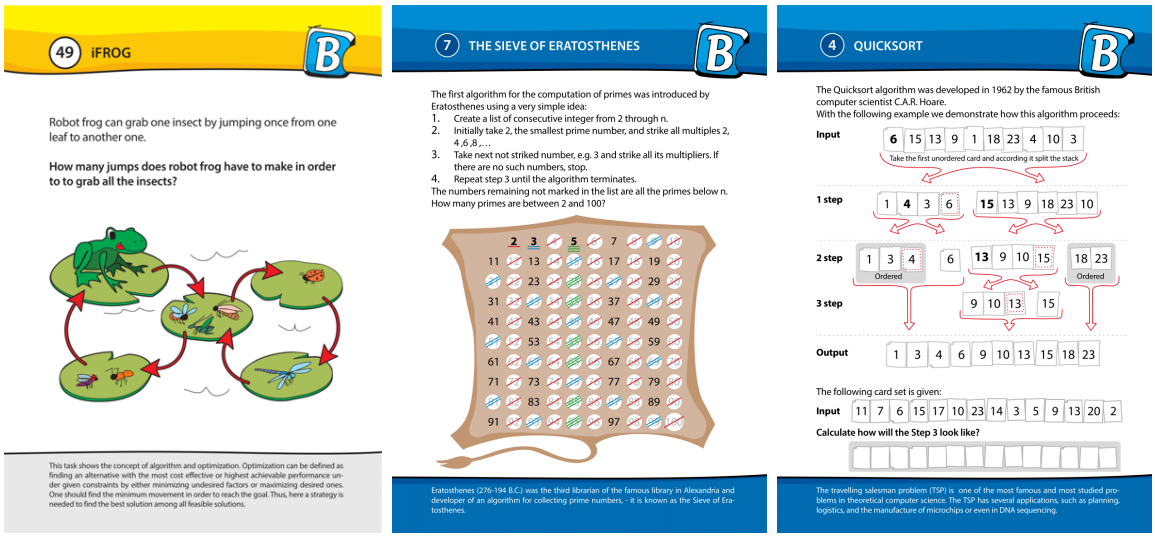


Figura 3.13: Exemplos de cartões da competição de Pensamento Computacional www.bebras.org (Dagiene & Stupuriene, 2017)

3.5.4.14. TAXONOMIA DE *BLOOM* APLICADA AO ENSINO DA PROGRAMAÇÃO

Feito este caminho de identificar as várias metodologias e estratégias que podem ajudar os alunos na aprendizagem da programação inicial, tornou-se necessário organizar as teorias e alinhando-as com a Taxonomia de *Bloom* (Bloom, 1956), uma ferramenta que permite categorizar e organizar objetivos de aprendizagem (Sprouts, 2019) de acordo com o nível cognitivo que implicam. À medida que subimos na complexidade cognitiva (lado esquerdo da pirâmide da Figura 3.14) também aumentamos a dificuldade pelo que, estes temas deveriam ser ensinados depois daqueles. Estes níveis cognitivos podem ser exemplificados com as competências de codificação que os alunos aprendem (lado direito da pirâmide) e estas podem ser observadas pelo prisma do pensamento computacional (tabela de termos do lado direito da figura).

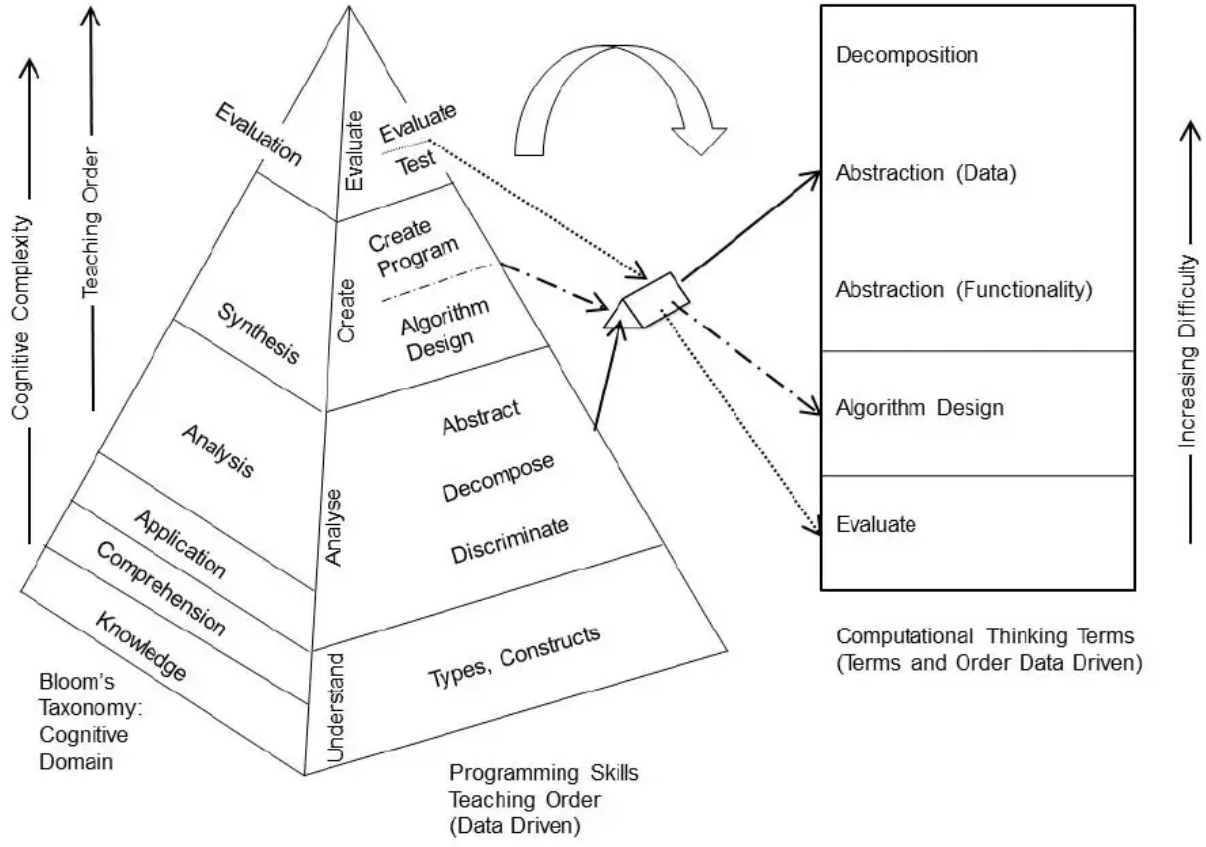


Figura 3.14: Modelo unificador do pensamento computacional, pedagogia da programação e a taxonomia de Bloom (Selby, 2015)

Esta teoria foi revista (Krathwohl & Anderson, 2009): passou-se a utilizar verbos em vez de nomes; e reorganizaram-se os dois níveis superiores passando a criação (anterior síntese) a estar num nível cognitivo abaixo da Avaliação. Esta alteração (ainda) não está refletida na Figura 3.14.

Na Figura 9.1 do Anexo A, pode ver-se outra apresentação das tarefas típicas da programação alinhadas com a Taxonomia de Bloom (Thompson, Luxton-Reilly, Whalley, Hu, & Robbins, 2008).

3.5.4.15. *UNPLUGGED COMPUTER SCIENCE* (ATIVIDADES DESLIGADAS)

Hoje em dia a informática está em todo o lado e todas as disciplinas a conseguem utilizar para melhorar o seu desempenho. Diante deste panorama, as necessidades de informáticos no mercado de trabalho dispararam pelo que as competências do século XXI foram tornadas obrigatórias nos anos de ensino obrigatório de muitos países, processo liderado pela UNESCO.

Acontece que para ensinar programação com os métodos tradicionais seria necessário equipar as escolas com computadores suficientes para os alunos. A maioria das escolas não tem capacidade financeira para fazer um investimento tão grande sem apoios estatais ou de benfeitores. Assim, a própria comunidade de professores começou a organizar-se para criar atividades que não precisassem de computadores mas que pudessem ser úteis para os alunos aprenderem as competências preconizadas (Bell, Alexander, Freeman, & Grimley, 2008) e (2009).

Foi criado um conjunto de 24 atividades ditas desligadas com que os alunos podem, à semelhança dos manipulativos de matemática, utilizar os conceitos para resolver problemas, mas sem utilizar computadores (*vide* csunplugged.org). Estas atividades promovem a oralidade e por isso constituem uma boa maneira de ensinar, mas também de aferir os raciocínios feitos pelos alunos (Bell & Vahrenhold, 2018). De há três anos a esta parte o *site* com as atividades tem estado a ser melhorado, mas as atividades antigas continuam a estar disponíveis (*vide* classic.csunplugged.org).

Na prática pedagógica que criámos na APECEF incluímos sempre algumas atividades com cariz desligado e a experiência que temos feito é que a sua utilização dá aos professores a oportunidade de 'observar os raciocínios dos alunos' e aos alunos a oportunidade de aprender resolução de problemas sem se distraírem com tecnologias.

Feaster, Segars, Wahba, & Hallstrom (2011) relatam uma experiência com alunos do ensino secundário, ou seja, da faixa etária dos da intervenção. Como resultado não melhorou a atitude dos alunos em relação à programação. No caso da intervenção, os alunos já têm uma boa atitude em relação à informática (Nishida, Idosaka, Hofuku, Kanemune, & Kuno, 2008), tanto que escolheram um curso profissional dessa área. Utilizaremos as atividades desligadas (Bell, Duncan, Jarman, & Newton, 2014) sobretudo para examinar os métodos de resolução de problemas já conhecidos (Chen, Lewandowski, McCartney, Sanders, & Simon, 2007), escrevendo os algoritmos na língua nativa – mantendo o nível cognitivo mais baixo do que se traduzirem para inglês ou alguma linguagem de programação (Roussel, Joulia, Tricot, & Sweller, 2017).

Com estas atividades, pode medir-se o pensamento computacional dos alunos pelo que, com os cartões Bebras que vimos (§ 3.5.4.13) e que também são atividades desligadas, teremos mais formas de avaliar os alunos (Rodriguez, Kennicutt, Rader, & Camp, 2017).

3.5.4.16. ENSINO GUIADO PELA CURIOSIDADE

A Educação guiada pela Curiosidade (L'Ecuyer, 2017) e apoiada no encontro com a Realidade (L'Ecuyer, 2018) traz algumas pistas interessantes para contornar o efeito que a presença contínua da tecnologia tem nos alunos. Esta teoria da aprendizagem tenta recuperar a curiosidade, fonte de espíritos científicos que, exploram o mundo para descobrir como funcionam as coisas.

3.5.4.17. GAMIFICAÇÃO

A gamificação é a inclusão de elementos de jogos em contextos diferentes dos habituais. No ensino, traduz-se em autocolantes e carimbos; níveis de conquista e barras de progresso; pontos, classificação e prémios; dinheiro virtual; prémios; desafios. O objetivo é motivar os alunos utilizando metáforas que são bem conhecidas de uma das suas atividades preferidas, os jogos, *online*, em consolas, telemóveis, ou de tabuleiro (Deterding, Dixon, Khaled, & Nacke, 2011).

Este método já foi utilizado com sucesso, diminuindo o abandono escolar numa área disciplinar que é trabalhosa e exige repetição laboriosa e tediosa (Butler & Ahmed, 2016).

As atividades desligadas já constituem uma certa gamificação, mas esta será completada com um ***escape room*** (Martens & Crawford, 2019), uma atividade onde os alunos têm que resolver uma série de desafios criptográficos para poderem abrir a porta e ficarem a 'salvo' (Ho, 2018).

3.5.4.18. *RUBBER DUCK DEBUGGING* (DEPURAÇÃO COM PATINHO DE BORRACHA)

Em engenharia de *software*, há um método de deteção e resolução de erros que consiste em explicar a um patinho de borracha aquilo que se codificou e o que se espera que aconteça (Hunt & David, 2000). Lendo o código, linha a linha, ao patinho, verbalizando aquilo que se fez e expondo o raciocínio seguido, é frequente encontrar o erro e conseguir corrigi-lo. Isto vem em linha com a investigação sobre a utilidade de visualizar ou esquematizar o funcionamento de um programa em memória (Nelson, Xie, & Ko, 2017). Este mecanismo cognitivo também nos dá pistas pedagógicas interessantes (Dastyni Loksa, 2016) sobre o tipo de *feedback* a dar aos alunos.

3.5.4.19. *ASK 3 BEFORE ME* (PERGUNTE A 3 ANTES DO PROFESSOR)

Seguindo a pista do *feedback* que acabámos de ver (§ 3.5.4.18), uma forma de ajudar os alunos a formalizar os seus raciocínios e a resolver os problemas e erros com que se deparam é adiar a ajuda que lhes damos. Seguindo o princípio de Maria Montessori de independência (Pound, 2014), a criança deve, primeiro que tudo, perguntar-se a si própria qual poderá ser o problema; de seguida pode perguntar ao patinho de borracha ou a um colega; se mesmo assim não conseguir, deverá procurar na documentação ou *online* em fóruns como o *Stack Overflow*. Só depois poderá chamar o professor.

3.6. MÓDULOS JÁ LECIONADOS: M1 E M2

Para adequar a planificação à turma e às suas características, e aos alunos e às suas características individuais, observaram-se algumas aulas do 1.º período e do início do 2.º. A Tabela 3.2 abaixo apresenta um resumo das temáticas abordadas e das atividades desenvolvidas.

	Data	Lição	Atividades Desenvolvidas / Sumário
Módulo I	17-09-2019	1	Apresentação do professor e dos alunos. Regras de funcionamento da disciplina. Conteúdos programáticos e aprendizagens essenciais da disciplina.
		2	Conceito de algoritmo.
	19-09-2019	3 e 4	Introdução à Lógica de programação.
	24-09-2019	5 e 6	Desenvolvimento de algoritmos em Portugol (texto).
	26-09-2019	7 e 8	Desenvolvimento de algoritmos em Portugol: conceito de variável. Tipos de variável.
	01-10-2019	9 e 10	Representação gráfica de algoritmos: fluxogramas: exemplos e exercícios
	03-10-2019	11 e 12	Trabalho prático: algoritmo em fluxograma para cálculo de áreas.
	04-10-2019	13 e 14	Trabalho prático: continuação de algoritmos em fluxograma para cálculo de áreas.
	15-10-2019	15 e 16	Trabalho prático: Construção de algoritmos em pseudocódigo.
	17-10-2019	17 e 18	Trabalho prático: Construção de algoritmos em pseudocódigo (continuação).
	22-10-2019	19 e 20	Miniprojecto: Equação do 2.º grau.
	24-10-2109	21 e 22	Conclusão do Miniprojecto: Equação do 2.º grau.
	29-10-2019	23 e 24	Correção do Miniprojecto: Equação do 2.º grau.
	31-10-2019	25 e 26	Miniprojecto: Cálculo de áreas.
	05-11-2019	27 e 28	Miniprojecto: Cálculo de áreas.
	07-11-2019	29 e 30	Miniprojecto: Cálculo de áreas.
	12-11-2019	**31 e 32**	**Miniprojecto: Algoritmo para uma Calculadora.** 👁
	14-11-2019	33 e 34	Miniprojecto: Programação de uma Calculadora no ambiente Portugol.
	19-11-2019	35 e 36	Exercício de revisão da matéria.
	21-11-2019	37 e 38	Teste de avaliação.
Módulo II	26-11-2019	1	Apresentação do Módulo. Instalação e preparação do ambiente de programação Visual Studio 2012. Criação de um projeto em C#: exemplificação. 👁
	26-11-2019	**2**	**Criação de um projeto em C#: continuação da exemplificação.** **Operadores aritméticos em C#.** **Estrutura de um programa e projeto em C#.** 👁
	28-11-2019	3 e 4	Exemplos de leitura e escrita em programas de consola em C#.
	03-12-2019	5 e 6	Operadores lógicos e relacionais em C#. Exemplos de aplicação.
	05-12-2019	7 e 8	Estruturas de decisão `if` e `switch`. Exemplificação.
	10-12-2019	9 e 10	Estruturas de repetição: `for`, `while`, do...`while`. Exemplificação.
	12-12-2019	11 e 12	Exercícios de aplicação: avaliação.
	17-12-2019	13 e 14	Exercícios de aplicação: estruturas de controlo de execução.
	07-01-2020	15 e 16	Correção dos exercícios anteriores. Esclarecimento de dúvidas sobre estruturas de controlo de decisão e repetição.
	09-01-2020	17 e 18	Correção dos exercícios anteriores. Esclarecimento de dúvidas sobre estruturas de controlo de decisão e repetição.
	14-01-2020	19 e 20	Correção dos exercícios anteriores. Esclarecimento de dúvidas sobre estruturas de controlo de decisão e repetição.
	16-01-2020	21 e 22	Correção dos exercícios anteriores. Esclarecimento de dúvidas sobre estruturas de controlo de decisão e repetição.
	21-01-2020	**23 e 24**	**Correção dos exercícios anteriores.** **Esclarecimento de dúvidas sobre estruturas de controlo de decisão e repetição.** 👁

Tabela 3.2: Sumários das aulas de PSI lecionadas antes da intervenção, com indicação das que foram observadas (👁).

3.7. MÓDULO 4: ESTRUTURAS DE DADOS ESTÁTICAS

O módulo onde decorrerá a intervenção debruça-se sobre estruturas de dados estáticas, em particular, as cadeias de caracteres (*Strings)* e os vetores[1] (*arrays*).

Pressupõe-se que os alunos já falaram dos mecanismos de controlo de execução disponíveis na linguagem para, depois de compreenderem o conceito dos vetores, os poderem 'varrer', ou seja, iterar sobre cada uma das suas células.

Neste módulo, e de acordo com a literatura apresentada, espera-se que os alunos possam trazer o conceito de variável pouco sólido, o que lhes vai dificultar a compreensão da forma de aceder às variáveis individuais guardadas nas células dos vetores.

3.7.1. PLANIFICAÇÃO DO MÓDULO #4: ESTRUTURAS DE DADOS ESTÁTICAS (16 HORAS / 32)

Os professores de cursos profissionais devem seguir os programas e currículos e encontrar as formas de propor esses conteúdos aos alunos para que estes os aprendam. Ao fazê-lo, devem considerar as condicionantes internas, como as características da turma; mas também as externas, como a organização da escola, o horário da turma ou o número de horas de que se dispõe para cada módulo (Hazzan, Lapidot, & Ragonis, 2014). Mas não se deve esquecer que quando forem aplicar a sua planificação terão que fazer os ajustes necessários, considerando a evolução da turma.

Como foi dito, serão seguidas as várias pistas pedagógicas já apresentadas (§ 3.5.4) na construção da proposta para o planeamento da intervenção. Será também seguido o cenário de aprendizagem desenvolvido (*vide* Figura 9.2 no anexo A) que indica um conjunto de atividades desligadas e em computador que se deverão desenvolver.

A intervenção utilizará 16 das 32 horas previstas no módulo. As restantes serão lecionadas no próximo ano letivo para terminar o módulo começado. Assim sendo, opta-se por abordar todos os conceitos previstos, exceto os vetores bidimensionais que ficam para mais tarde. Isto porque, esse tópico corresponde a uma maior carga cognitiva e não havia horas suficientes neste período. Acresce que no arranque da 2.ª parte do módulo será necessário fazer revisões daquilo que aprenderão este ano, antes de se poder avançar para aquele tema.

Assume-se que os alunos já concluíram com sucesso os módulos anteriores, facto a verificar no início da intervenção e que poderá requerer ajustes do planeamento.

---

[1] Note-se que a palavra inglesa '*array*' pode ser traduzida para 'vetor' ou 'lista'. Neste relatório, usaremos a palavra 'vetor' para nos referirmos aos *arrays* estáticos. Reservamos a palavra 'lista' para nos referirmos às classes implementadas com memória dinâmica através de ponteiros (módulos seguintes) e para as implementações próprias da linguagem.

Ano: 10.º ano do Ensino Secundário Profissional **Grupo de Docência:** Informática [550]	**Data:** 12 de maio a 09 de junho de 2020 **Horário:** 11 blocos de 100 minutos

Objetivos Gerais de Aprendizagem do Módulo 4 – Estruturas de Dados Estáticas

- ☐ Definição de *String* como variável capaz de guardar um número finito de valores do tipo CHAR
- ☐ Declaração e Manipulação de variáveis do tipo *String*
- ☐ Definição de vetor como variável capaz de "agregar" um número finito de valores do mesmo tipo
- ☐ Declaração e manipulação de variáveis do tipo vetor
- ☐ Estudo de algoritmos de manipulação de vetor
- ☐ Iniciação de vetores
- ☐ Pesquisa sequencial em vetores e cadeias de caracteres
- ☐ Inserção e remoção de elementos de um vetor: no Inicio (à cabeça) ; no Fim (à cauda).
- ☐ Ordenação crescente ou decrescente dos elementos de um vetor
- ☐ Inserção e remoção de elementos em vetor ordenados

Objetivos Específicos	Conteúdos Curriculares	Atividades a desenvolver	Metodologias e Estratégias	Recursos Técnicos e Pedagógicos	Critérios e Instrumentos de Avaliação

⊕ Conhecer as *Strings*, sua declaração e manipulação
⊕ Conhecer a necessidade de vetores
⊕ Conhecer os vetores, sua declaração e manipulação
⊕ Distinguir situações em que se deve usar vetores (ou não)
⊕ Desenvolver as capacidades de resolução algorítmica de problemas
	`String`, `char[]`
	Vetores: `int[]`
	`float[]`, `double[]`
	`String[]`
	Estrutura de um vetor: coleção ordenada de células do mesmo tipo com um nome partilhado e um índice único para cada elemento
	Varrimento de vetor
	Pesquisa em vetor
	Ordenação de vetor
	Big O: Notação assimptótica
		Atividades desligadas:
		pesquisa linear
		➥ gavetas
		pesquisa binária
		➥ dicionário
		ordenação
		➥ gavetas
		filas e pilhas
		cifras de César
		Leitura de código
		Representação esquemática da memória durante execução de algoritmo
		Escrita de código
		Depuração de código
		Escape Room Criptográfico
			Expositivo: exposição dialogada dos problemas.
			Demonstrativo: demonstração de procedimentos
			Ativo: levar os alunos a procurar as soluções dos problemas propostos
			Interrogativo: questionamento para apoio à resolução de problemas e *debug*. *(Rubber Duck method)*
				✓ Computadores
				✓ Videoprojector
				✓ Quadro de Giz
				✓ Formulários de Avaliação em `Google Forms`.
				✓ *Software* apropriado: Visual Studio 2012
				✓ csfieldguide.org.nz/en/interactives
				✓ www.sorting-algorithms.com
				✓ programmingforthepuzzled.github.io/puzzled
					Avaliação Formativa: Observação direta; Grelha de Observação,
					Exit Ticket: pequeno questionário sobre a aula e sobre os conteúdos abordados
					Escape Room

Padrões de desempenho:

- Conhece as razões que tornam os vetores necessários
- Sabe aceder a cada célula de um vetor. Distingue a posição do valor: `a1` =/= `a[1]`. Distingue `i` de `a[i]`
- Representa esquematicamente um vetor em memória
- Conhece os vários métodos de pesquisa de valores em vetores (linear, binária, aleatória)
- Programa os algoritmos de pesquisa abordados
- Conhece os vários métodos de ordenação de valores armazenados em vetores
- Programa os algoritmos de ordenação abordados (*bubble sort*, *insertion sort*, *selection sort* e *quick sort*)
- Intui a necessidade de estruturas mais robustas: ponteiros, registos, objetos

Articulações Curriculares:

- As instruções e os exemplos dos problemas podem ser escritas na língua estrangeira (no caso, Inglês);
- As palavras cifradas podem estar relacionadas com os conteúdos de outras disciplinas;
- Aplicação das técnicas aprendidas no apoio à resolução de Jogos Matemáticos;
- Implementação com vetores das fórmulas de coluna aprendidas no módulo de Folhas de Cálculo de TIC.

Tabela 3.3: Planificação da temática do módulo 4 da disciplina de PSI a ser lecionado durante a intervenção

3.7.1. CALENDARIZAÇÃO DO MÓDULO #4: ESTRUTURAS DE DADOS ESTÁTICAS (16 HORAS / 32)

Dada o calendário escolar para este ano letivo (*vide* Figura 3.15), as ausências planeadas do professor cooperante, o número de horas indicadas para o módulo, e a planificação da Tabela 3.3, chegou-se à planificação que se apresenta na Tabela 3.4

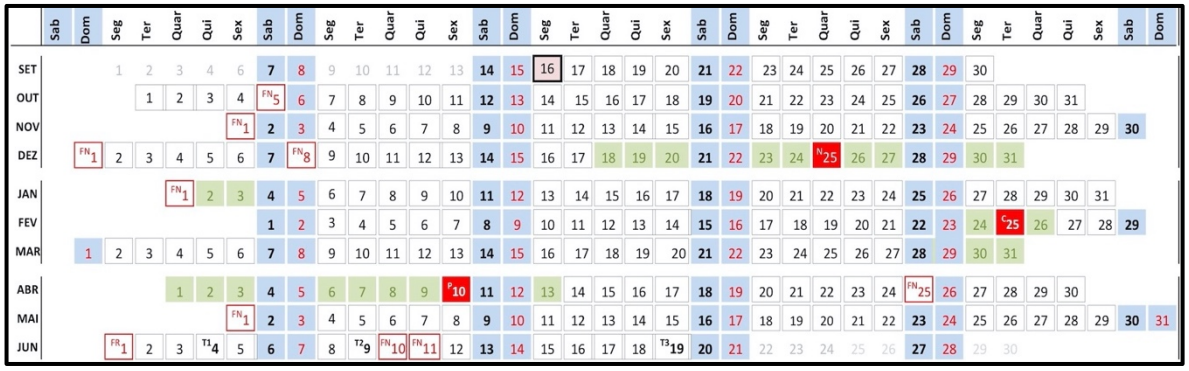


Figura 3.15:Calendário do ano letivo 2019/20.

O módulo é o último a lecionar este ano pelo que ocupará as últimas aulas do 3.º período. As duas primeiras semanas terão o dobro das horas às terças, uma vez que o professor cooperante estará ausente a seguir à Páscoa e na semana anterior à intervenção.

Nas restantes semanas, com a devida aprovação do conselho de turma, as aulas de quinta serão lecionadas às sextas, aproveitando a manhã livre dos alunos e resolvendo o conflito com as aulas na escola onde leciono (às quais faltarei nas duas primeiras semanas).

	Data	Lição	Atividades a Desenvolver	Foco
3.º Período	12-05-2020	M4.01 M4.02	Avaliação diagnóstica – compreensão de procura e ordenação (Bebras, nível 1); Atividades desligadas de procura e ordenação – escrita dos algoritmos em português; Atividades desligadas de criptografia simples	
		M4.03 M4.04	– cifras de César, ROT13, de transposição, PigPen, Vigenère, e grelhas; Conversão de binário para `ASCII` (revisão de Arquitetura de Computadores, Módulo 1).	Unplugged
	14-05-2020	M4.05 M4.06	Leitura de (bom) código de procura e ordenação; Identificação de algoritmos.	Read
	19-05-2020	M4.07 M4.08	Leitura de (bom) código; Representação do estado da memória ao longo da execução dos programas:	
		M4.09 M4.10	declaração de vetores; procura de valores em vetores; ordenação de vetores. Análise Crítica de Complexidade e Eficiência.	Draw
	21-05-2020	M4.11 M4.12	Escrita de código que implemente os métodos criptográficos abordados em M4.03 e 04; Análise Crítica de Complexidade e Eficiência.	Write
	26-05-2020	M4.13 M4.14	Depuração de Código (próprio e de colegas); Resolução de erros comuns – dificuldades mais frequentes.	Debug
	28-05-2020	M4.15 M4.16	`String` como cadeia de caracteres (`char[]`) e o caracter `NULL` (`\0`); Representações alternativas de `String`: tamanho da cadeia de caracteres.	String
	02-06-2020	M4.17 M4.18	Cadeias de outros tipos de dados (float, long, double); Representação em memória (8, 16, 32 e 64 bits).	Bits & Bytes
	04-06-2020	M4.19 M4.20	Vetores alinhadas: representação de informação Vetores de `Strings`, (paralelismo com colunas de Bases de Dados, intuição de estruturas).	String[]
	09-06-2020	M4.21 M4.22	*Escape Room* Criptográfico Problema de aplicação do conhecimento adquirido; Autoavaliação; Avaliação da intervenção.	Assessment

Tabela 3.4: Calendarização das aulas de PSI a serem lecionadas durante a intervenção

4. INTERVENÇÃO PEDAGÓGICA

A intervenção pedagógica é planeada considerando tudo o que foi dito até agora: o programa da disciplina onde decorrerá a intervenção, as dificuldades típicas dos alunos nesses módulos e as pistas didáticas, pedagógicas e científicas que se apresentaram. Considera também as observações feitas diretamente na escola, com a turma que terá as aulas. Tentar-se-á propor atividades diferente das habituais no ambiente daquela escola, aliás como é desejado que aconteça nesta fase da formação inicial de professores em que me encontro (Arends, 2014).

4.1. OBSERVAÇÃO DE AULAS

Para poder adequar as aulas que iria lecionar a estes alunos, fiz a observação de várias aulas registando comportamentos, atitudes, métodos e resultados (Reis, 2011).

A descrição da turma (§2.1.3) e das aulas da disciplina (§ 2.2) foram feitas considerando aquilo que se observou, juntamente com os documentos oficiais (notas e conselhos de turma).

Na primeira aula observada, já no fim do primeiro módulo, o professor cooperante acolheu a visita e pediu-me para apoiar a aula e fiquei responsável por ajudar um conjunto de alunos. Esta opção dificultou uma análise mais sistemática e o seu registo extensivo, mas permitiu que os alunos que enquadrassem rapidamente e me tenham começado a tratar por "professora" e me tenham aceitado com a mesma autoridade que concediam ao professor cooperante.

Na aula seguinte, sendo a primeira do segundo módulo, houve mais momentos expositivos, guiados de forma explícita pelo professor, como a preparação do software para ser utilizado pelos alunos e a escrita dos primeiros programas com a nova linguagem de programação. Dado que esta aula foi bastante expositiva fiquei na parte de trás da sala a fazer uma observação mais formal e a trabalhar como aluna, seguindo as instruções e tentando perceber as dificuldades dos alunos.

Ficou claro que na minha intervenção terei de reduzir ao mínimo o método expositivo, uma vez que os alunos ouvem música durante a aula e a exposição os obriga a parar, mas também porque os alunos têm pouca paciência para esse método, dispersando-se rapidamente. Pelo contrário quando há exercícios e problemas para resolver, são pró-ativos e empenhados.

Nas aulas observadas o professor aproveita o intervalo para descansar um pouco e ir à cantina comer ou à sala de professores. Os alunos também fazem questão de sair durante o intervalo e, inclusivamente, regressam um pouco atrasados. Isto dificulta o aproveitamento das aulas em bloco para fazer atividades um pouco mais extensas, mas, ainda assim, na primeira aula que terá 4 tempos, 200 minutos, combinarei com os alunos que só terão um intervalo no meio de cada bloco, mas que, em compensação, poderão sair um pouco mais cedo, sendo o último tempo do dia.

4.2. INTERVENÇÃO PLANEADA

Nesta secção apresentam-se os planos de aulas. Apresentam-se em blocos de 100 minutos, uma vez que nessa altura haverá intervalo (e será preciso retomar temas) mas também por uma questão de flexibilidade, para o caso de ser necessário trocar alguma aula com outro professor ou fazer algum ajuste de calendário ou horário.

Como veremos no capítulo 6, dedicado à componente investigativa da intervenção, será necessário recolher evidências sistemáticas da progressão dos alunos em termos de compreensão dos conceitos para ter uma indicação do impacto que cada um dos métodos escolhidos tem na aprendizagem dos alunos. Por isso, além dos ***Exit Tickets*** com que tenho terminado as aulas (Pardal, 2019), teremos alguns ***Admission Tickets*** no início das aulas com perguntas muito curtas que registem aquilo de que os alunos ainda se lembram das aulas anteriores.

Uma vez que não é possível fazer avaliação com base na escrita de código no início da avaliação, e porque a escrita de código ainda mascara a real compreensão dos temas por parte dos alunos, vamos utilizar como métrica o pensamento computacional e como base dos exercícios utilizaremos os cartões do concurso do Bebras (Weigend, Vaníček, Pluhár, & Pesek, 2019) que testam os mesmos conceitos a vários níveis, tendo exercícios para as idades do 1.º, 2.º e 3.º ciclos (*vide* Figura 4.1).

Figura 4.1: Coleção de cartões Bebras com problemas de Pensamento Computacional dirigidos a didades 7+, 10+ e 13+

4.2.1. PLANO DAS AULAS #1 E 2: ATIVIDADES DESLIGADAS 1

Escola Secundária de São João da Talha			Professor Cooperante: José António Cruz	
Professora: Joana Paulo Pardal	Data: 12/maio, terça		Hora: 14:15 – 16:10	Duração: 2 x 50 = 100 min
Turma: 10.º E	Sala: F-48	Programação e Sistemas de Informação	Módulo: 4	Aulas: 1 e 2
Sumário: Atividades Desligadas – intuição de estruturas de dados, procura e ordenação				
Recursos: ✓ Formulário de Avaliação Diagnóstica Formativa ✓ Computer Science Field Guide, Searching Algorithms: csfieldguide.org.nz/en/interactives/searching-algorithms ✓ Demonstração de algoritmos de ordenação: www.sorting-algorithms.com ✓ Caixas fechadas, armários com portas, armários com gavetas, balança de braços				

Conteúdos	Objetivos	Atividades	Metodologias e Estratégias	Avaliação	Tempos (mins)
Programa da Disciplina e do Módulo 4	Objetivos e Temas a desenvolver no módulo	☆ Apresentação dos professores ☆ Apresentação do módulo	Expositivo Interrogativo		10
Listas Necessidade de listas na vida real	Avaliar o conhecimento inicial	☆ *Admission Ticket*: compreensão de procura e ordenação no dia-a-dia	Interrogativo	Diagnóstica (Bebras, nível 1);	30
Procura Ordenação Complexidade Escrita de Algoritmos Leitura de Algoritmos Metáforas e Analogias de Algoritmos (Forišek & Steinová, 2012)	Distinguir situações em que se deve usar vetores (ou não)	☆ Atividades desligadas de procura e ordenação: - procura de números de acordo com diferentes critérios - procura de palavras em dicionário - ordenação de pessoas, de livros e cromos ☆ Escrita dos algoritmos em português ☆ Execução das instruções escritas pelos colegas	Expositivo Demonstrativo Ativo	Formativa Grelha de Observação	50
	Avaliar o conhecimento adquirido	☆ *Exit Ticket*: Pesquisa e Ordenação Bebras, nível 2	Interrogativo	Formativa	10

Tabela 4.1: Plano da Aula #1 e 2: Atividades Desligadas – intuição de estruturas de dados, procura e ordenação

Nota: esta é a primeira parte de uma aula de 200 minutos.

4.2.2. PLANO DAS AULAS #3 E 4: ATIVIDADES DESLIGADAS 2

Escola Secundária de São João da Talha			Professor Cooperante: José António Cruz	
Professora: Joana Paulo Pardal	Data: 12/maio, terça		Hora: 16:15 – 18:10	Duração: 2 x 50 = 100 min
Turma: 10.º E	Sala: F-48	Programação e Sistemas de Informação	Módulo: 4	Aulas: 1, 2, 3 e 4
Sumário: Atividades Desligadas – criptografia, binário e ASCII				
Recursos: ✓ Formulário de Avaliação Diagnóstica Formativa ✓ Telegrafo binário ✓ Descodificadores rotativos ✓ Alfabetos e tabelas de símbolos				

Conteúdos	Objetivos	Atividades	Metodologias e Estratégias	Avaliação	Tempos (mins)
Procura Ordenação Representação de dados em memória Sistema de Numeração Binária, Decimal, Octal e Hexadecimal	Avaliar o conhecimento inicial	☆ *Admission Ticket*: revisão de Arquitetura de Computadores, M1	Interrogativo	Diagnóstica	10
	Compreender a representação em memória da `String`	☆ Conversão de binário para ASCII	Expositivo Demonstrativo Ativo	Formativa Grelha de Observação	20
		☆ Conversão de binário para ISO Latin 1 e UTF-8			20
Métodos de ocultação de informação: - tinta invisível - símbolos - rotação de alfabeto - números	Compreender diferentes formas de representar informação e de a ocultar	☆ Atividades desligadas de criptografia simples – cifras de César, – ROT13, – transposição, – PigPen, – Vigenère, – grelhas.			40
Sistema de Numeração Binária Criptografia	Avaliar o conhecimento adquirido	☆ *Exit Ticket*: Criptografia Bebras, nível 2	Interrogativo	Formativa	10

Tabela 4.2: Plano da Aula #3 e 4: Atividades Desligadas – criptografia, binário e ASCII

4.2.3. PLANO DAS AULAS #5 E 6: LEITURA DE (BOM) CÓDIGO

Escola Secundária de São João da Talha			Professor Cooperante: José António Cruz	
Professora: Joana Paulo Pardal	Data: 14/maio, quinta		Hora: 09:15 – 11:10	Duração: 2 x 50 = 100 min
Turma: 10.º E	Sala: F-48	Programação e Sistemas de Informação	Módulo: 4	Aulas: 5 e 6
Sumário: Leitura de (Bom) Código; Reconhecimento de Algoritmos com Vetores				
Recursos: ✓ Formulário de Avaliação Diagnóstica Formativa ✓ Exemplos de (bom) código de procura linear e binária ✓ Exemplos de (bom) código de ordenação				

Conteúdos	Objetivos	Atividades	Metodologias e Estratégias	Avaliação	Tempos (mins)
Listas Procura Ordenação	Avaliar o conhecimento inicial	☆ *Admission Ticket*: Pesquisa e Ordenação Bebras, nível 3	Interrogativo	Diagnóstica	10
Necessidade de listas na programação Vetores: `int[]`	Desenvolver capacidades de resolução de problemas	☆ Leitura de (bom) código de procura	Expositivo Demonstrativo Ativo	Formativa Grelha de Observação	30
		☆ Leitura de (bom) código de ordenação			30
		☆ Identificação de algoritmos conhecidos			20
Varrimento de vetor Pesquisa em vetor Ordenação de vetor	Avaliar o conhecimento adquirido	☆ *Exit Ticket*: Identificação de objetivos de pequenas funções implementadas	Interrogativo	Formativa	10

Tabela 4.3: Plano da Aula #5 e 6: Leitura de (Bom) Código; Reconhecimento de Algoritmos com Vetores

4.2.4. PLANO DAS AULAS #7 E 8: REPRESENTAÇÃO DE *STRINGS* EM MEMÓRIA

Escola Secundária de São João da Talha			Professor Cooperante: José António Cruz	
Professora: Joana Paulo Pardal	Data: 19/maio, terça		Hora: 14:15 – 16:10	Duração: 2 x 50 = 100 min
Turma: 10.º E	Sala: F-48	Programação e Sistemas de Informação	Módulo: 4	Aulas: 7 e 8
Sumário: Representação de Strings em memória ao longo da execução de algoritmos				
Recursos: ✓ Formulário de Avaliação Diagnóstica Formativa ✓ Exemplos de (bom) código que utilize Strings				

Conteúdos	Objetivos	Atividades	Metodologias e Estratégias	Avaliação	Tempos (mins)
Listas Procura e Ordenação Complexidade	Avaliar o conhecimento inicial	☆ *Admission Ticket*: Criptografia Bebras, nível 3	Interrogativo	Diagnóstica	10
`String`, `char[]` Acesso a letras individuais em `String` Representação de `String` em memória	Compreender a representação de dados em memória	☆ Leitura de (bom) código	Expositivo Demonstrativo Ativo	Formativa Grelha de Observação	20
		☆ Representação esquemática do estado da memória ao longo da execução dos programas			30
	Comparar diferentes métodos	☆ Análise Crítica de Complexidade e Eficiência			40

Tabela 4.4: Plano da Aula #7 e 8: Representação de variáveis em memória ao longo da execução de algoritmos

Nota: esta é a primeira parte de uma aula de 200 minutos.

4.2.5. PLANO DAS AULAS #9 E 10: REPRESENTAÇÃO DE VETORES EM MEMÓRIA

Escola Secundária de São João da Talha			Professor Cooperante: José António Cruz	
Professora: Joana Paulo Pardal		Data: 19/maio, terça	Hora: 16:15 – 18:10	Duração: 2 x 50 = 100 min
Turma: 10.º E	Sala: F-48	Programação e Sistemas de Informação	Módulo: 4	Aulas: 9 e 10
Sumário: Representação de vetores de inteiros em memória				
Recursos: ✓ Formulário de Avaliação Diagnóstica Formativa ✓ Exemplos de (bom) código que utilize vetores				

Conteúdos	Objetivos	Atividades	Metodologias e Estratégias	Avaliação	Tempos (mins)
Vetores: `int[]` Estrutura de um vetor: coleção ordenada de células do mesmo tipo com um nome partilhado e um índice único para cada elemento Varrimento de vetor Pesquisa em vetor Ordenação de vetor Big O: Notação assimptótica Posição e valor: `a1` vs `a[1]` `i` de `a[i]`	Analisar criticamente as diferentes opções de implementação	☆ Leitura de (bom) código	Expositivo Demonstrativo Ativo	Formativa Grelha de Observação	20
		☆ Representação esquemática do estado da memória ao longo da execução dos programas			30
		☆ Análise Crítica de Complexidade e Eficiência			40
	Avaliar o conhecimento adquirido	☆ *Exit Ticket*: representação esquemática de *Strings* e vetores em memória	Interrogativo	Formativa	10

Tabela 4.5: Plano da Aula #9 e 10: Representação de vetores em memória

4.2.6. PLANO DAS AULAS #11 E 12: ESCRITA DE CÓDIGO

Escola Secundária de São João da Talha			Professor Cooperante: José António Cruz	
Professora: Joana Paulo Pardal	Data: 21/maio, quinta		Hora: 09:15 – 11:10	Duração: 2 x 50 = 100 min
Turma: 10.º E	Sala: F-48	Programação e Sistemas de Informação	Módulo: 4	Aulas: 11 e 12
Sumário: Escrita de Código e sua Análise Crítica (Complexidade e Eficiência)				
Recursos: ✓ Formulário de Avaliação Diagnóstica Formativa ✓ Software apropriado: Visual Studio 2012 Express				

Conteúdos	Objetivos	Atividades	Metodologias e Estratégias	Avaliação	Tempos (mins))
Varrimento de vetor Métodos de pesquisa de valores em vetores (linear, binária, aleatória) Algoritmos de ordenação: *bubble sort*, *insertion sort*, *selection sort* e *quick sort* *Big O*: Notação assimptótica	Avaliar o conhecimento inicial	☆ *Admission Ticket*: Pesquisa e Ordenação Bebras, nível 4	Interrogativo	Diagnóstica	10
	Conhecer as *Strings*, e os vetores sua declaração e manipulação	☆ Escrita de código que implemente métodos criptográficos	Expositivo Demonstrativo Ativo	Formativa Grelha de Observação	30
		☆ Escrita de código que implemente pesquisas simples			20
		☆ Escrita de código que implemente ordenações simples			30
	Avaliar o conhecimento adquirido	☆ *Exit Ticket*: reconhecimento de diferentes algoritmos	Interrogativo	Formativa	10

Tabela 4.6: Plano da Aula #11 e 12: Escrita de Código e sua Análise Crítica (Complexidade e Eficiência)

4.2.7. PLANO DAS AULAS #13 E 14: DEPURAÇÃO DE CÓDIGO

Escola Secundária de São João da Talha			Professor Cooperante: José António Cruz	
Professora: Joana Paulo Pardal		Data: 26/maio, terça	Hora: 16:15 – 18:10	Duração: 2 x 50 = 100 min
Turma: 10.º E	Sala: F-48	Programação e Sistemas de Informação	Módulo: 4	Aulas: 13 e 14
Sumário: Depuração de Código (de colegas e de erros comuns)				
Recursos: ✓ Formulário de Avaliação Diagnóstica Formativa ✓ Software apropriado: Visual Studio 2012 Express				

Conteúdos	Objetivos	Atividades	Metodologias e Estratégias	Avaliação	Tempos (mins))
Implementações de pesquisa e ordenação	Avaliar o conhecimento inicial	☆ *Admission Ticket*: Pesquisa e Ordenação Bebras, nível 4	Interrogativo	Diagnóstica	10
Métodos de análise crítica de erros Mecanismos de *debug* interativo Inspeção de memória durante execução no editor *Rubber duck debugging*	Distinguir a posição do valor: `a1` vs `a[1]` Distinguir `i` de `a[i]`	☆ Depuração de código próprio	Expositivo Demonstrativo Ativo	Formativa Grelha de Observação	20
		☆ Depuração de Código de colegas			20
		☆ Depuração de Código com erros mais comuns			40
	Avaliar o conhecimento adquirido	☆ *Exit Ticket*: correções de algoritmos de acesso a posições de vetores de acordo com os erros mais comuns	Interrogativo	Formativa	10

Tabela 4.7: Plano da Aula #13 e 14: Depuração de Código (de colegas e de erros comuns)

4.2.8. PLANO DAS AULAS #15 E 16: STRING COMO CADEIA DE CARACTERES

Escola Secundária de São João da Talha			Professor Cooperante: José António Cruz	
Professora: Joana Paulo Pardal	Data: 28/maio, quinta		Hora: 09:15 – 11:10	Duração: 2 x 50 = 100 min
Turma: 10.º E	Sala: F-48	Programação e Sistemas de Informação	Módulo: 4	Aulas: 15 e 16
Sumário: String como cadeia de caracteres (char[]) e o caracter NULL				
Recursos: ✓ Formulário de Avaliação Diagnóstica Formativa ✓ Software apropriado: Visual Studio 2012 Express				

Conteúdos	Objetivos	Atividades	Metodologias e Estratégias	Avaliação	Tempos (mins)
Criptografia	Avaliar o conhecimento inicial	☆ *Admission Ticket*: Criptografia Bebras, nível 4	Interrogativo	Diagnóstica	10
`String` como cadeia de caracteres: `char[]` Caracter `NULL`: `\0` Tamanho da cadeia de caracteres Representações alternativas de `String`	Implementar algoritmos de manipulação de strings	☆ Escrita de código que implemente métodos criptográficos	Expositivo Demonstrativo	Formativa Grelha de Observação	40
		☆ Escrita de código que implemente pesquisas simples	Ativo		40
	Avaliar o conhecimento adquirido	☆ *Exit Ticket*: codificação de algoritmo simples de acesso a posições de Strings	Interrogativo	Formativa	10

Tabela 4.8: Plano da Aula #15 e 16: String como cadeia de caracteres (char[]) e o caracter NULL

4.2.9. PLANO DAS AULAS #17 E 18: VETORES DE OUTROS TIPOS DE DADOS

Escola Secundária de São João da Talha			Professor Cooperante: José António Cruz	
Professora: Joana Paulo Pardal	Data: 02/junho, terça		Hora: 16:15 – 18:10	Duração: 2 x 50 = 100 min
Turma: 10.º E	Sala: F-48	Programação e Sistemas de Informação	Módulo: 4	Aulas: 17 e 18
Sumário: Vetores de outros tipos de dados (float, long, double)				
Recursos: ✓ Formulário de Avaliação Diagnóstica Formativa ✓ Software apropriado: Visual Studio 2012 Express				

Conteúdos	Objetivos	Atividades	Metodologias e Estratégias	Avaliação	Tempos (mins)
Algoritmos de pesquisa e ordenação	Avaliar o conhecimento inicial	☆ *Admission Ticket*: Pesquisa e Ordenação Bebras, nível 4	Interrogativo	Diagnóstica	10
Cadeias de outros tipos de dados `float[]`, `long[]`, `double[]`	Codificar algoritmos de manipulação de vetores de vários tipos	☆ Escrita de código que implemente pesquisas simples	Expositivo Demonstrativo	Formativa Grelha de Observação	40
		☆ Escrita de código que implemente ordenações simples	Ativo		40
Representação em memória (8, 16, 32 e 64 bits)	Avaliar o conhecimento adquirido	☆ *Exit Ticket*: codificação de algoritmo simples de acesso a posições de vetores	Interrogativo	Formativa	10

Tabela 4.9: Plano da Aula #17 e 18: Vetores de outros tipos de dados (float, long, double)

4.2.10. PLANO DAS AULAS #19 E 20: VETORES DE STRINGS

Escola Secundária de São João da Talha			Professor Cooperante: José António Cruz	
Professora: Joana Paulo Pardal	Data: 04/junho, quinta	Hora: 09:15 – 11:10	Duração: 2 x 50 = 100 min	
Turma: 10.º E	Sala: F-48	Programação e Sistemas de Informação	Módulo: 4	Aulas: 19 e 20
Sumário: Vetores de Strings e vetores alinhados				
Recursos: ✓ Formulário de Avaliação Diagnóstica Formativa ✓ Software apropriado: Visual Studio 2012 Express				

Conteúdos	Objetivos	Atividades	Metodologias e Estratégias	Avaliação	Tempos (mins)
Listas de Strings Listas de palavras	Avaliar o conhecimento inicial	☆ *Admission Ticket*: Alinhamento de Listas aparentemente independentes	Interrogativo	Diagnóstica	10
Vetores alinhadas: representação de informação	Manipular vetores de forma sincronizada	☆ Escrita de código que aceda a vetores alinhados: com o mesmo tamanho e onde a mesma posição se refere a um mesmo 'objeto'	Expositivo Demonstrativo Ativo	Formativa Grelha de Observação	30
Paralelismo com colunas de Bases de Dados	Distinguir tipos de dados em vetores				30
Intuição de Estruturas de Dados (e objetos)	Intuir a necessidade de representacões melhores				20
Conteúdo teórico do módulo	Avaliar o conhecimento adquirido	☆ *Exit Ticket*: codificação de algoritmo simples de acesso sincronizado a posições de vetores de *Strings*	Interrogativo	Formativa Sumativa	10

Tabela 4.10: Plano da Aula #19 e 20: Vetores de Strings e vetores alinhados

4.2.11. PLANO DAS AULAS #21 E 22: *ESCAPE ROOM* CRIPTOGRÁFICO

Escola Secundária de São João da Talha			Professor Cooperante: José António Cruz	
Professora: Joana Paulo Pardal	Data: 09/junho, terça		Hora: 16:15 – 18:10	Duração: 2 x 50 = 100 min
Turma: 10.º E	Sala: F-48	Programação e Sistemas de Informação	Módulo: 4	Aulas: 21 e 22
Sumário: *Escape Room* Criptográfico				
Recursos: ✓ Formulário de Avaliação Diagnóstica Formativa ✓ Software apropriado: Visual Studio 2012 Express ✓ Garra metálica de 6 cadeados; cadeados de vários tipos (chave, 3-digitos, 4-digitos, datas, letras, símbolos) ✓ Canetas de tinta invisível com lâmpada de luz ultravioleta				

Conteúdos	Objetivos	Atividades	Metodologias e Estratégias	Avaliação	Tempos (mins)
Conceito de *Escape Room* Regra do jogo	Clarificar as regras específicas da aula	Apresentação da última aula e do desafio	Expositivo		10
Síntese da matéria dada neste módulo	Avaliar o conhe-cimento adquirido	*Escape Room* Criptográfico conjunto de atividades de resolução de problemas e pensamento crítico	Ativo Interrogativo	Formativa Grelha de Observação Sumativa	60
		Exit Ticket: criptografia, algoritmia	Interrogativo	Sumativa	10
	Avaliar-se a si próprio	*Exit Ticket:* expectativas, atitudes e valores	Interrogativo	Autoavaliação	10
	Avaliar métodos e estratégias utilizadas	*Exit Ticket*: qualidade das aulas desenvolvidas	Interrogativo	Avaliação da intervenção	10

Tabela 4.11: Plano da Aula #21 e 22: *Escape Room* Criptográficozx

5. AVALIAÇÃO DA INTERVENÇÃO

A avaliação da intervenção divide-se em várias partes, sendo feita de vários pontos de vista.

Primeiro que tudo, avalia o cumprimento do objetivo principal: **os alunos** aprenderam? Atingiram todos os objetivos de aprendizagem estabelecidos? Quais as dificuldades que tiveram? Foram detetadas pela avaliação sumativa? Foram colmatadas de forma adequada? A tempo?

Avalia também a qualidade percecionada **das aulas lecionadas**: os alunos gostaram dos métodos utilizados? Acham que foram úteis para a sua aprendizagem? Ou foram distrativos? Foram tidos como inovadores? De que é que não gostaram? O que é que mudavam? Qual a opinião do professor cooperante?

Finalmente, e no espírito da melhoria contínua do ensino e do docente, há as questões relativas à **dimensão investigativa**: qual o impacto das estratégias e metodologias utilizadas no desenvolvimento do pensamento computacional dos alunos? Qual o impacto das atividades desligadas na compreensão conceptual dos algoritmos de pesquisa e ordenação de vetores? Qual o impacto da leitura de (bom) código na implementação dos algoritmos relacionados? Como é que a representação pictórica do funcionamento de um programa em memória melhora a compreensão dos algoritmos?

Dado o caráter prático da disciplina, a avaliação dos alunos neste módulo é feita, sobretudo, com base naquilo que os alunos forem capazes de fazer durante as aulas. A primeira ferramenta de registo deste progresso é o ***Passaporte de Aprendizagem*** que deverá estar todo preenchido, mostrando como os alunos passaram por todas as etapas previstas. Não haverá lugar a prova escrita, mas o ***Escape Room Criptográfico*** final terá várias componentes teóricas para que se possa aferir não só a capacidade de resolver problemas práticos, mas também a compreensão teórica dos conceitos, o raciocínio crítico sobre os mesmos, e a capacidade de analisar o código desenvolvido.

A participação e a atitude dos alunos são registadas durante o seu desenvolvimento, enquanto aplicam os conceitos aprendidos (Prifti, Levkovskyi, Knigge, & Krcmar, 2018) seguindo, para isso, a grelha de avaliação da própria escola.

Estes instrumentos terão uma componente formativa na medida em que serão devolvidos aos alunos corrigidos com *feedback* e indicações de melhoria, tanto os que forem feitos em papel, como os que forem feitos com recurso a plataformas digitais. Terão também uma componente diagnóstica porque serão utilizados para (re)avaliar a planificação que agora se apresenta, adequando-a às aprendizagens feitas e às dificuldades encontradas. Serão ainda utilizados na componente investigativa para, utilizando uma escala de desempenho, nos darem uma medida do progresso de cada aluno.

5.1. AVALIAÇÃO DIAGNÓSTICA

A avaliação diagnóstica permite aos docentes conhecerem as suas turmas, os seus alunos, as suas características, o patamar de conhecimentos em que cada um se encontra e as dificuldades expectáveis. Com base nesta avaliação é possível planear o processo de ensino-aprendizagem de forma mais adequada aos alunos, definindo estratégias que possam ajudar a superar as dificuldades identificadas e ajustando o ritmo às capacidades encontradas.

No agrupamento de escolas onde decorrerá a intervenção é obrigatória a realização deste tipo de avaliação para determinar o ponto de partida da turma, identificar dificuldades, ajustar planificações e formular estratégias que superem as dificuldades encontradas. Tipicamente, o diretor de turma aplica um teste genérico que capture o estado geral dos alunos e depois, na medida das necessidades, cada professor faz a avaliação diagnóstica complementar que for necessária.

Uma vez que os módulos anteriores ao da intervenção terão avaliação sumativa final, usaremos esses instrumentos como diagnóstico para informar o módulo em que iremos intervir. Desta forma não sobrecarregaremos os alunos com demasiadas avaliações, nem os docentes com demasiadas correções.

Também no final de cada aula faremos uma pequena avaliação formativa, um ***Exit Ticket***, que servirá de avaliação diagnóstica para a aula seguinte, informando sobre os erros dos alunos e as dificuldades encontradas. De acordo com esta informação far-se-á a adequação das aulas seguintes, esclarecendo equívocos com exemplos elucidativos, escolhendo exemplos que permitam exercitar os pontos de esforço, e ajustando o ritmo à capacidade existente de facto. Sem baixar a "fasquia" nem desistir de nenhum objetivo de aprendizagem, mas procurando novos caminhos para as alcançar. Haverá também lugar à diferenciação pedagógica individual que for considerada necessária, sobretudo dos três alunos com necessidades educativas especiais.

Uma vez que a avaliação feita apenas com provas escritas é pouco significativa e não é suficiente para avaliar o desenvolvimento da compreensão dos alunos dos temas abordados, utilizar-se-ão fichas de trabalho (uma por aula) e mini projetos. A avaliação destes, terá uma componente relativa à atitude e empenho dos alunos.

A utilização do pensamento computacional (Oliveira, 2017), sobretudo do desenho de algoritmos, pretende baixar a carga cognitiva ao manter acessível a linguagem de escrita das perguntas, mas sobretudo de resposta às perguntas. Esta componente será avaliada no início de várias aulas utilizando um ***Admission Ticket***.

5.2. AVALIAÇÃO FORMATIVA

O agrupamento de escolas determinou nos critérios de avaliação gerais que a avaliação formativa deve ser a principal modalidade de avaliação de forma a regular o ensino e as aprendizagens, permitindo a recolha de informação que mostre como se ensina e como se aprende, justificando medidas e estratégias pedagógicas.

Pede-se que este tipo de avaliação seja feito de forma contínua e sistemática, de formas diversificadas que permitam uma maior abrangência da informação recolhida.

A capacidade de melhoria dos alunos está, de facto, ligada com esta possibilidade de receberem *feedback* contínuo (Yan, Hu, & Piech, 2019) isto sem terem (ainda) a pressão de uma classificação numérica quando ainda estão a (tentar) aprender os novos conteúdos que lhes são apresentados. Em aulas de implementação de código, este *feedback* tem, muitas vezes, a forma de ajuda na resolução de erros (depuração). O método de providenciar esta ajuda deve ter como objetivo a explicitação do modelo mental que está na origem do erro (Ko & Myers, 2008) isso pode ser feito a partir da comparação do resultado (errado) obtido e a saída esperada (Ko, 2006).

5.2.1. CRITÉRIOS GERAIS DE AVALIAÇÃO, ENSINO PROFISSIONAL

No documento de Critérios Gerais de Avaliação deste ano letivo (Conselho Pedagógico, 2019) são indicados os seguintes descritores de desempenho para o Ensino Profissional:

Objeto de Avaliação	Itens/Parâmetros	Instrumentos
Domínio Cognitivo e Psicomotor	Conteúdos Programáticos	Prova de Avaliação Avaliações Práticas (montagens) Trabalhos de grupo (teórico/prático) Trabalhos Individuais (teórico/prático) Fichas formativas Fichas de trabalho Apresentação oral de um tema Portefólio
Metodologia de Trabalho	Intervém com pertinência e a propósito. Apresenta material necessário para a aula. Facilidade de expressão escrita e Oral. Realiza as tarefas propostas. Utiliza e domina métodos e técnicas de trabalho.	Utilização das TIC Portefolios Organização Documen- tal Manuseamento de materiais, aparelhos ou modelos didáticos Apresentação oral de trabalhos Utilização de meios complementar de comunicação e imagem
Atitudes/Valores	Participação Responsabilidade Relacionamento Interpessoal Interesse/Desempenho/Autonomia	Grelha com descritores de desempenho aprovada no agrupamento.

Tabela 5.1: Critérios Gerais de Avaliação, 2019/20, Ensino Profissional, Agrupamento de Escolas de São João da Talha.

5.2.2. PASSAPORTE DE APRENDIZAGEM

Uma forma de *gamificar* a aprendizagem ao mesmo tempo que se dá *feedback* contínuo aos alunos, é arranjar uma representação física das tarefas propostas onde se possa marcá-las como terminadas com sucesso. Para isso, foi criado um passaporte de aprendizagem (*vide* Figura 5.1). que lista todas as tarefas que os alunos irão realizar ao longo do módulo (*vide* Tabela 5.3).

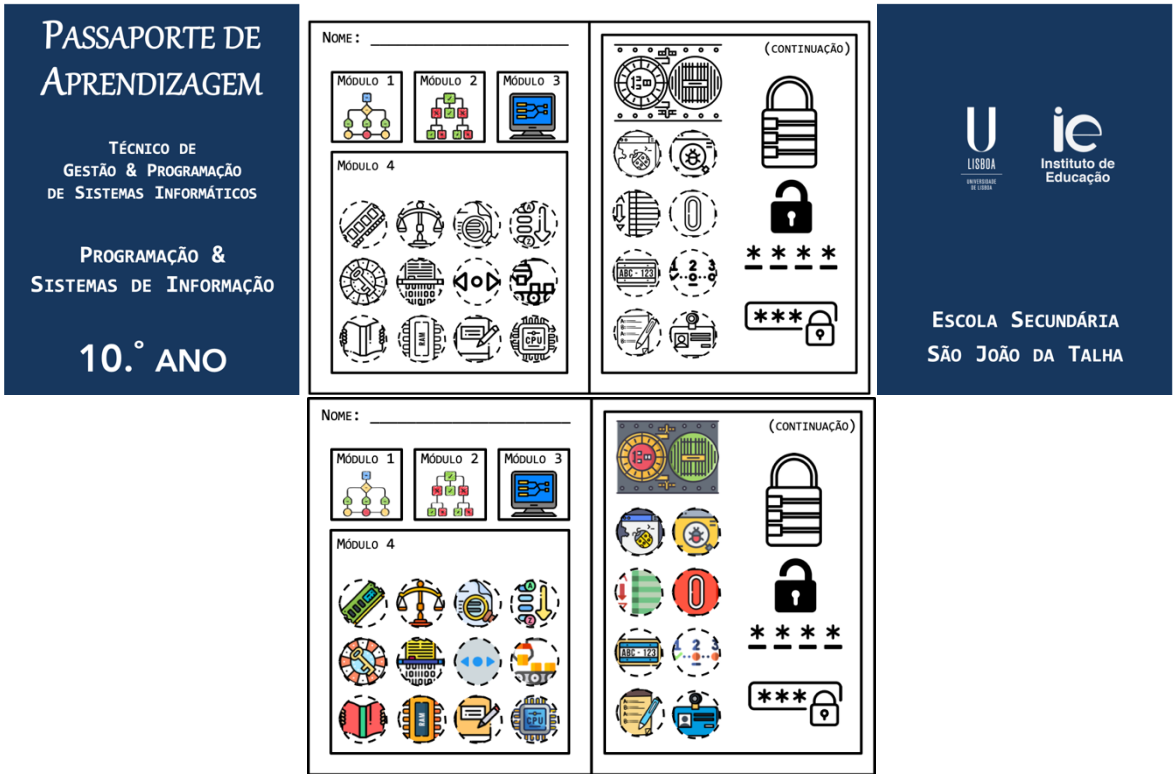


Figura 5.1: Passaporte de Aprendizagem criado para a intervenção a realizar

A primeira linha tem a referência aos módulos anteriores (*vide* Tabela 5.2), servindo de motivação (porque se espera que estejam terminados) ou de aviso a professores e aluno que é preciso, da parte do professor, uma diferenciação pedagógica para corrigir esse atraso, e da parte do aluno, um empenho adicional para conseguir recuperar o atraso.

Módulo 1: Introdução à Programação e Algoritmia	
Módulo 2: Mecanismos de Controlo de Execução	
Módulo 3: Programação Estruturada	

Tabela 5.2: Módulos anteriores representados no passaporte de aprendizagem

Módulo 4: Estruturas de Dados Estáticas	
Pesquisa em vetores (com caixas, gavetas ou portas) Pesquisa aleatória, linear, binária	
Ordenação de vetores (do conteúdo) Ordenação com explicitação do critério	
Filas Pilhas	
Intuição de Memória Espaço ocupado com cada tipo de dado	
Criptografia simples Cifras de César	
Binário ASCII, ISO Latin 1, UTF 8 e 16	
Leitura de código Reconhecimento dos algoritmos usados	
Leitura de código Representação de algoritmos em memória	
Escrita de código Implementação dos algoritmos estudados	
Big O: notação assimptótica Análise da complexidade e eficiência	
Debug Análise de código dos colegas	
Debug de erros mais comuns Exemplos de entrevistas de emprego (livros)	
`String` = `char[]` Acesso a letras como células de vetor	
`NULL`, `\0` Marcador de fim de `String`	
`float[]`, `long[]`, `double[]`	
`String[]` Intuição de matriz: célula e a letra da `String`	
Vetores alinhados: a mesma posição em cada Registo (BD), Classes (Objetos)	

Tabela 5.3: Tarefas relativas ao módulo 4 a realizar para completar o passaporte de aprendizagem

5.3. AVALIAÇÃO SUMATIVA

Os critérios de avaliação gerais do agrupamento aprovados para este ano letivo indicam que, tal como indicado no Decreto-Lei 55/2018, a avaliação deverá ser diversificada e as provas escritas não terão uma valoração superior a 50%.

Nos Critérios de Avaliação Específicos da disciplina, aprovados em Conselho Pedagógico da escola no início do ano foi decidido que os instrumentos de avaliação desta disciplina são: Provas escritas, Apresentações, Debates, Relatórios, Guiões, Trabalho de pesquisa/investigação, Práticas simuladas, Práticas experimentais, Exercícios práticos, Fichas de trabalho, Portefólios, Projetos, Trabalho interdisciplinar, Mostras à Comunidade, DAC, Grelha única do agrupamento para registo de Literacia Tecnológica, Grelha única do agrupamento para registo de Valores Sociais e de Cidadania.

Esta avaliação será feita na penúltima aula uma vez que a última é dedicada a uma atividade mais imersiva o que perturbará as condições para a execução de uma prova escrita.

5.4. AVALIAÇÃO DAS AULAS LECIONADAS

Na avaliação final, no último dia, além de se fazerem perguntas que visam a avaliação da compreensão dos conceitos por parte dos alunos, também serão feitas perguntas que recolham a opinião dos alunos sobre as aulas e os seus temas, e a sua apreciação das mesmas. Será feita uma apreciação dos temas abordados e das aulas lecionadas de acordo com três perguntas genéricas: “O que gostaste mais?”, “O que gostaste menos?” e “O que é que mudarias?”. As respostas serão agrupadas em temas para avaliação da qualidade da intervenção, do ponto de vista dos alunos.

Não serão feitas as perguntas tradicionais de avaliação da professora e do seu estilo porque nem sempre esse tipo de questões permite, de facto, aferir a qualidade das aulas, ou identificar problemas.

6. ATIVIDADES E ESTRATÉGIAS DE APRENDIZAGEM E SEU IMPACTO NA APRENDIZAGEM DOS ALUNOS

Durante a preparação da intervenção, foram vários os autores e as temáticas que motivaram discussões sobre como ensinar estes tópicos e quais as metodologias e técnicas mais promissoras. Seguindo a tradição portuguesa de ensinar a *"Ler, Escrever e Contar"*, surgiu a pergunta da relação entre a leitura e a escrita e qual a ordem dos fatores no ensino da Programação. Isto porque, muitas vezes, o foco está em copiar código e aprender esquemas, não obrigando os alunos a aprofundar o significado das técnicas utilizadas.

Assim, quisemos criar um espaço didático-pedagógico onde o foco, antes de ser a resolução dos problemas em si, fosse a capacidade de expressão e comunicação de ideias que os alunos já têm do senso comum, do dia-a-dia, obrigando-os à explicitação e sistematização das mesmas, para depois as poderem analisar criticamente. Para isso, utilizaremos tarefas do tipo puzzle matemático ou lógico, evitando os problemas de tradução das ideias para linguagens formais como a matemática ou as de programação.

Em particular abordar-se-ão as seguintes questões de investigação:

Q_1: Qual o impacto das atividades desligadas na compreensão conceptual dos algoritmos?

Q_2: Qual o impacto da leitura de (bom) código na implementação dos algoritmos relacionados?

Q_3: Como é que a representação pictórica do funcionamento de um programa em memória melhora a compreensão dos algoritmos?

As questões de investigação surgem, então, da vontade de perceber como os alunos conseguem aprender estes conceitos de várias formas e qual o impacto de cada estratégia nessa aprendizagem.

A revisão bibliográfica apontou várias pistas interessantes: a utilização de atividades desligadas; a leitura de bom código e não apenas a cópia de *templates* esvaziadas do devido aprofundamento teórico; a representação pictórica ou esquemática dos conceitos como passagem do concreto (experimentado nas atividades desligadas) para o abstrato (concretizado no código escrito numa determinada linguagem de programação); a correção de erros próprios e dos outros; e as técnicas de autorregulação. Seguimos essas pistas na planificação da intervenção e na avaliação procuraremos medir o seu impacto.

Dada a pequena dimensão da turma, não será possível fazer um estudo com significado estatístico. Optaremos, por isso, por um estudo de natureza experimental, que siga as técnicas habituais de recolha de dados: testes de aferição antes e depois das várias intervenções (concretizados nos *Admission* e *Exit Tickets*), observação com registo em grelha própria e questionários aos alunos. As respostas serão classificadas de acordo com uma escala, de forma a poderem ser comparadas e os progressos medidos. Utilizar-se-ão as quatro componentes do Pensamento Computacional: Abstração, Decomposição, reconhecimento de Padrões e Algoritmia, com especial ênfase neste último.

7. CONCLUSÃO

Neste relatório, do âmbito da unidade curricular de *Iniciação à Prática Profissional III*, apresentou-se a planificação da *Prática Supervisionada* que se realizará no âmbito da unidade curricular de *Iniciação à Prática Profissional IV*, que decorrerá na *Escola Secundária de São João da Talha*, do *Agrupamento de Escolas de São João da Talha*, de maio a junho de 2020, o professor cooperante José António Cruz, na turma do 10.º E, do Ensino Secundário Profissional. na disciplina de *Programação e Sistemas de Informação*, no módulo 4.

Relataram-se as observações já realizadas. Esse tempo serviu para conhecer a turma e para observar as técnicas pedagógicas e de gestão de sala de aula do professor cooperante. Foi também instrumental para identificar os ritmos e métodos da turma de forma a adaptar os materiais para irem de encontro ao estilo a que os alunos estão habituados.

Será feita uma intervenção de 1100 minutos (22 aulas de 50 minutos, com 2 aulas de 4 tempos, 200 minutos, e 7 aulas de 2 tempos, 100 minutos) na disciplina de *Programação e Sistemas de Informação* no módulo 4 referente a ***Estruturas de Dados Estáticas***. Serão avaliados os conhecimentos adquiridos pelos alunos de forma formativa e sumativa ao longo da intervenção. Será feita uma apreciação dos temas abordados e das aulas lecionadas

7.1. BALANÇO REFLEXIVO

Para alguém como eu, já com alguns anos de experiência letiva a vários níveis de ensino (do Pré-Escolar, ao Ensino Superior, passando pela Formação de Adultos), esta foi uma experiência muito interessante que me tem permitido observar uma turma do Ensino Secundário Profissional.

A exigência do presente relatório também me obrigou a considerar vários aspetos que, se não tivessem sido sistematizados, provavelmente não teriam emergido na minha reflexão.

Um professor deve lecionar utilizando métodos de aprendizagem ativa, por exemplo, promovendo o debate em sala de aula, partindo da experiência dos alunos. Deve também aplicar novas técnicas, aprendidas com os melhores professores. Exemplo desta tentativa é o Passaporte de Aprendizagem e as Atividades Desligadas. Não se conseguirá utilizar a Sala de Aula Invertida dado o horário preenchido que têm (que visa aproximar-se o mais possível do horário laboral das 35 a 40 horas), os alunos chegam a casa já tarde e sem muito tempo para fazerem trabalhos de casa de reforço ou para preparar a aula seguinte. Há também a dificuldade de encontrar materiais interessantes em Português e a proficiência em Inglês não é grande.

8. Referências

Direção Geral da Educação. (26 de julho de 2017). Perfil dos Alunos à Saída da Escolaridade Obrigatória. Despacho n.º 6478/2017.

Sultana, S. G., & Reed, P. L. (2017). Curriculum for an Introductory Computer Science Course: Identifying Recommendations from Academia and Industry. doi:10.21061/jots.v43i2.a.3

Prifti, L., Levkovskyi, B., Knigge, M., & Krcmar, H. (2018). Developing an Evaluation Model for Information Systems Curricula. *Multikonferenz Wirtschaftsinformatik*. Lüneburg, Germany.

Município de Loures. (2006). *Carta Educativa do Município de Loures* . Loures.

Alimisis, D. (2013). Educational Robotics: Open questions and new challenges. *Themes in Science and Technology Education*.

Benitti, F. (Abril de 2012). Exploring the educational potential of robotics in schools: A systematic review. *Computers & Education, 58*(3), 978-988.

Kay, J. S. (Dezembro de 2003). Teaching Robotics from a Computer Science perspective. *Journal of Computing Sciences in Colleges, 19*(2), 329-336. Obtido de http://dl.acm.org/citation.cfm?id=948785.948831

Pöhner, N., & Hennecke, M. (2018). Learning Problem Solving Through Educational Robotics Competitions: First Results of an Exploratory Case Study. *Proceedings of the 13th Workshop in Primary and Secondary Computing Education*. New York, NY, USA: ACM.

Diário da República. (6 de julho de 2018). Decreto-Lei n.º 55/2018. *1.ª série(n.º 129)*. Obtido de https://dre.pt/application/conteudo/115652962

Direção-Geral da Educação. (19 de julho de 2018). Aprendizagens Essenciais de Tecnologias da Informação e Comunicação para o 7.º ano, 3.º ciclo. *Despacho n.º 6944-A/2018*. Obtido de https://www.dge.mec.pt/sites/default/files/Curriculo/Aprendizagens_Essenciais/3_ciclo/tic_3c_7a_ff.pdf

Pedro, A., Matos, J. F., Piedade, J., & Dorotea, N. (2017). *Probótica: Programação e Robótica no Ensino Básico*. Lisboa, Portugal: Instittuto de Edicação da Universidade de Lisboa.

Schmitz, E. X. (2016). *Sala de aula invertida: uma abordagem para combinar metodologias ativas e engajar alunos no processo de ensino-aprendizagem*. (U. F. Maria, Ed.) Brasil. Obtido de https://repositorio.ufsm.br/handle/1/12043

Arends, R. (2014). *Learning to teach*. McGraw-Hill.

Cristo, A. H. (2013). *Escolas Para o Século XXI: Liberdade e Autonomia na Educação*. Lisboa, Portugal: Fundação Francisco Manuel dos Santos. doi:9789898424976

Lima, R. (2017). *A Escola que Temos e a Escola que Queremos: O que se passa com a educação?* Lisboa: Manuscrito Editora.

Bona, C. (2017). *A Nova Educação: o Professor que está a revolucionar a Escola*. Lisboa: Objectiva.

Lopes, J., & Santos Silva, H. (2012). *50 Técnicas de Avaliação Formativa*. Lisboa: Lidel.

Greenstein, L. (2010). *What Teachers Really Need to Know About Formative Assessment*. Alexandria, VA, USA: Association for Supervision & Curriculum Development.

L'Ecuyer, C. (2017). *Educar na Curiosidade: como educar num mundo frenético e hiperexigente?* Lisboa, Portugal: Editorial Planeta.

L'Ecuyer, C. (2018). *Educar na Realidade: alternativas para educar os nossos filhos face às novas tecnologias*. Lisboa, Portugal: Editorial Planeta.

Reis, P. (2011). *Observação de aulas e avaliação do desempenho docente* (Cadernos do CCAP #2 ed.). Lisboa, Portugal: Ministério da Educação - Conselho Científico para a Avaliação de Professores. Obtido de https://repositorio.ul.pt/handle/10451/4708

Witherspoon, E. B., Higashi, R. M., Schunn, C. D., Baehr, E. C., & Shoop, R. (2017). Developing Computational Thinking through a Virtual Robotics Programming Curriculum. *ACM Transactions on Computing Education (TOCE) , 18*(1). doi:doi.org/10.1145/3104982

Shoop, R., Schunn, C. D., Flot, J., Friez, T., & Witherspoon, E. B. (2016). Can Computational Thinking Be Taught In Robotics Classrooms? *International Technology and Engineering Conference*. National Harbor, Washington DC, USA.

NCW&IT. (2009). *Pair Programming-in-a-Box: The Power of Collaborative Learning*. National Center for Women & Information Technology. Obtido de www.ncwit.org/pairprogramming

Williams, L., Wiebe, E., Yang, K., Ferzli, M., & Miller, C. (2002). In Support of Pair Programming in the Introductory Computer Science Course. *Computer Science Education, 12*(3), 197-212. doi:doi.org/10.1076/csed.12.3.197.8618

Cockburn, A., & Williams, L. (2001). The Costs and Benefits of Pair Programming. Em G. Succi, & M. Marchesi, *Extreme Programming Examined* (pp. 223-243). Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. Obtido de dl.acm.org/citation.cfm?id=377517.377531

Bryant, S., Romero, P., & Du Boulay, B. (2008). Pair programming and the mysterious role of the navigator. *International Journal of Human-Computer Studies, 66*(7), 519-529. doi:doi.org/10.1016/j.ijhcs.2007.03.005

Williams, L., McCrickard, S. D., Layman, L., & Hussein, K. (2008). Eleven Guidelines for Implementing Pair Programming in the Classroom. *Agile 2008 Conference*. IEEE. doi:10.1109/Agile.2008.12

Matos, J. F. (2010). *Princípios Orientadores para o Desenho de Cenários de Aprendizagem*. Lisboa: Projeto LEARN.

McDowell, C., Werner, L., Bullock, H. E., & Fernald, J. (8 de agosto de 2006). Pair Programming improves Student Retention, Confidence, and Program Quality. *Communications of the ACM - Music Information ~Retrieval, 49*(8), pp. 90-95. doi:doi.acm.org/10.1145/1145287.1145293

Ulloa, M. (julho de 1980). Teaching and Learning Computer Programming: A Survey of Student Problems, Teaching Methods, and Automated Instructional Tools. *ACM SIGCSE Bulletin, 12*(2), pp. 48--64. doi:doi.acm.org/10.1145/989253.989263

Livingston, K. (2016). Teacher education's role in educational change. *European Journal of Teacher Education, 39*(1), 1-4. doi:doi.org/10.1080/02619768.2016.1135531

Palfrey, J. G., & Gasser, U. (2008). *Born Digital: Understanding the first generation of digital natives*. New York, NY, USA: Basic Books.

Bosse, Y., & Gerosa, M. A. (2016). Why is Programming So Difficult to Learn?: Patterns of Difficulties Related to Programming Learning Mid-Stage. *ACM SIGSOFT Software Engineering Notes, 41*(6), 1-6. doi:10.1145/3011286.3011301

Schmidt, P. (2017). When Students' Prejudices Taint Reviews of Instructors. *The Chronicle of Higher Education*.

Kornell, N., & Hausman, H. (25 de abril de 2016). Do the Best Teachers Get the Best Ratings? *Frontiers in Psychology, 7*(570). doi:doi:10.3389/fpsyg.2016.00570

Hazzan, O., Lapidot, T., & Ragonis, N. (2015). *Guide to Teaching Computer Science: an activity-based approach*. London: Springer-Verlag.

Bruce, K. B., Danyluk, A., & Murtagh, T. (2005). *Why structural recursion should be taught before arrays in CS1*. (Vol. 37). ACM SIGCSE Bulletin.

Selby, C. (2015). Relationships: Computational Thinking, Pedagogy of Programming, and Bloom's Taxonomy. *The 10th Workshop in Primary and Secondary Computing Education* (pp. 80-87). New York, NY, USA: ACM.

Watson, C. E., Terry, K. P., & Doolittle, P. E. (2012). 19: Please read while texting and driving. Em J. E. Groccia, *To Improve the Academy: Resources for Faculty, Instructional, and Organizational Development* (pp. 294-309). John Wiley & Sons.

Carvalho, J. (2005). *Programa da Componente de Formação Técnica: Disciplina de Redes de Comunicação | Cursos Profissionais de Nível Secundário: Técnico de Gestão e Programação de Sistemas Informáticos*. Obtido de Agência Nacional para a Qualificação e o Ensino Profissional, IP: http://www.anqep.gov.pt//wwwbase/wwwinclude/ficheiro.aspx?access=1&id=7490

Pinheiro, A. R. (2005). *Programa da Componente de Formação Técnica: Disciplina de Sistemas Operativos | Cursos Profissionais de Nível Secundário: Técnico de Gestão e Programação de Sistemas Informáticos*. Obtido de Agência Nacional para a Qualificação e o Ensino Profissional, IP: http://www.anqep.gov.pt//wwwbase/wwwinclude/ficheiro.aspx?access=1&id=7487

Rodrigues, R. (2005). *Programa da Componente de Formação Técnica: Disciplina de Arquitetura de Computadores | Cursos Profissionais de Nível Secundário: Técnico de Gestão e Programação de Sistemas Informáticos*. Obtido de Agência Nacional para a Qualificação e o Ensino Profissional, IP: http://www.anqep.gov.pt//wwwbase/wwwinclude/ficheiro.aspx?access=1&id=7488

DGFV. (2005). *Programa da Componente de Formação Técnica: Disciplina de Programação e Sistemas de Informação | Cursos Profissionais de Nível Secundário: Técnico de Gestão e Programação de Sistemas Informáticos*. Obtido de Agência Nacional

para a Qualificação e o Ensino Profissional, IP: http://www.anqep.gov.pt//wwwbase/wwwinclude/ficheiro.aspx?access=1&id=7489

Alves Marques, J., & Guedes, P. (1998). *Fundamentos de Sistemas Operativos.* Editorial Presença.

Tanenbaum, A. S. (1987). *Operating Systems: Design and Implementation.* Londres: Prentice-Hall International.

Shuaibu, M. B., & Ibrahim, R. A. (2017). Web application development model with security concern in the entire life-cycle. *4th International Conference on Engineering Technologies and Applied Sciences* (pp. 1-6). Salmabad: IEEE.

Stallings, W., & Brown, L. (2015). *Computer Security: Principles and Practice, 3rd Edition.* Pearson Education Limited.

Norris, D. (2015). *The Internet of things: do-it-yourself projects with Arduino, Raspberry Pi, and BeagleBone Black.* McGraw-Hill Education.

Metcalf, D., Milliard, S. T., & Gomez, M. (2016). Wearables and the internet of things for health: Wearable, interconnected devices promise more efficient and comprehensive health care. *IEEE Pulse, 7*(5), 35-39.

Victor, B. (2011). *Up and Down the Ladder of Abstraction*. Obtido de Bret Victor, purveyor of impossible dreams: http://worrydream.com/LadderOfAbstraction/

Krivitsky, A. (2017). *#LEGO4SCRUM: SCRUM simulation with LEGO.* Obtido de https://www.lego4scrum.com/

Carmo, D., Pardal, J. P., & Venâncio, S. (Outubro de 2013). *LEGO4SCRUM em Português v2.0.* Obtido de Simulação de Scrum com Peças de Lego - Edição para Pequenos e Médios Negócios : https://www.lego4scrum.com/translations/

Athalye, A., Gjengset, J., & Ortiz, J. G. (2020). *The Missing Semester of your CS Education*. Obtido de https://missing.csail.mit.edu/

Laudon, K. C., & Laudon, J. P. (2019). *Management Information Systems: Managing the Digital Firm.* Pearson Education.

Mullins, P., Whitfield, D., & Conlon, M. (2009). Using Alice 2.0 as a first language. *Journal of Computing Sciences in Colleges, 24*(3), 136-143.

Cooper, S., Dann, W., & Pausch, R. (2000). Alice: a 3-D tool for introductory programming concepts. *Journal of Computing Sciences in Colleges, 15*(5), 107-116.

Adobe. (2020). *Flash & The Future of Interactive Content.* Obtido de Adobe Blog: https://theblog.adobe.com/adobe-flash-update/

Bolkan, S., & Goodboy, A. K. (2019). Instruction, example order, and student learning: reducing extraneous cognitive load by providing structure for elaborated examples. *Communication Education*, 1-17.

Bolkan, S., & Goodboy, A. K. (2019). Examples and the facilitation of student learning: should instructors provide examples or should students generate their own? *Communication Education, 68*(3), 287-307.

Vrachnos, E., & Jimoyiannis, A. (2017). Secondary education students' difficulties in algorithmic problems with arrays: An analysis using the SOLO taxonomy. *Themes in Science and Technology Education, 10*(1), 31-52.

Schulte, C., & Bennedsen, J. (2006). What do teachers teach in introductory programming? *Proceedings of the 2nd international workshop on Computing Education Research*, (pp. 17-28).

Dale, N. B. (junho de 2006). Most difficult topics in CS1: results of an online survey of educators. *InRoads , 38*(2), pp. 49- 53.

Manso, A., Oliveira, L., & Marques, C. G. (2009). Ambiente de aprendizagem de algoritmos – Portugol IDE. *Conferência Internacional de TIC na Educação, 6.*

Papadakis, S. J., & Orfanakis, V. (janeiro de 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).

Teague, D., & Lister, R. (2014). Longitudinal think aloud study of a novice programmer. *Proceedings of the 16h Australasian Computing Education Conference. 148*, pp. 41-50. Auckland, New Zealand: Australian Computer Society, Inc.

Yan, L., Hu, A., & Piech, C. (2019). Pensieve: Feedback on coding process for novices. *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (pp. 253-259). ACM.

Öqvist, M., & Nouri, J. (junho de 2018). Coding by hand or on the computer? Evaluating the effect of assessment mode on performance of students learning programming. *Journal of Computers in Education, 5*(2), 199-219.

Danielsiek, H., Wolfgang, P., & Vahrenhold, J. (2012). Detecting and understanding students' misconceptions related to algorithms and data structures. *Proceedings of the 43rd ACM technical symposium on Computer Science Education* (pp. 21-26). Raleigh, North Carolina, USA: ACM SIGCSE.

McCracken, M., Almstrum, V., Diaz, D., Guzdial, M., Hagan, D., Kolikant, Y. B.-D., . . . Wilusz, T. (2001). A multi-national, multi-institutional study of assessment of programming skills of first-year CS students. *ACM SIGCSE Bulletin , 33*(4), pp. 125-140.

Dasgupta, S., & Hill, B. M. (2017). Learning to code in localized programming languages. *Proceedings of the 4th Conference on Learning@Scale* (pp. 33-39). ACM.

Weintrop, D., & Wilensky, U. (2019). Transitioning from introductory block-based and text-based environments to professional programming languages in high school computer science classrooms. *Computers & Education, 142*(103646), 1-17.

Ball, D. L., Hill, H. C., & Bass, H. (2005). Knowing Mathematics for Teaching: who knows mathematics well enough to teach third grade, and how can we decide? *American Educator*.

Malan, D. J. (2019). *This is CS50*. Obtido de Harvard University: https://cs50.harvard.edu/

Papancea, A., Spacco, J., & Hovemeyer, D. (2013). An open platform for managing short programming exercises. Em A. C. Beth Simon, *Proceedings of the 9th annual international ACM conference on International computing education research* (pp. 47-51). New York, NY, United States: Association for Computing Machinery.

Malan, D. J. (2010). Reinventing CS50. *Proceedings of the 41st ACM technical symposium on Computer Science Education* (pp. 152-156). Milwaukee, Wisconsin, USA: ACM SIGCSE.

MacWilliam, T. M., Aquino, R. J., & Malan, D. J. (2013). Engaging students through video: integrating assessment and instrumentation.

Lai, C. H., Yang, J. C., Liang, J. S., & Chan, T. W. (2005). Mobile learning supported by learning passport. *5th IEEE International Conference on Advanced Learning Technologies* (pp. 595-599). IEEE ICALT'05.

Robertson, S. A., & Lee, M. P. (1995). The application of second natural language acquisition pedagogy to the teaching of programming languages-a research agenda. *ACM SIGCSE Bulletin, 27*(4), pp. 9–12.

Baldwin, L. P., & Macredie, R. D. (1999). Beginners and Programming: insights from second language learning and teaching. *Education and Information Technologies, 4*(2), pp. 167-179.

Milková, E. (2015). Development of programming capabilities inspired by foreign language teaching. *Procedia - Social and Behavioral Sciences, 171*(5th ICEEPSY International Conference on Education & Educational Psychology), 172 – 177.

Xie, B., Loksa, D., Nelson, G. L., Davidson, M. J., Dong, D., Kwik, H., . . . Ko, A. J. (2019). A theory of instruction for introductory programming skills. *Computer Science Education*, 1-49.

Deus, J. d. (1876). *Cartilha Maternal ou Arte de Leitura.* Porto: Typ. de Antonio José da Silva Teixeira.

Bennett, R. B. (2014). *The effect of Math in Focus: the Singapore approach on elementary students' mathematics achievement.* Union University.

Xie, B., Nelson, G., & Ko, A. J. (2018). An Explicit Strategy to Scaffold Novice Program Tracing,. *Technical Symposium on Computer Science Education, Research Track* (pp. 344-349). ACM SIGCSE.

Knop, K. (2019). *Weights and Algorithms: From Puzzles to Tasks.* School Mathematical Circles, Litres.

Knuth, D. E. (1985). Algorithmic Thinking and Mathematical Thinking. *The American Mathematical Monthly, 92*(3), pp. 170-181.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33-35.

Dagiene, V., & Stupuriene, G. (2017). Algorithms Unplugged: a Card Game of the Bebras-like Tasks for High Schools Students. *The 10th International Conference on Informatics in Schools (ISSEPS).*

Dagiene, V., Futschek, G., Koivisto, J., & Stupurienė, G. (2017). *The card game of Bebras-like tasks for introducing informatics concepts.*

Lockwood, J. &. (2018). *Developing a Computational Thinking Test using Bebras Problems.*

Chiazzese, G. M. (dezembro de 2019). Educational Robotics in Primary School: Measuring the Development of Computational Thinking Skills with the Bebras Tasks. *Informatics, 6*(4), p. 43.

Bloom, B. S. (1956). *Taxonomy of educational objectives. Vol. 1: Cognitive domain.* New York: McKay.

Sprouts. (3 de janeiro de 2019). *Bloom's Taxonomy: Structuring The Learning Journey.* Obtido de YouTube: https://www.youtube.com/watch?v=ayefSTAnCR8

Krathwohl, D. R., & Anderson, L. W. (2009). *A Taxonomy for Learning, Teaching, and Assessing: a revision of Bloom's taxonomy of educational objectives.* Longman.

Thompson, E., Luxton-Reilly, A., Whalley, J. L., Hu, M., & Robbins, P. (2008). Bloom's taxonomy for CS assessment. *Proceedings of the 10th conference on Australasian Computing Education*, *78*, pp. 155-161.

Bell, T., Alexander, J., Freeman, I., & Grimley, M. (2009). *Computer Science Unplugged: school students doing real computing without computers, 13*(1), 20-29.

Feaster, Y., Segars, L., Wahba, a. K., & Hallstrom, J. O. (2011). Teaching CS unplugged in the high school (with limited success). *Proceedings of the 16th annual joint conference on Innovation and Technology in Computer Science Education*, (pp. 248-252).

Bell, T., Alexander, J., Freeman, I., & Grimley, M. (2008). Computer science without computers: new outreach methods from old tricks. *Proceedings of the 21st annual conference of the national advisory committee on computing qualifications.*

Bell, T., & Vahrenhold, J. (2018). CS Unplugged - How is it used, and does it work? Em H.-J. Böckenhauer, D. Komm, & W. Unger, *Adventures Between Lower Bounds and Higher Altitudes* (Vol. 11011, pp. 497-521). Zürich, Switzerland: Springer, Cham.

Rodriguez, B., Kennicutt, S., Rader, C., & Camp, T. (2017). Assessing computational thinking in cs unplugged activities. *Proceedings of the Technical Symposium on Computer Science Education* (pp. 501-506). ACM SIGCSE.

Martens, S., & Crawford, K. (2019). Embracing Wonder and Curiosity: Transforming teacher practice through escape room design. *Childhood Education, 95*(2), 68-75.

Ho, A. M. (2018). Unlocking ideas: using escape room puzzles in a cryptography classroom. *Primus, 28*(9), 835-847.

Butler, S., & Ahmed, D. T. (2016). Gamification to engage and motivate students to achieve computer science learning goals. *International Conference on Computational Science and Computational Intelligence* (pp. 237-240). IEEE.

Deterding, S., Dixon, D., Khaled, R., & Nacke, L. (2011). From game design elements to gamefulness: defining "gamification". *Proceedings of the 15th international academic MindTrek conference: Envisioning future media environments* (pp. 9-15). ACM.

Chen, T.-Y., Lewandowski, G., McCartney, R., Sanders, K., & Simon, B. (2007). Commonsense computing: using student sorting abilities to improve instruction. *Proceedings of the 38th technical symposium on Computer Science Education* (pp. 276-280). ACM SIGCSE.

Hunt, A., & David, T. (2000). *The Pragmatic Programmer: from Journeyman to Master.* Boston, MA, USA: Addison Wesley.

Nelson, G. L., Xie, B., & Ko, A. J. (2017). Comprehension first: evaluating a novel pedagogy and tutoring system for program tracing in CS1. *Proceedings of the ACM Conference on International Computing Education Research* (pp. 2-11). ACM SIGCSE.

Dastyni Loksa, A. J. (2016). Programming, Problem Solving, and Self-Awareness: Effects of Explicit Guidance. *Conference on Human Factors in Computing Systems* (pp. 1449-1461). ACM SIGCHI.

Pound, L. (2014). Maria Montessori and the Montessori method. Em L. Pound, *How Children Learn* (pp. 38-41). London, UK: Practical Pre-School Books.

Hazzan, O., Lapidot, T., & Ragonis, N. (2014). 11. Teaching Planning. Em O. Hazzan, T. Lapidot, & N. Ragonis, *Guide to Teaching Computer Science: An Activity-Based Approach* (pp. 207-219). London: Springer Publishing Company, Incorporated.

Nishida, T., Idosaka, Y., Hofuku, Y., Kanemune, S., & Kuno, Y. (2008). New methodology of information education with "computer science unplugged". *International Conference on Informatics in Secondary Schools - Evolution and Perspectives* (pp. 241-252). Berlin, Heidelberg: Springer.

Roussel, S., Joulia, D., Tricot, A., & Sweller, J. (2017). Learning subject content through a foreign language should not ignore human cognitive architecture: a cognitive load theory approach. *Learning and Instruction, 52*, 69-79.

Bell, T., Duncan, C., Jarman, S., & Newton, H. (2014). *Presenting computer science concepts to high school students.* Obtido de Computer Science Field Guide: https://csfieldguide.org.nz

Loksa, D., Ko, A. J., Jernigan, W., Oleson, A., Mendez, C., & Burnett, M. M. (2016). Programming, Problem Solving, and Self-Awareness: Effects of Explicit Guidance. *Conference on Human Factors in Computing Systems* (pp. 1449-1461). ACM CHI.

Pardal, J. P. (2019). *Introdução à Programação e Robótica: Observação e Intervenção.* São João da Talha, Loures: Iniciação à Prática Profissional II.

Weigend, M., Vaníček, J., Pluhár, Z., & Pesek, I. (2019). Computational Thinking Education through Creative Unplugged Activities. *Olympiads in Informatics, 13*, pp. 171–192.

Devadas, S. (2017). *Programming for the Puzzled: Learn to Program while Solving Puzzles.* Cambridge, MA, USA: MIT Press.

Oliveira, A. (2017). *The Digital Mind: How Science Is Redefining Humanity*. Cambridge, MA: MIT Press.

Forišek, M., & Steinová, M. (2012). Metaphors and analogies for teaching algorithms. *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, (pp. 15-20).

Ko, A. (2006). Debugging by asking questions about program output. *Proceedings of the 28th international Conference on Software Engineering*, (pp. 989-992).

Ko, A., & Myers, B. (2008). Debugging reinvented: Asking and Answering Why and Why Not Questions about Program Behavior. *30th International Conference on Software Engineering* (pp. 301-310). ACM/IEEE.

Conselho Pedagógico. (2019). *Critérios Gerais de Avaliação 2019/2020.* São João da Talha, Loures, Portugal: Agrupamentos de Escolas de São João da Talha.

9. ANEXOS

A. TAXONOMIA DE BLOOM APLICADA AO PENSAMENTO COMPUTACIONAL

Figura 9.1: Adequação do tipo de questões à competência de Pensamento Computacional

B. PLANO DO RELATÓRIO

De acordo com as normas dos Mestrados em Ensino do Instituto de Educação, foi registado o tema da tese e indicado o plano previsto para o Relatório de Prática Supervisionada, que se segue.

1. Introdução
- 1.1. Objetivos Gerais e Específicos
- 1.2. Problemática e Questões de Investigação
- 1.3. Estratégias e Metodologias a explorar
  - 1.3.1. Atividades Desligadas (do Inglês: Unplugged)
  - 1.3.2. Leitura antes da Escrita (em linguagens humanas e artificiais)
  - 1.3.3. Representação Pictórica como interface entre o Concreto e o Abstrato
  - 1.3.4. Aprendizagem Baseada em Problemas
  - 1.3.5. Mecanismos de Autorregulação
    - 1.3.5.1. Persistência e Mentalidade de Crescimento (do Inglês: *Growth Mindset*)
    - 1.3.5.2. Depuração com patinhos de borracha (do Inglês: *Rubber Duck Debugging*)
    - 1.3.5.3. Pergunta a três antes de mim (Cérebro, Web, Colega)
    - 1.3.5.4. Passaporte de Aprendizagem

2. Contexto Escolar da Intervenção
- 2.1. Contexto Social
  - 2.1.1. Agrupamento de Escolas de São João da Talha
  - 2.1.2. Escola Secundária de São João da Talha
    - 2.1.2.1. Oferta Educativa
- 2.2. Contexto de Turma
  - 2.2.1. Breve Caracterização da Turma 10.º E
    - 2.2.1.1. Horário da Turma
    - 2.2.1.2. Resultados dos Módulos 1, 2 e 3
  - 2.2.2. Professor cooperante
  - 2.2.3. Sala de Aula F48
  - 2.2.4. Recursos Didáticos disponíveis na Escola
  - 2.2.5. Tecnologias e Aplicações disponíveis

3. Enquadramento Curricular
- 3.1. Currículo do Curso Profissional de nível Secundário de Técnico de Gestão e Programação de Sistemas Informáticos
- 3.2. Análise Crítica das disciplinas Técnicas do Curso
  - 3.2.1. Sistemas Operativos
  - 3.2.2. Arquitetura de Computadores
  - 3.2.3. Redes de Comunicação
  - 3.2.4. Programação e Sistemas de Informação
- 3.3. Módulos da disciplina de Programação e Sistemas de Informação
  - 3.4. Módulo #4: Estruturas de Dados Estáticas
  - 3.4.1. Calendarização
  - 3.4.2. Planificação
- 3.5. Conceitos associados
  - 3.5.1. Estruturas de dados
    - 3.5.1.1. Estruturas de dados estáticas
    - 3.5.1.2. Outras estruturas de dados
- 3.6. Dificuldades de aprendizagem associadas à Programação.
- 3.7. Estratégias e Metodologias de Ensino e Aprendizagem
- 3.8. Estratégias de Autorregulação

4. Intervenção Pedagógica
- 4.1. Observação de aulas dos Módulos 1, 2 e 3
  - 4.1.1. Módulo 1
  - 4.1.2. Módulo 2
  - 4.1.3. Módulo 3
- 4.2. Planificação
  - 4.2.1. Objetivos e Competências
  - 4.2.2. Estratégias e Metodologias
  - 4.2.3. Recursos
  - 4.2.4. Cenário de Aprendizagem
- 4.3. Operacionalização descritiva
  - 4.3.1. Aula #01 e 02: Atividades Desligadas – intuição de estruturas de dados, procura e ordenação
  - 4.3.2. Aula #03 e 04: Atividades Desligadas – criptografia, binário e ASCII
  - 4.3.3. Aula #05 e 06: Leitura de (Bom) Código (pesquisa e ordenação de inteiros); Identificação dos algoritmos
  - 4.3.4. Aula #07 e 08: Leitura de (Bom) Código e representação esquemática do seu funcionamento em memória
  - 4.3.5. Aula #09 e 10: Análise Crítica de Complexidade e Eficiência
  - 4.3.6. Aula #11 e 12: Escrita de Código e sua Análise Crítica (Complexidade e Eficiência)
  - 4.3.7. Aula #13 e 14: Depuração de Código (de colegas e de erros comuns)
  - 4.3.8. Aula #15 e 16: String como cadeia de caracteres (char[]) e o caracter NULL
  - 4.3.9. Aula #17 e 18: Cadeias de outros tipos de dados (float, long, double)
  - 4.3.10. Aula #18 e 19: Vetores alinhadas: representação de informação
  - 4.3.11. Aula #20 e 21: Escape Room Criptográfico; Autoavaliação; Avaliação da Intervenção

5. Avaliação da Intervenção
- 5.1. Avaliação Diagnóstica
- 5.2. Avaliação Formativa
- 5.3. Avaliação Sumativa
- 5.4. Autoavaliação
- 5.5. Avaliação das Aulas Lecionadas
  - 5.5.1. Apreciação das Aulas
    - 5.5.1.1. Comportamento, Atitudes, Assiduidade e Pontualidade
  - 5.5.2. Avaliação da Aprendizagem dos Alunos

6. Componente Investigativa da Intervenção
- 6.1. Plano Metodológico
  - 6.1.1. Paradigma Investigativo
  - 6.1.2. Estratégias e Métodos de Investigação
  - 6.1.3. Técnicas de Recolha de Dados
- 6.2. Dados recolhidos
- 6.3. Avaliação e Discussão de Resultados

7. Balanço Reflexivo

8. Conclusão

9. Referências Bibliográficas

10. Anexos

C. CENÁRIO DE APRENDIZAGEM: PASSAPORTE DE APRENDIZAGEM

Figura 9.2: Cenário de Aprendizagem: Passaporte de Aprendizagem e *Escape Room* Criptográfico

D. OFERTA NO AGRUPAMENTO DE ESCOLAS EM 2018/19 E 2019/20

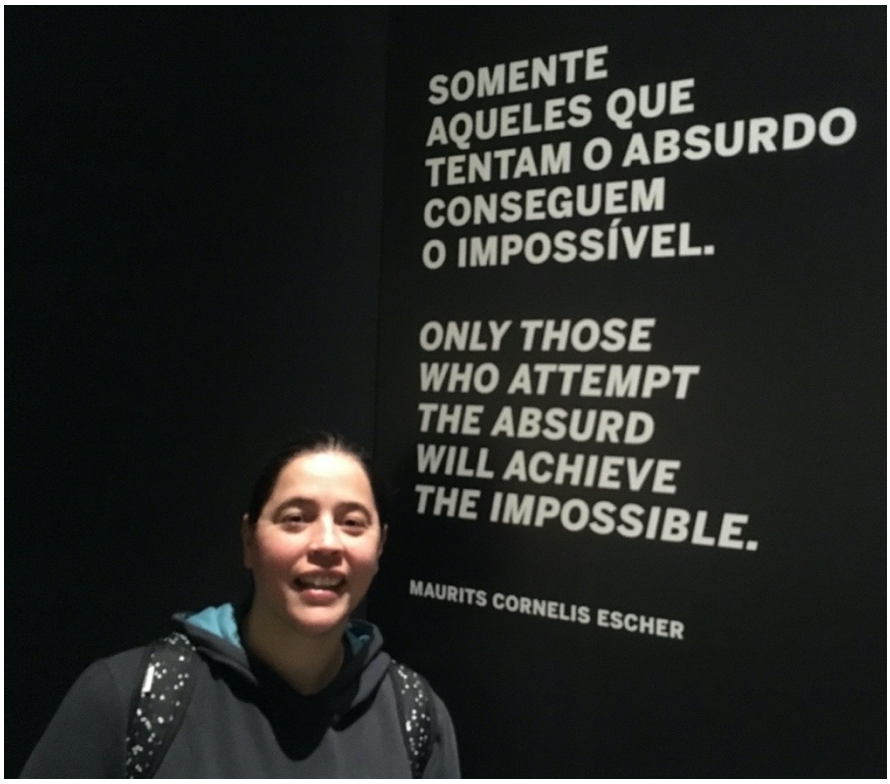
Fonte: Página oficial da Escola Secundária de São João da Talha
https://sites.google.com/a/aesjt.pt/agrupamento-de-escolas-de-s-joao-da-talha/noticias/ofertaformativadenivelsecundariopara20192020

E. OFERTA NO AGRUPAMENTO DE ESCOLAS EM 2017/18

Fonte: Página do Facebook do Agrupamento de Escolas de São João da Talha
https://www.facebook.com/agrupamentoescolassaojoaodatalha/photos/a.159127807602960/813204695528598/

F. OFERTA NO AGRUPAMENTO DE ESCOLAS EM 2016/17

Fonte: Página do Facebook do Agrupamento de Escolas de São João da Talha
https://www.facebook.com/agrupamentoescolassaojoaodatalha/photos/a.159127807602960/648515405330862/



Joana Paulo Pardal, 22678