UNIVERSIDADE FEDERAL DO TOCANTINS
CAMPUS UNIVERSITÁRIO DE PALMAS
CURSO DE CIÊNCIA DA COMPUTAÇÃO

ANÁLISE DA APLICAÇÃO DA METODOLOGIA CODE CLUB EM ESCOLAS PÚBLICAS DO TOCANTINS

BRUNA COQUI RODRIGUES

PALMAS (TO)
2019

BRUNA COQUI RODRIGUES

ANÁLISE DA APLICAÇÃO DA METODOLOGIA CODE CLUB EM ESCOLAS PÚBLICAS DO TOCANTINS

Trabalho de Conclusão de Curso II apresentado à Universidade Federal do Tocantins para obtenção do título de Bacharel em Ciência da Computação, sob a orinetação do(a) Prof.(a) Dsc. Eduardo Ferreira Ribeiro.

Orientador: Dsc. Eduardo Ferreira Ribeiro

PALMAS (TO)
2019

Dados Internacionais de Catalogação na Publicação (CIP)
Sistema de Bibliotecas da Universidade Federal do Tocantins

R696a Rodrigues, Bruna.
ANÁLISE DA APLICAÇÃO DA METODOLOGIA CODE CLUB EM ESCOLAS PÚBLICAS DO TOCANTINS. / Bruna Rodrigues. – Palmas, TO, 2021.
75 f.

Monografia Graduação - Universidade Federal do Tocantins – Câmpus Universitário de Palmas - Curso de Ciências da Computação, 2021.
Orientador: Eduardo Ribeiro

1. Programação. 2. Ensino de programação. 3. Programação para crianças. 4. Codeclub. I. Título

CDD 004

Elaborado pelo sistema de geração automatica de ficha catalográfica da UFT com os dados fornecidos pelo(a) autor(a).

BRUNA COQUI RODRIGUES

ANÁLISE DA APLICAÇÃO DA METODOLOGIA CODE CLUB EM ESCOLAS PÚBLICAS DO TOCANTINS

Trabalho de Conclusão de Curso II apresentado à UFT – Universidade Federal do Tocantins – Campus Universitário de Palmas, Curso de Ciência da Computação foi avaliado para a obtenção do título de Bacharel e aprovada em sua forma final pelo Orientador e pela Banca Examinadora.

Data de aprovação: 8 / 8 / 2019

Banca Examinadora:

Prof. Dsc. Eduardo Ferreira Ribeiro

Prof. Msc. Juliana Leitão Dutra

Prof. Dsc. Warley Gramacho da Silva

Obrigada, papai do céu, por me dar forças e permitir que esse projeto fosse concluído.

AGRADECIMENTOS

Hoje todos os meus agradecimentos vão para o professor Eduardo, meu orientador. Esse projeto não teria sido concluído se ele não tivesse se disposto tanto a realmente me ajudar. Foi uma luta, mas foi ele que tornou possível essa aprovação, e eu serei eternamente grata pela ajuda competente e humana que tive. Obrigada professor, que papai do céu te dê em dobro toda a felicidade que o senhor me causou pela finalização desse projeto.

RESUMO

Há um projeto mundial chamado Code Club World, que é patrocinado por grandes empresas, como a Google e a ARM, que visa levar o início do ensino de programação para as crianças de 9 à 13 anos do mundo inteiro. O projeto já conta com uma estrutura de aulas e exercícios que foi desenvolvido pelo próprio Code Club, que guia as crianças para aprenderem scratch, HTML e python. Os projetos introduzem gradualmente conceitos de programação, de uma maneira que permite que as crianças possam construir seu repertório de conhecimento de maneira incremental (FOUNDATION, 2017).O constante desenvolvimento tecnológico é algo inegável. A enorme importância e influência que a tecnologia e seus dispositivos têm na vida das pessoas também. Com isso, não são mais somente as pessoas que querem seguir uma carreira computacional que precisam saber e entender sobre computadores: esse uso cotidiano tornou o conhecimento em programação de extrema importância para todos. Visando a necessidade de ensino de programação para as crianças, este trabalho tem como objetivo estudar a viabilidade da implementação de um Code Club nas escolas públicas da cidade de Palmas-TO. Para isso, técnicas de Interface Homem Máquinas são testadas, como a avaliação preditiva e a avaliação por observação para a realização dessa análise. Os dados coletados durante a avaliação por observação são analisados para concluir qual o alcance desse projeto e quais as maiores dificuldades encontradas para a implantação do mesmo em um contexto local, considerando o nível dos alunos de Palmas antes e depois do projeto.

Palavra-chave: programação. ensino de programação. programação para crianças. codeclub. scratch.

ABSTRACT

There is a worldwide project called Code Club World, which is sponsored by huge companies, as Google and ARM, and it aims to bring the programming learning to children from 9 to 13 years old of the whole world. The project already has an structure of classes and exercises that were developed by Code Club itself, which guides the children to learn scratch, HTML CSS and python. The projects gradually insert concepts of programming, in a way that allows children to build their repertoire of knowledge in an incremental way (FOUNDATION, 2017). The constant technological development is something undeniable. The huge importance and the influence that the technology and its devices have in people's lives too. Thereby, it's not just the people who want to persue a computational career that need to know and understand about computers anymore: this daily use has become the knowledge in programming of extreme importance for all. Seeing the need of the learning of programming for children, the goal of this paper is to study the feasibility of implementation of a Code Club in the public schools of Palmas - TO. For that, Human Interface Machines techniques are tested, as predictive assessment and observation assessment for the achievement of this analysis. The data collected during the observation assessment are analyzed to conclude what is the scope of this project and which are the biggest difficulties found to its implementation in a local context, considering the level of the students in Palmas before and after the project.

Keywords: programming. learning how to program. programming for kids. codeclub. scratch.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

Na sociedade atual, em decorrência da popularização dos computadores, os mesmos se tornaram uma parte padrão da maioria das casas (SEYBERT, 2012). Isso resultou em uma nova geração de crianças e jovens que já são familiares com computadores no começo da sua educação primária (PALFREY AND GASSER, 2008). Eles possuem conhecimento intermediário de computadores e já estão familiarizados com conceitos básicos. Esse fato permite métodos diferentes de ensiná-los, um método que não era possível de ser aplicado para as gerações anteriores. O uso correto de computadores, acrescido dos métodos de ensino padrão, podem ajudar a melhorar a experiência de aprendizado para crianças e jovens, apresentando a eles problemas específicos de uma maneira que seja apropriada para a idade deles. A ideia básica é incorporar o aprendizado à algo que crianças e jovens tenham como divertido (MELUSO et al, 2012).

Computadores são considerados uma das invenções mais importantes do século 20, e uma das implicações acordadas universalmente é que as pessoas deveriam ser alfabetizadas em computadores, e isso cria a necessidade da introdução de computadores no currículo do ensino médio (GAL-EZER et al., 1995), e a inserção de lógica de programação para crianças.

O século 21 é caracterizado pela presença constante da tecnologia na vida cotidiana (SAELI et al., 2010). Ou seja, é indiscutível a presença e a importância cotidiana da tecnologia na vida das pessoas, e a facilidade que as crianças e jovens tem no entendimento do uso de dispositivos tecnológicos torna mais compreensível o estudo dos fragmentos da ciência da computação.

Em 1985, Abelson and Sussman afirmaram que ciência da computação é "uma disciplina de construção apropriada de linguagens descritivas" (ABELSON; SUSSMAN; SUSSMAN, 1985). E Denning afirmou que ciência da computação consiste em mecânica (computação, comunicação, coordenação, automação e recoleção), princípios de design (simplicidade, performance, envolvabilidade e segurança) e práticas (programação, sistemas de engenharia, modelo e validação, inovação e aplicação.) (DENNING, 2003). O modelo de currículo para ensino fundamental e médio de ciência da computação da ACM (TUCKER, 2003), define ciência da computação como: "não é nem programar, e nem literatura computacional, mas sim o estudo de computadores e processos algoritímicos incluindo seus princípios, hardware e software, suas aplicações e o impacto na sociedade".

Por essas razões, e com a visão de tamanha importância que a tecnologia tem no mundo, em 2016, o presidente da maior potêncial mundial do mundo - ex-presidente Barack Obama- apresentou uma iniciativa de "ciência da computação para todos", que investiu 4 bilhões de dólares em financiamentos para a educação de ciência da computação

para ensino fundamental e médio nos EUA (SMITH, 2016). Esse interesse em ciência da computação é refletido em iniciativas similares pelos EUA e pelo mundo também (PRETZ, 2014), (GROSS, 2015) (WHEATLEY, 2016). O objetivo não era somente prover aos estudantes as habilidades de programação necessárias para carreiras de alta tecnologia, mas também para fornecer a eles as habilidades de pensamento computacional que eles precisarão para carreiras em um mundo dirigido pela tecnologia.

Do outro lado do mundo, o discurso de Eric Schmidt na MacTaggart em Edinburgh em 2011, estimulou uma revisão na provisão educacional do ensino fundamental e médio no Reino Unido. Ele ressaltou que a geração das crianças da escola primária não estava recebendo uma educação computacional substantiva (SCHMIDT, 2011). "Para ser contundente, as lições de ICT1[1] hoje consistem principalmente em aprender Office. Nós estamos criando uma nação de secretárias?" Clare Sutclife, fundadora do Code Club - que é explanado na seção 2.2.

Por fim, após esse discurso foi implantado o novo currículo nacional obrigatório pelo governo no Reino Unido, o qual inclui diversas mudanças, como a introdução de algoritmos para crianças abaixo de 7 anos de idade. No entanto, essa implantação do currículo inglês demorou mais tempo do que o esperado devido a falta de professores especialistas no assunto (FURBER, 2012). Porém, essa preocupação levou a Google a disponibilizar 15 milhões para capacitação e treinamento de professores.

E tal fato não é diferente no Brasil, pois o máximo de incentivo que é oferecido para a capacitação de professores para o ensino de programação infantil é por meio da licenciatura de informática, que somam apenas 110 no país (CASTRO; VILARIM, 2013). Nas últimas duas décadas, computação nas escolas (quando existente) resume-se a escrever planilhas administrativas no Excel, e isso é limitante para a educação dos jovens, e ruim para a economia. A geração jovem deveria ser educada não somente na aplicação e no uso de tecnologia digital, mas também em como trabalhar nos seus princípios fundamentais (JONES, 2013). O primeiro problema do ensino de programação para crianças e jovens é que simplesmente não há professores suficientes com conhecimento e entendimento o bastante na disciplina, para dar um currículo rigoroso de ciência da computação e tecnologia da informação em escolas (FURBER, 2012).

Apesar da falta de profissionais qualificados, no Reino Unido foi fundado o Code Club, um clube extracurricular de ensino de programação para crianças de 9 a 13 anos, do qual os professores são voluntários. Hoje, é um projeto mundialmente conhecido e apoiado e já atendeu mais de 150 mil crianças pelo mundo todo. Porém, eles também encontraram, desde o começo, um déficit de professores com conhecimentos suficientes para ensinar programação a crianças e jovens (FURBER, 2012).

O Code Club ensina crianças de 9 à 13 anos a programação, porém, de acordo com a teoria de Piaget de desenvolvimento cognitivo, as crianças do ensino primário

[1]Information, Communication and Techonology - Informação, Comunicação e Tecnologia

estão entrando na fase a qual eles estão começando a desenvolver o pensamento lógico e habilidades de solucionar problemas (GILLANI, 2013), e com isso mostra-se interessante e proveitoso que o ensino de programação seja feito desde criança.

Nesse trabalho é explanado sobre essa importância do ensino de progamação ser contínuo, ter um início na infância ainda, e ter continuação. O Code Club se mostra uma ótima metodologia para o ensino de programação, mas por ser muito restrito em relação a faixa etária. Esse trabalho traz argumentos que mostram o quanto se torna imprescindível nessa era tecnológica que haja esse seguimento constante no aprendizado de lógica de programação e algoritmos.

Quando o Code Club foi desenvolvido, foi feita uma análise de qual seria a variação de idade ideal para o objetivo que eles visavam e foi definido que o projeto seriam de crianças de 9 à 13 anos (SMITH; SUTCLIFFE; SANDVIK, 2014). A seção 2.2 fala sobre o ensino de programação de 9 a 13 anos e sobre o método Code Club, que é uma opção de aprendizado de programação para essa idade - que é o foco desse trabalho. A seção 2.2.1 faz a continuação desse estudo, e mostra o ensino de programação dos 13 aos 17 anos, com exemplos de como as escolas de ensino médio de diversos países lidam com a tecnologia e como fazem uso da programação para os jovens: muitos países inclusive, incluíram nos seus currículos escolares a disciplina de algoritmos e/ou programação (GOV.UK, 2017) (BORNELI, 2015).

Com essa visão dos benefícios do Code Club e da programação para crianças e jovens, esse trabalho faz um estudo de caso com alunos de uma escola municipal de Palmas, capital do estado do Tocantins, utilizando a metodologia Code Club para analisar se há bons resultados quando compara-se o antes e o depois dos alunos nas aulas do projeto. Esse estudo usa como base teórica avaliação de interfaces, que é explanada na seção 2.3, objetivando a análise e comparação de resultados obtidos com questionários no projeto.

Espera-se, ao final do estudo de caso, colher dados suficientes para responder às seguintes questões: Ao final da aplicação do método, os objetivos de aprendizagem foram atingidos? Esse método motivou e interessou os alunos a aprenderem lógica de programação? Ele facilitou o aprendizado da lógica? Quais os benefícios trazidos pelo método?

2 FUNDAMENTAÇÃO TEÓRICA

2.1 A importância do ensino de programação contínuo desde a infância

> "Quando nós tivemos linguagem, não aprendemos apenas como ouvir, mas como falar. Quando tivemos texto, não aprendemos apenas como ler, mas como escrever. Agora que temos computadores, estamos aprendendo a usá-los, mas não como programá-los. (RUSHKOFF, 2012, s. p.).

As crianças do século 21 estão cercadas por tecnologia: em casa, na escola, casa de amigos ou familiares, etc. Consequentemente, torna-se quase impossível impedir que elas tenham contato com tantos avanços. Vários estudiosos acreditam que o ensino de linguagens de programação é o futuro da educação. Aprender programação, não se trata apenas de ter o conhecimento de programar computadores, ou então de saber várias linguagens: ela também altera e aumenta a percepção da pessoa em relação ao mundo, por saber lidar com comandos específicos. Em uma era que é baseada em tecnologia, códigos e algoritmos, não são somente as pessoas que almejam uma carreira tecnológica que devem saber/dominar a linguagem dos computadores: tornou-se imprescindível que qualquer pessoa que quiser projetar algo para celulares, computadores e notebooks, por menor que seja, saibam lidar com programação.

Celulares, notebooks, tablets, computadores, videogames, jogos de realidade virtual, relógios digitais (...) são objetos que tornaram-se comum no dia-a-dia de todos, particularmente de crianças e jovens, trata-se de um mundo extremamente diferente daquele de 10, 20 anos atrás. Porém, assim como aulas de matemática e inglês, a programação está se tornando cada dia mais uma das habilidades essenciais desse século, quando se trata de um currículo apresentável e conhecimentos necessários para bons empregos.

"Nós fomos ensinados física na escola porque nosso mundo é governado pelas leis da física. Agora nós estamos em uma era digital, e seria estranho não ensinar crianças como programar computadores e a criar aplicações e ferramentas que nós usamos todos os dias." disse Clare Sutcliffe, fundadora do Code Club, explanado na seção 2.2.

Ao ouvir a palavra "programador" ou "códigos de programação", muitas pessoas nem sabem qual sua finalidade, ou pra que servem, apenas imaginam uma pessoa que fica horas em frente a uma tela de computador digitando códigos ininterruptamente. E mostra-se importante ressaltar, que o objetivo desse trabalho não é que as crianças se tornem programadoras profissionais, e sim apenas que seus cérebros sejam desenvolvidos e trabalhados para um outro lado: as crianças de hoje em dia já estão submersas em tecnologia, indiscutivelmente, então por que não aproveitar esse cenário e trabalhar o

raciocínio lógico delas, por exemplo?!

E essa é a primeira e principal razão: fazer com que o cérebro infantil possa desenvolver o raciocínio lógico. Programação se assemelha muito a matemática: primeiramente você precisa analisar e entender o problema. São etapas: só é possível começar a pensar em como montar a solução, uma vez que houver o entendimento de todos os elementos envolvidos no problema. Com "o que tem que ser feito" em mente é possível começar a trabalhar na solução. E na programação, o modo como se resolve um problema, pode fazer uma grande diferença. É possível que um código seja escrito com a mesma finalidade, com 20 ou 100 linhas, depende do quanto a sua intuição e lógica conseguem "afunilar" o problema em questão.

Ou seja, o ensino da programação auxilia na solução de problemas, pois, ao programar é necessário aprender a estruturar o pensamento de forma clara e objetiva, para que a máquina possa entender o passo-a-passo, e executar corretamente as ordens do programador. Tal exercício diário, desenvolve habilidades utilizadas para solucionar problemas corriqueiros com mais naturalidade e rapidez.

O objetivo é que haja um desenvolvimento de competências fundamentais, como o raciocínio lógico, e a resolução de problemas; o estímulo a criatividade, e ao pensamento crítico, e ainda uma melhora no desempenho escolar em disciplinas essenciais, como ciências, matemática e inglês. Outro motivo é que, nos próximos anos, será tão importante "falar a língua" das máquinas como é fundamental ler e escrever nos dias de hoje.

Na era da tecnologia, um dos principais público-alvo das grandes indústrias de tecnologia, são as crianças e jovens: o acesso a jogos e aplicativos se tornou muito fácil e comum por eles. Apesar de os adultos, (principalmente eles, por não terem tido a quantidade de contato com tecnologia semelhante às crianças de hoje) acharem que programar é uma coisa extremamente complicada, para as crianças mostra-se mais fácil esse aprendizado.

"Imagine mandar um estudante a um colégio no qual não se ensina a fotossíntese ou o sistema digestivo. Ninguém chamaria isso de educação. Usam computadores com mais frequência do que seu sistema digestivo, e merecem que lhes expliquem como ambos funcionam" (Partovi, South Summit). O pensamento de Partovi acredita que o aprendizado da programação seja tão importante quanto o ensino de biologia: são exemplos de atividades que acontecem diariamente ao redor das pessoas e é imprescindível que se entenda tanto o porquê da existência de um intestino, quanto o porquê da existência de um *loop*. Claramente, o currículo moderno de uma escola de ensino médio deveria refletir o crescimento dessa importância (GAL-EZER et al., 1995).

Não faz diferença qual área a criança estudará, trabalhará ou se seguirá carreira em tecnologia e informação. O objetivo de estudar programação desde novo, não é que ele se torne um programador profissional, o objetivo é que ele saiba, aprenda, entenda, e isso irá beneficiá-lo futuramente. Além disso, por meio desse aprendizado, diversas ha-

bilidades serão trabalhadas: comunicação, resolução de problemas, desenvolvimento da criatividade, etc. A criança já passa, de qualquer modo, um tempo significante na frente das telas, então com os recursos apresentados nesse trabalho, será possível que as crianças tenham maneiras de serem engajadas em atividades tecnológicas que as beneficiam e as divertem, porém, usando de maneira voltada ao treino da lógica e linguagem de programação (dependendo da idade).

Mas se for o caso, de no futuro a criança ter interesse em realmente se tornar um programador profissional, ela já vai estar em destaque em relação a outros candidatos. Nos próximos 10 anos, irão existir 1,4 milhões de vagas em ciência da computação somente nos EUA e somente 400 mil profissionais se qualificarão para elas, ou seja, haverá um déficit de 1 milhão de profissionais [CODE.ORG, 2014]. É claro que no contexto do aprendizado infantil esse argumento não é o principal, mas existem também os benefícios profissionais sim, o fato de ter 10 anos de experiência a mais do que o seu concorrente, faz toda a diferença. E é fato que muitas profissões já estão se beneficiando da programação como uma habilidade complementar, que garante, através do uso de dados e automatizações, acurácia e produtividade extra na realização das suas tarefas. E além disso, a demanda do mercado de tecnologia por bons programadores só cresce, em pouco tempo, saber programar no século 21, será tão importante quanto saber ler e escrever no século 20, o que expandirá ainda mais as oportunidades para profissionais da área.

Pode-se evidenciar também que, geralmente, aprender algo novo quando se é jovem é mais fácil, é como o aprendizado de um segundo idioma durante a infância e adolescência. A criança e o adolescente têm naturalmente, a tendência de aprenderem algo novo, mais facilmente do que adultos, e isso se dá devido a janela de oportunidade[1], como é mostrado na figura 2.1. Além disso, aplicativos e outras ferramentas de aprendizagem utilizam uma linguagem bastante visual e interativa para ensinar. Isto possibilita, literalmente, aprender brincando. Programar é criar todo o tipo de coisa possível e ainda não imaginadas no computador, ou seja, a programação além de tudo ainda estimula a criatividade, por ser um ótimo estímulo ao potencial criativo das crianças, uma eficaz ferramenta que os incentiva, de maneira divertida, a explorarem suas capacidades.

Para a fundadora do Playground da Inovação, Fernanda Furia, “a programação é um dos passos para a alfabetização digital e será cada vez mais uma habilidade para toda a vida, pois desenvolve capacidades como resolução de problemas, raciocínio lógico, relação entre causa e consequência, raciocínio matemático, linguística, etc”. E João Vilhete, (professor e pesquisador da Unicamp), diz que ensinar programação para as crianças é o mesmo que “ensinar a pensar”. Para ele “ao aprender a programar, crianças e jovens desenvolvem a capacidade de lidar com problemas, já que colocam em prática uma série

[1]Estudos recentes na área de neurociências comprovam que há um prazo biológico para que determinadas tarefas sejam executadas com eficiência. Esse prazo, as janelas de oportunidade, é um momento crucial para a aprendizagem de um segundo idioma (ALBUQUERQUE, 2011).

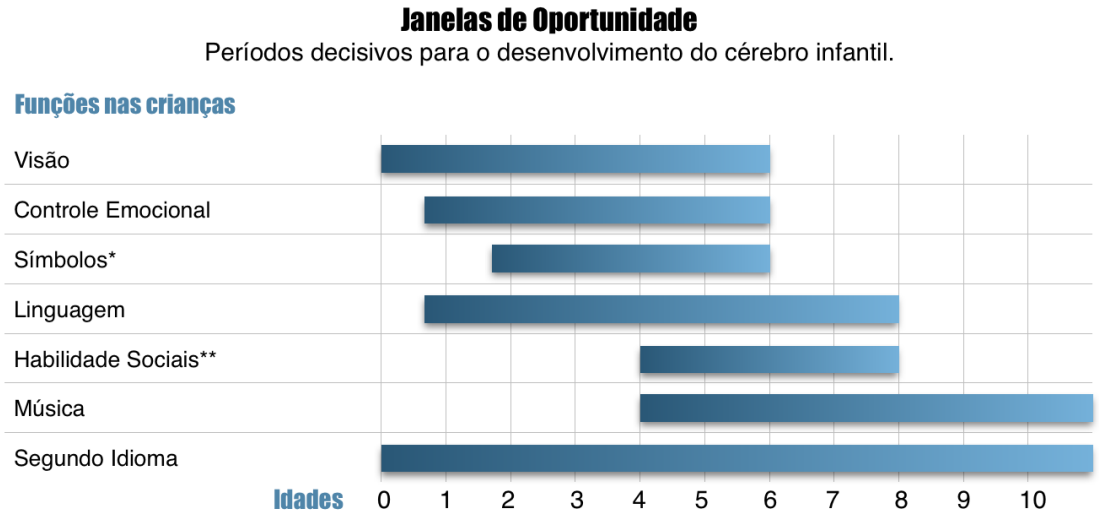


Figura 2.1 – Idades com janelas de oportunidades abertas (FóZ, 2015)

de teorias que são ensinadas em física, matemática e química, por exemplo."

Piaget provou que a criança, desenvolve diferentes maneiras de aprender, desde pequena, até mesmo as que não tenham ido a escola. Um exemplo de conceito que a criança pode aprender sem ser explicitamente ensinada, ou que ela perceba que está sendo ensinada algo mais importante e difícil do que parece, é o de volume: um copo estreito e alto pode conter a mesma quantidade de líquido que um copo largo e baixo. "O aluno, ao usar as linguagens de programação, transforma seu conhecimento em procedimentos, ou seja, descreve todos os passos necessários para atingir certo objetivo, para atingir a resolução de certo problema" (ALMEIDA, 1999). Dessa forma, segundo Almeida, a partir do momento que a criança têm um conjunto de instruções pra seguir, a mesma consegue chegar a solução de um problema, ainda que não tenha sido ensinado a ela do modo convencional.

2.2 O método Code Club

O Code Club[2] é uma rede mundial gerenciada completamente por voluntários e educadores de clubes de programação extracurriculares gratuitos, com o objetivo de ensinar programação às crianças. O método usado são simples lições (baseadas em um passo-a-passo da metodologia própria e unificada do Code Club) que desenvolvem as habilidades das crianças a fim de aprenderem scratch, HTML, CSS, python, e ainda criarem jogos, animações e web sites. A principal motivação do clube é então, inspirar crianças com um senso de diversão e conquista na programação, e a critividade digital. O clube objetiva a introdução de conceitos de programação gradualmente para que a criança aprenda, mas

[2]https://www.codeclubworld.org/

Figura 2.2 – Mapa dos locais onde há Code Clubs (FOUNDATION, 2017)

aos poucos, permitindo assim, que elas construam seus leques de conhecimento de maneira incremental. A intenção é que elas estejam à vontade no clube, e que assim, seja possível que compreendam que são os responsáveis pelas ações do computador: que eles devem dar os comandos que esperam que aconteçam.

"A missão do Code Club é dar a todas as crianças do mundo uma chance de aprender a programar, através de uma rede de voluntariado que cria clubes de programação e materiais didáticos exclusivos" (FOUNDATION, 2017).

Visto que as crianças de escola primária estavam sendo abandonadas pela provisão de educação atual do Reino Unido (ICT), a designer Clare Sutcliffe e a desenvolvedora de interface Linda Sandvik resolveram fazer algo a respeito, e em 2012, elas fundaram o Code Club no Reino Unido (SMITH; SUTCLIFFE; SANDVIK, 2014). Esse projeto voluntário foi fundado em 2012 para ensinar programação nas escolas do Reino Unido, e foi um sucesso desde o começo: no primeiro dia já houve mais de mil inscritos. Em 2013, o Code Club começou a operar no Brasil, pouco tempo depois já contava com 502 clubes ativos nas 5 regiões do país, e já alcançou mais de 150.000 crianças no mundo, sendo sete mil só no Brasil. No Reino Unido são englobados 900 clubes operando em 120 escolas (GARNER, 2013). O clube é distinto porque é direcionado exclusivamente para crianças da escola primária, em um escopo nacional e cada vez mais internacional. É uma iniciativa que responde ao apelo de como atualizar o currículo computacional (BARR; STEPHENSON, 2011).

Há mais de 10.000 clubes criados em 140 países, e os módulos já foram traduzidos para mais de 30 línguas diferentes. O projeto foi testado com crianças em vinte escolas do Reino Unido e a média de aceitação e aprovação foi 9,2. Com sua grande expansão

mundial, tornou-se uma rede mundial de atividades extracurriculares gratuitas e recebeu patrocínio de empresas como a ARM e a Google. Na figura 2.2, é possível ver o alcance do Code Club pelo mundo.

Há diversos programas de estudo existentes para ensinar à crianças o básico de computadores e programação (BRENNAN; BALCH; CHUNG, 2011) (COMPUTER..., 2012), mas nenhum era adequado para o Code Club. O currículo publicado geralmente diz que o ensinamento será feito durante um programa de estudo baseado em sala de aula, com um professor, durante mais de um ano. Para isso, o Code Club é um clube de sessões de uma hora após a escola, por dez semanas. A presença em clubes após a escola é mais errático. Os programas já existentes focam nas habilidades e comportamentos que as crianças deveriam adquirir, enquanto a intenção do Code Club é criar um senso de diversão e emoção nas crianças participantes e manifestar um desejo de seguir com computação, com habilidade de desenvolvimento como preocupação secundária.

No Code Club foi adotado uma pedagogia baseada em projetos (BLUMENFELD et al., 2011), onde cada projeto cria uma aplicação de *scratch*[3] específica, como um jogo ou um pacote de desenho. Os projetos são amplamente semelhantes aos outros desenhados para um grupo de idade similar (PROJECT, 2013), apesar dos projetos anteriores serem menores. A atividade do primeiro bloco consiste em nove projetos, e cada projeto é esperado que tome uma ou duas sessões. Cada projeto é dividido em séries de passos, os primeiros passos são dirigidos com um escopo restrito e visam dar à criança uma aplicação básica de trabalho com cerca de 20 minutos. Passos subsequentes extendem o projeto básico. Alguns passos são direcionados e adicionam peças específicas de funcionalidade adicional. Alguns passos descrevem uma extensão do projeto, mas deixam a implementação para o pupilo. Outros são mais abertos, convidando os pupilos à explorar como eles podem customizar ou extender a aplicação. A medida que os projetos progridem nos termos, o nível de direcionamento diminui e os pupilos são cada vez mais solicitados a desenhar e implementar as suas próprias ideias e extensões. O primeiro termo conclui com um projeto final de conclusão onde os pupilos desenham e constroem o próprio jogo, desenhando junto o que eles aprenderam nos projetos anteriores. Todos os passos do projeto incluem prompts específicos para os pupilos testarem seus projetos e salvarem seus trabalhos. Os projetos fazem uso extensivo dos cartões do scratch (SCRATCH..., 2012).

O segundo bloco dos projetos de scratch é organizado em três grandes projetos (um monstro animado, um conjunto de intrumentos musicais com gravação e playback, e uma plataforma de jogo), cada um destinado a ser completado com 3 ou 4 semanas. Esses projetos contém um componente de desenho substancial, com pelo menos uma sessão dedicada aos estudantes desenharem seus projetos. Esses projetos contém pequenas amostras de código para as crianças copiarem. Ao invés, é esperado que eles criem seus

[3]Scratch é uma linguagem de programação visual que possibilita a criação de jogos e animações pela própria criança, por meio de combinações de blocos lógicos que se "arrastam".

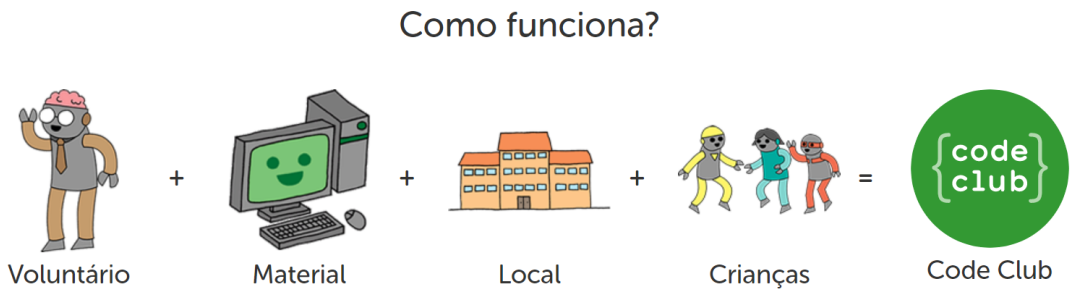


Figura 2.3 – Passos para criação de um Code Club (CLUB, 2017)

próprios *scripts*, solicitado pelo que eles já têm alcançado, ou por amostras sugeridas nos projetos. A abordagem do projeto base tem muitas vantagens para o Code Club. Primeiro, ela foca no trabalho em problemas autênticos. Isso motiva as crianças a engajarem com o material e consequentemente, aprenderem as habilidades de programação que elas precisam. Segundo, os pupilos produzem um artefato discreto, o qual dá as crianças um senso de conquista, assim como a habilidade de mostrar o produto do seu trabalho para amigos e família (projetos do scratch são facilmente compartilhados via website). Terceiro, ele permite muito mais flexibilidade no atendimento: o material permite que as crianças percam sessões em diferentes dias e ainda sejam participantes ativos quando voltarem. E por fim, ele permite a diferenciação entre os pupilos em diferentes habilidades, com a velocidade mais adequada para cada um pelos seus projetos. Há um livro com todos os projetos do *scratch* de uma maneira lúdica e fácil para as crianças acompanharem (CROSS; GARDNER, 2018).

Para a criação de um Code Club, o primeiro passo é alguém que saiba programação de computadores se cadastrar como voluntário no site do Code Club World, e criar um Code Club. Para que essa criação seja possível, é imprescindível que haja um local adequado pré-definido: os lugares mais utilizados são escolas, bibliotecas e centros comunitários, mas o necessário é uma sala de informática com acesso à internet de qualidade. Assim que for feito o registro no Code Club, o voluntário tem acesso ao material das aulas. A função principal do voluntário é criar um ambiente agradável de ensino para que seja possível o desenvolvimento do projeto e estabelecer uma rotina e regras com as crianças - contando que cada aula do cronograma do Code Club acontece uma vez por semana, com duração de uma hora, durante um período de dez semanas. Nas cidades em que já existe um Code Club ativo, é possível também tornar-se voluntário de um clube já existente. Outra maneira de se voluntariar - indiretamente - com o projeto, é se envolver com as traduções de material didático e vídeos do projeto. A figura 2.3 mostra uma captura de tela do site do Code Clube, mostrando o passo-a-passo para a criação de um Code Clube.

Quando o Code Club foi fundado, havia diversos programas de estudo existen-

Figura 2.4 – Página inicial de acesso dos voluntários às lições (FOUNDATION, 2017)

tes para ensinar crianças o básico de computação e programação (BRENNAN; BALCH; CHUNG, 2011) (COMPUTER. . . , 2012), mas nenhum servia para o Code Club, pois os currículos geralmente eram baseados no fato do ensino ser feito durante um aula formal, em salas de aula com um professor em um programa de estudo de mais de um ano (SMITH; SUTCLIFFE; SANDVIK, 2014).

Em 2013, no Brasil ja havia mais de 200 Code Clubs ativos: "Nosso objetivo é que cada escola do Brasil tenha um clube de programação onde as crianças possam aprender informática. Temos quase 200 mil escolas no Brasil, temos muito trabalho pela frente, mas acreditamos que isso seja possível!" (Everton Hermann, fundador do Code Club Brasil). A filosofia do clube é aprender por meio da diversão, criatividade e que a aprendizagem seja feita pela descoberta. "O clube do código brasileiro, surgiu em 2013, e possui a mesma metodologia: os cursos 1 e 2, utilizam o scratch como ferramenta para ensinar programação. O curso 3 é uma introdução ao desenvolvimento web e utiliza HTML e CSS, e o curso 4 utiliza python." (GERALDES, 2014)

O método do Code Club é pré-definido e já tem a ordem e as lições prontas. O curso é fragmentado em scratch, HTML e CSS, e python, cada um em dois módulos, e cada módulo com seis lições - mais as extras. Scratch é uma linguagem de programação visual que possibilita a criação de jogos e animações pela própria criança, por meio de combinações de blocos lógicos que se "arrastam", como citado anteriormente. O HTML, juntamente com CSS, é a linguagem padrão para criar e estruturar sites na Internet e servem para formatar o visual aos elementos Html, enquanto python é uma linguagem textual.

Módulo 1: Scratch

O curso 1 e 2 são mais básicos e ensinam a base da programação utilizando scratch,

Tabela 2.1 – Scratch - Módulo 1

Banda de Rock	As crianças aprendem como codificar os seus próprios instrumentos musicais.
Perdido no espaço	As crianças aprendem a programar sua própria animação.
Caça fantasmas	As crianças fazem um jogo de caça fantasmas .
Robô falante	As crianças programam um robô falante.
Lousa mágica	As crianças fazem um programa de pintura.
Corrida de barco	As crianças fazem um jogo que - usando o mouse - navega até uma ilha deserta.

que é um ambiente visual de aprendizado de linguagem de programação que permite o usuário criar projetos interativos e rico em mídia (MALONEY et al., 2010), que foi criado com a ideia de tornar possível que iniciantes rapidamente criem programas sem ter que aprender como escrevê-los sintaticamente correto. A intenção é motivar outros aprendizados por meio de experimentos divertidos e criação de projetos, como animações interativas, jogos, etc (Maloney et al, 2010). O scratch usa uma interface de usuário gráfica, a qual previne a necessidade de memorizar a linguagem de programação, e ao invés disso, permite para os usuários que explorem suas diferentes funções. É feito de códigos fragmentados chamados de blocos, que podem ser arrastados na área para fazer programas. Os blocos são organizados em diferentes categorias como movimento, controle, sensor, etc para um uso mais fácil. Diferentes tipos de blocos tem diferentes tipos de cores e formas. O scratch é a linguagem educacional de programação mais popular usada por milhões de novos aprendizes em salas de aula e em casas espalhadas pelo mundo. Ao arrastar blocos coloridos de código, crianças podem aprender conceitos de programação de computadores e fazer jogos e animações. A última versão, scratch 2, traz a linguagem diretamente pro seu navegador sem a necessidade de baixar o software (SCRATCH. . . , 2017).

O ensino do scratch é dividido em dois módulos, o primeiro está descrito na Tabela 2.1, tem duração de um trimestre e introduz a programação de computadores usando a ferramenta scratch, e conta com seis lições - mais as extras.

O segundo módulo está descrito na Tabela 2.2, e nele é dado início ao segundo trimestre, o qual ainda utiliza a ferramenta scratch, mas de maneira mais avançada. Neste módulo os alunos utilizarão seus conhecimentos para criar seus próprios projetos. Ele também conta com seis lições - mais as extras.

O foco deste módulo é desenvolver habilidades de pensamento criativo e codificação que permitam que as crianças desenvolvam seus próprios projetos. Assim, a primeira proposta será conceber um personagem que seja feito exclusivamente por ela, um monstro, por exemplo. O pupilo tem que pensar e inventar aspectos dele, como aparência, movimentos e etc, antes de iniciar a programação propriamente dita. As codificações serão auxiliadas por cartões de auxílio, que os alunos juntaram para desenvolverem uma solução

Tabela 2.2 – Scratch - Módulo 2

Jogo da Memória	As crianças aprendem a criar um jogo da memória para repetir uma sequência aleatória de cores.
EsquivaBol	As crianças aprendem a criar um jogo de plataforma que se esquiva das bolas em movimento.
Jogo da Tabuada	As crianças aprendem a criar um jogo de tabuada, no qual têm-se que acertar o máximo possível de respostas em 30 segundos.
Capturando Bolinhas	As crianças aprendem a criar um jogo que captura bolinhas coloridas com o lado certo de um controle redondo.
Guerra dos Clones	As crianças aprendem a criar um jogo para salvar a terra de monstros espaciais.
Crie seu próprio mundo	As crianças aprendem a criar um jogo de aventura em um mundo aberto.

Tabela 2.3 – HTML e CSS - Módulo 1

Feliz aniversário	As crianças começa sua jornada pelo HTML e CSS aprendendo a fazer seu próprio cartão de aniversário customizado.
Conte uma história	As crianças aprende como criar a sua própria página na web para contar uma história, anedota, piada ou poema.
Procurado!	A criança aprende como fazer o próprio cartaz.
Receita	As crianças aprende a criar uma página na internet para sua receita favorita.
Carta misteriosa	As crianças aprende a fazer uma carta como se cada palavra tivesse sido tirada de um lugar.
Projeto mostruário	As crianças cria um mostruário de seus projetos HTML e aprende sobre links e recursos embutidos.

criativa na criação do monstro.

Módulo 2: Html e CSS

Na sequência, é iniciado o trabalho de HTML e CSS: Desenvolvimento Web - Módulo 1. Esse módulo de web permite aos alunos montarem suas próprias páginas na internet. A Tabela 2.3 mostra quais são as aulas do módulo.

Em sequência, no módulo 2 de desenvolvimento web é introduzido posicionamento, efeito gradiente, animações em css entre outros; já criando websites com HTML e CSS. Na Tabela 2.4, é possível visualizar as etapas das aulas do módulo 2.

Módulo 3: Python Após a conclusão do módulo de HTML e CSS, começa-se o estudo de python, no módulo 1: comandos básicos e primeiros passos em python são ensinados para as crianças. A Tabela 2.5 mostra quais são as aulas desse módulo.

E na sequência, por fim, inicia-se o módulo 2 de python - no qual a ideia das crianças sobre programação já está bem mais clara e avançada. A Tabela 2.6 mostra a sequência de aulas lecionadas nesse módulo.

Tabela 2.4 – HTML e CSS - Módulo 2

Construa um robô	As crianças aprendem sobre posicionamente CSS construindo seu próprio robô.
Adesivos!	As crianças começam a ver gradientes lineares e radiais em CSS. Elas também vão aprender mais sobre bordas e posicionamento.
Nascer do sol	As crianças aprendem a animar uma cena simples usando CSS. Elas vão usar a regra CSS @keyframes para animar várias propriedades de imagens e divs.
Cômodos interligados	As crianças são introduzidas a várias páginas web interligadas no mesmo projeto, cada uma com seu próprio arquivo CSS.
Revista	As crianças aprendem a criar um layout de duas colunas. Elas também vêm vários códigos HTML e CSS que foram aprendidos em outros projetos.
Arte em pixels	As crianças criam um editor de arte em pixels. Elas são introduzidas ao javaScript para alterar a cor dos pixels. Elas também aprendem a usar tabelas HTML para criar uma malha quadriculada de pixels.

Tabela 2.5 – Python - Módulo 1

Arte em ASCII	As crianças aprendem como executar um programa em python, e como exibir texto na tela do computador.
O Ano 2025	As crianças aprendem a escrever um programa que dirá quantos anos você vai ter em 2025.
Quiz	As crianças criam um quiz de desafio para os amigos.
O poder da Tartaruga	As crianças aprendem a usar a tartaruga para desenhar figuras e padrões fantásticos.
Porta da Fortuna	As crianças aprendem como fazer um jogo de adivinhação para descobrir qual porta esconde o prêmio.
Gerador de Cumprimentos	As crianças aprendem a usar listas para armazenar vários dados em 1 variável.

Tabela 2.6 – Python - Módulo 2

Ensinamento de Tartarugas	As crianças aprendem funções e passagem de parâmetros, criando e usando funções para desenhar formas e padrões personalizados.
Conversor de expressões	As crianças aprendem a usar dicionários como uma forma de armazenar dados.
Pokedex	As crianças aprendem a fazer uma interface gráfica de usuário (GUI), através da criação de uma Pokedex para mostrar informações de um Pokémon.
Mensagens Secretas	As crianças aprendem iteração sobre uma string de texto, que pode ser usada para criar uma cifra de César.
Minecraft 2D	As crianças aprendem alguns aspectos de desenho gráfico e inteligência artificial, fazendo um clone muito básico do Minecraft 2D.
RPG	As crianças aprendem desenvolvimento de jogos por meio da criação de um jogo de labirinto no estilo RPG.

2.2.1 O ensino de programação dos 13 aos 17 anos

Uma discussão foi iniciada por Jeanette Wing sobre o papel do pensamento computacional sobre todas as disciplinas (WING, 2006), e ali foi iniciado um engajamento com questões primordiais sobre o que é ciência da computação e como poderia contribuir para soluções de problemas corriqueiros. Ela argumentou que todo e qualquer avanço tecnológico permite qualquer pesquisador, de qualquer área a visionar novas estratégias que solucionem seus problemas nos seus respectivos campos.

A computação tornou possível saltos enormes na inovação e facilita nossos esforços para resolver problemas de importância mundial, como a prevenção ou cura de doenças, por exemplo. Esses avanços, no entanto, têm a necessidade de indíviduos tecnologicamente educados que possam manter o suporte computacional, e que possam ser as pessoas que vão descobrir essas curas. Já não é mais suficiente esperar até que os estudantes estejam na faculdade para introduzir esses conceitos. Todos os estudantes dos dias de hoje vão viver uma vida muito influenciada pela tecnologia, e muitos vão trabalhar em campos que envolvem a necessidade de computação. Eles devem começar a trabalhar com métodos de soluções de problemas com ferramentas e algoritmos computacionais no ensino médio (BARR; STEPHENSON, 2011).

Nessa idade, os jovens se encontram no ensino médio, e na seção 2.1, foi mostrado que é importante esse ensino de computação nessa idade. Nessa idade é importante que consiga-se ver que o adolescente já tem uma base montada, e já tem o conhecimento de lógica de programação, mas ver também que ele precisa trabalhar nas linguagens de programação, para melhorar sua técnica de desenvolvimento.

Aprender a programar é considerado difícil. Cursos de programação em universidades geralmente têm as maiores taxas de desistência e são frequentemente consideradas

as mais dificeis (Yadin, 2011). E esse é um fator importante para mostrar a necessidade da programação para adolescentes.

2.3 Avaliação de Interfaces

A avaliação é um dos principais processos do design de sistemas interativos, onde idealmente, ela deve ocorrer durante o ciclo de vida do design e seus resultados utilizados para melhorias gradativas da interface. Avaliar algo trata-se de determinar o seu valor, apreciar, julgar e ponderar o objeto em análise. É chamada de avaliação de IHC aquela atividade profissional especializada que visa julgar a qualidade de interação que um sistema ou artefato computacional oferece aos seus usuários.

O objetivo de uma avaliação de interface não é executar testes experimentais muito extensivos durante o processo inteiro de design, mas sim a utilização de técnicas informais e analíticas (ROCHA; BARANASKAS,). Ou seja, quando lida-se com avaliação de interfaces de interação homem computador, o objetivo é que seja feita uma análise criteriosa da qualidade da experiência dos usuários ao interagirem com sistemas computacionais, sendo que essa análise é feita baseada em conhecimento técnico. Se trata de uma atividade profissional, e não de uma emissão de opinião baseada em preferências ou conjecturas pessoais - usuários comuns sempre têm opiniões e julgamentos sobre a qualidade dos sistemas com que interagem, porém não são avaliações de IHC.

Essas avaliações são feitas quando se revisa, experimenta ou testa uma ideia de design, software, produto ou serviço visando descobrir se ele atende a alguns critérios. Em geral, é feita a avaliação com a finalidade de explorar a vontade dos usuários e os problemas que eles experimentam, uma vez que quanto mais bem informados os designers estiverem sobre seus usuários, melhor serão os design de seus produtos, mas a tabela 2.7 especifica mais ainda, mostrando os principais objetivos de se fazer as avaliações.

Em geral, os critérios das avaliações são se o sistema é aprendível, efetivo e adaptável, mas isso é algo que pode variar muito dependendo da característica do objetivo. A preocupação quando lida-se com avaliações não se trata apenas de algo superficial, como por exemplo, a imagem adequada para o ícone, ou o significado do ícone, mas também se o sistema é adequado para o seu propósito, se ele é agradável e envolvente, por exemplo. “A avaliação está intimamente ligada às outras atividades do design de sistemas interativos: o entendimento, design e antecipação” (BENYON, 2010), ou seja, primeiramente, precisa-se entender o que está sendo avaliado, os seus parâmetros e suas características importantes, e é necessário que haja essa compreensão antes da existência de problemas nas interfaces.

É possível dividir em três partes os artefatos que são avaliados:

1) Os aspectos cognitivos e funcionais relativos à realização de tarefas apoiadas pelo sistema - ou seja, é fácil de aprender? é rápido? pode-se confiar? é possível reverter

Tabela 2.7 – Principais objetivos da avaliação de interfaces

Principais objetivos da avaliação de interfaces:
1. Conferir o entendimento dos projetistas sobre as necessidades e as preferências dos usuários.
2. Apurar como uma interface afeta a forma de trabalhar dos usuários (a forma de trabalhar, de se divertir, etc.).
3. Equiparar alternativas de projeto de interface.
4. Encontrar problemas de interação ou de interface.
5. Averiguar se foram alcançados objetivos quantificáveis em métricas de usabilidade.
6. Averiguar conformidade com um padrão ou conjunto de heurísticas.
7. Arquitetar material de treinamento e de apoio, como manual de instruções e sistema de ajuda.

erros cometidos (facilmente)?

2) Os aspectos sócio-culturais do uso do artefato no local - ou seja, a tecnologia se integra fácil no ambiente físico? Causam algum tipo de problema com outras tecnologias ou com pessoas que estejam usando outras tecnologias? Pode-se prever mudanças das práticas do ambiente sócio-cultural previsto?

3) Os aspectos afetivos - ou seja, as pessoas vão gostar? Vão achar bonito e agradável?

A importância da existência da avaliação é justamente essa: objetiva-se consertar problemas antes, não depois, de um produto ser lançado, e além disso, o tempo para colocar o produto no mercado diminui. Quando há essa avaliação durante o processo as energias da equipe são direcionadas, visto que o problema real é encontrado, sabe-se exatamente o que precisa de atenção para correção. Outro ponto válido de ressaltar na importância da avaliação é que engenheiros, no geral, codificam o produto, mas é preciso que haja testes para que o resultado chegue o melhor possível no usuário final. Quando um sistema não passa pela etapa de avaliação, há uma grande chance de se tornar uma interface de baixa qualidade, e interfaces de baixa qualidade:

1) Requerem treinamento excessivo.
2) Desmotivam a exploração.
3) Confundem os usuários.
4) Induzem os usuários ao erro.
5) Geram insatisfação.
6) Diminuem a produtividade.
7) Não trazem o retorno de investimento previsto.

São necessários diferentes tipos de avaliação em diferentes estágios do design. Ao lidar com estágios mais iniciais - onde ideias estão sendo exploradas e tentadas - testes bastante informais são suficientes, no geral. Cita-se um exemplo que após uma sessão

de *brainstorming*[4] onde o objetivo é explorar diferentes metáforas, o conjunto inicial de opções certamente estará bem reduzido, já que várias são descartadas logo no começo. Porém outras vezes, principalmente, em estágios um pouco mais avançados do processo, avaliações mais formais devem ser planejadas. Os fatores determinantes de um plano de avaliação incluem (Nielsen, 1993; Hix and Hartson, 1993; Preece et al., 1994; Schneiderman, 1998):

- o estágio em que o design se encontra (inicio, meio ou fim);
- o quão pioneiro o projeto é (se ele é bem definido ou exploratório);
- qual é a quantidade esperada de usuários;
- o quão crítica a interface é (um sistema de controle de tráfego aéreo é muito mais crítico do que um sistema de orientação de um shopping, por exemplo);
- qual é o custo do produto e o orçamento que foi alocado para o teste ;
- qual é o tempo que se tem disponível;
- qual é a experiência dos designers e dos avaliadores.

Muitos autores tratam a atitude de não efetuar forma alguma de avaliação irresponsável, e a consideram imprescindível. Mas vale a pena frisar que mesmo após extenuantes testes utilizando múltiplos métodos, há um grau de incerteza - por menor que seja. Ou seja, no planejamento deve-se prever métodos de avaliação contínua e reparo de problemas ao longo de todo o ciclo de vida de uma interface.

Ao lidar com um plano de avaliação, o custo dele geralmente varia de 1 a 10 por cento do custo total de um projeto, mas apesar de haver avaliações que tratam de testes simples de três dias com seis usuários para um pequeno sistema interno de contabilidade, há também grandes avaliações, detalhadas e específicas, como um novo sistema de controle de tráfego aéreo, do qual não pode haver nenhum tipo de erro, e é necessário se certificar disso quantas vezes for preciso.

Há dois tipos principais de avaliação, o primeiro é feito com um designer de interação ou especialista em usabilidade visando fazer correções de alguma forma de versão antecipada de um design - esses são os métodos baseados em especialista. O outro tipo de avaliação acarreta no recrutamento de pessoas para que façam uso de uma versão antecipada do sistema - esses são os métodos com participantes. Nessa avaliação com participantes, o intuito é que as pessoas que farão uso da versão sejam representativas,

[4]Brainstorming é uma técnica de discussão em grupo que se vale da contribuição espontânea de ideias por parte de todos os participantes, no intuito de resolver algum problema ou de conceber um trabalho criativo.

ou seja, sejam o mais perto possível do público-alvo do sistema ou do software que, em geral, são chamados de usuários finais.

Porém alternativamente, os participantes podem ser outras pessoas (outro designer, estudantes ou quem estiver por perto) que sejam convidadas para desempenhar o papel de quem fará uso do sistema. As características da população-alvo podem ser captadas por meio de personas (GREENBAUM e KYNG, 1991). Pelo fato da avaliação acontecer ao longo do processo de design de interação, em cada estágio um metódo será mais eficaz, e em outro será menos, por exemplo, mas vale ressaltar que a forma de antecipação dos sistemas futuros também é crítica quando ao que pode ser avaliado (BENYON, 2010).

Nas avaliações há coletas de dados, que podem ser questionários e entrevistas, observação em laboratório, observação em campo e por meio de registros. É possível coletar dados por meio da utilização de um ou mais dos paradigmas explanados a seguir.

2.3.1 Paradigmas de Avaliação

Ao lidar com avaliações de interface, há os paradigmas de avaliação. Um paradigma de avaliação se trata de um modelo ou uma forma de conceber a atividade e que geralmente é atrelado ao objetivo que se quer atingir com ela. Toda avaliação é guiada explícita ou implicitamente por uma série de crenças e expectativas, das quais diversas vezes elas têm origem em teorias. Um paradigma de interface homem-máquina é uma teoria, e diversas teorias apoiam o interface homem máquina. A junção dessas crenças e expectativas com as teorias a que podem se referir, constituem o paradigma de avaliação. Uma teoria, abordagem ou paradigma de Interface Homem Computador (IHC) é essencial para que se torne possível fortalecer explicações e previsões para fenômenos de interação entre o usuário e o sistema, e além disso, subsidiar resultados práticos para o design da interface de usuários (BENYON, 2010).

A base do juízo de valor são crenças e expectativas, que são em sua maioria originárias de teorias ou então de experiências empíricas. Isto é, a frase “pelo jeito hoje choverá” mostra uma crença, mas se essa crença for correlacionada com os métodos que são utilizados para executar a avaliação (emitindo assim um juízo de valor), o resultado é a constituição de um paradigma de avaliação: “As condições atmosféricas indicam que hoje choverá”.

Como mostra a tabela 2.8, existe quatro paradigmas para a avaliação de IHC: o rápido e rasteiro, os testes de usabilidade, os estudos em campo, e a avaliação preditiva. O primeiro é o “rápido e rasteiro”. Ele é uma prática comum no design de IHC, e ele é informal, pode ser feito a qualquer momento (no laboratório ou no ambiente natural mesmo), já que o que interessa nesse caso é a rapidez do *feedback*. No rápido e rasteiro, os designers contam com opiniões informais de usuários ou consultores sobre que fizeram ou estão fazendo. Em geral, é pedido à pessoas que tenham maior conhecimento em usabilidade ou pessoas que são os usuários finais, pois dessa maneira é possível checar se suas

Tabela 2.8 – Os quatro paradigmas de avaliação de interfaces

Os quatro paradigmas de avaliação:	
Rápido e rasteiro	Preza o feedback imediato.
Testes de usabilidade	Conta com registros e resolução de tarefas.
Estudo de campo	Não há interferências alguma com os usuários.
Avaliação Preditiva	Especialistas fazem a previsão, não há usuários.

ideias coincidem com as necessidades e vontades dos usuários (ROCHA; BARANASKAS,).

As avaliações fazem jus ao nome Rápido e Rasteiro, já que se trata de avaliações que são literalmente rápidas e rasteiras, as quais podem se repetir em vários pontos do ciclo de desenvolvimento. Por ser um *feedback* rápido, não se objetiva a cautela, mas sim o feedback mais imediato possível.

O segundo paradigma são os testes de usabilidade. Esse paradigma é o contrário do primeiro, as avaliações são feitas em laboratórios e são controladas pelo avaliadores e deve haver usuários típicos (que sejam usuários reais ou só usuários potenciais). Tarefas são preparadas cuidadosamente e esses usuários devem fazê-las, e é por meio do tempo necessário para responder e das respostas corretas que é avaliado o desempenho (BENYON, 2010).

Os testes de usabilidade contam com entrevistas e questionários para que seja possível descobrir o grau de satisfação dos participantes e também, os aspectos psicológicos envolvidos (PRATES; BARBOSA, 2015). Conta-se também com registros - esses registros tornam possível mais conhecimentos sobre o comportamento do usuário, trata-se de vídeos do usuário, áudios de comentários ou uma tensão muscular, por exemplo. Esses registros podem ser utilizados objetivando a predição e a explicação de certas ocorrências de desempenho, assim como para auxiliar na correção e prevenção de erros de interação.

O terceiro paradigma é o Estudos de Campo. Ele é realizado no ambiente natural dos participantes e ele tem como objetivo o aumento da compreensão de atividades dos usuários com a influência que a tecnologia exerce sobre eles. Nesse caso, os participantes são estudados sem que haja entremetimento algum, ou seja, não são feitas propostas de tarefas específicas, é a observação de como as pessoas usam e lidam com artefatos de tecnologia (BERG, 2016).

Entre as vantagens do Estudo de Campo, pode-se frisar que ele prospecta a introdução de novas tecnologias, determina requisitos para o design, decide estratégias que promovem a adoção de tecnologias e também descobre como uma tecnologia é de fato utilizada, já que não há interferência.

E por fim, o último paradigma é o da avaliação preditiva, onde não há a participação do usuário, é feita inteiramente por especialistas que conhecem os usuários típicos - personas. A avaliação preditiva é interessante devido ao seu custo bem mais baixo e por

ainda sim, ser eficaz. A avaliação preditiva é guiada por avaliações heurísticas, visando prever problemas de usabilidade.

Ou seja, nesse paradigma conta-se com a experiência dos especialistas que aplicam seu conhecimento sobre os usuários e também sobre situações que são comuns para que façam uma previsão dos efeitos que um certo produto ou tecnologia terá sobre os seus usuários. Nele, o foco é no problema e na sugestão de melhorias, e no fim, o resultado é um relatório com os problemas encontrados, juntamente com as inferências e as soluções que são propostas pelo avaliador (BERG, 2016).

2.3.2 Avaliação Preditiva

O paradigma de avaliação preditiva é o paradigma que permite que sejam coletadas opiniões de especialistas, e não de usuários, visando possibilitar a prevenção de problemas que os usuários poderiam vir a enfrentar durante a execução das suas tarefas. Dessa maneira, acaba-se identificando os problemas de interface com relação ao grau de dificuldade da aprendizagem por exploração.

Ou seja, nesse tipo de avaliação baseia-se em especialistas para aplicar o conhecimento de usuários típicos. São usados as personas - que são usuários típicos de um sistema - e também cenários para que se possa melhorar a compreensão dos objetivos de usabilidad e se aplicam modelos teóricos como GOMS[5].

Entre as vantagens da avaliação preditiva, frisa-se que ela ajuda a prever problemas de usabilidade e na maneira a qual os usuários irão realizar certas tarefas. Porém, seu ponto fraco se dá justamente por não haver envolvimento algum diretamente com usuários, ou um contexto real de uso do sistema (SANTOS, 2008).

Dentro da avaliação preditiva, há diversas técnicas que podem ser utilizadas para avaliação. A técnica que vai ser usada em dado momento varia de acordo com o paradigma que se encontra, em alguns casos uma técnica se aplica melhor, em outros pior e em alguns nem se aplica. As técnicas que podem ser usadas são:

- Observação de usuários
- Perguntas aos usuários
- Consulta a especialistas
- Testes com usuários
- Modelo de desempenho dos usuários

[5]Goals, operators, methods, selection rules

2.3.3 Avaliação Por Observação

A avaliação por observação é quando o avaliador coleta dados com usuários em situações reais de uso, querendo que os problemas que ocorreram sejam identificados (GONçALVES, 2013). Esse método de avaliação por observação é realizado com usuários reais, ao mesmo tempo em que fazem uso do sistema. São coletados dados durante esse evento e ele são, em sua maior parte, focados em problemas encontrados pelas pessoas avaliadas.

Ela pode ser realizada em laboratório ou em campo, porém se for em laboratório, em geral é realizado um roteiro para que seja possível que a pessoa consiga manter o foco nos pontos-chave do que está sendo avaliado. Todavia, se for realizada em campo, são coletados mais dados e de uma forma mais ampla uma vez que os fatores que influenciam no sistema final estão sendo testados também (GONçALVES, 2013).

Esse tipo de avaliação examina a usabilidade a partir de observações de experiências de uso, e os objetivos da avaliação dependem de quais critérios de usabilidade são esperados a serem medidos. Por exemplo, é possível medir a facilidade de aprendizado ou de recordação, a satisfação do usuário, a eficiência e a segurança no uso. É possível medir a facilidade de aprendizado medindo a quantidade de erros que os usuários cometem nas primeiras sessões de uso, a quantidade de usuários que conseguiram terminaram uma certa tarefa e quantas vezes o usuário pesquisou ou fez consultas, por exemplo.

A primeira etapa é o estudo-piloto que tem como objetivo analisar a viabilidade do sistema, e também se está apto a conduzir o procedimento inteiro e se está apto para fazê-lo bem. Visa analisar também se os recursos estão todos certos (os scripts, as entrevistas, e se os questionários estão claros). Esse estudo-piloto tem que ser feito pelo menos uma vez, mas pode - e é recomendado - ser feito quantas vezes forem necessárias para que o experimento possa se tornar melhor na qualidade.

Ao longo do processo de avaliação, a cada tarefa que é realizada por cada participante, é possível obter alguns resultados (SANTOS, 2008):

- O grau de sucesso da execução.
- O total de erros cometidos.
- A quantidade de erros de cada tipo que ocorreram.
- A quantidade de tempo que levou para a conclusão.
- O grau de satisfação do usuário

Todo o processo da avaliação por observação visa chegar no resultado da observação, que para cada caso analisado pode ser um diferente. Por exemplo, é possível se obter como resultado da observação questões de interação, uma percepção da necessidade

de mais observações e até mesmo uma confirmação que tenha sido feita previamente pelos avaliadores mesmo. E há maneiras de encontrar esses resultados, que é pelas formas de observação. As formas possíveis são 1) rápida e rasteira 2) testes em laboratório, 3) observação em campo e há também a observação indireta.

2.3.4 Avaliação Por Inspeção

A avaliação por inspeção é feita por especialistas em interface que se passam por usuários. Enquanto eles acessam o que o usuário usaria, eles estão em busca de erros: analisa-se uma tela em busca das falhas da interface de diversas maneiras. Essa se trata de uma avaliação analítica, ela é rápida e é de baixo custo, e pode ser realizada ao longo de todo o projeto.

O objetivo geral da avaliação por inspeção é que se determine o valor de algo, que diga quão bom ele é, se quando for ponderado, ele é uma boa interface de usuário ou se há erros demais, falhas demais e que prejudicariam a experiência do usuário final. É feito uma análise criteriosa da qualidade dessa experiência que o usuário vai ter quando interagem com um certo sistema computacional. E vale ressaltar que só uma interface bonita não é suficiente, precisa-se de que haja um bom desempenho, para que o usuário tenha uma experiência final boa, já que o desenvolvimento anda junto com o design.

É importante que essa avaliação seja feito para que possa haver a correção dos erros antes que seja utilizado pelo usuário final, já que qualquer sistema que está em uso por um usuário está sendo julgado: ou você acha o sistema bom, ruim, ou mediano, compreensível, mas sempre pensa-se algo sobre o sistema, se a experiência foi boa ou se foi muito difícil lidar com aquela aplicação.

Na sessão 2.3.5 é explanada, a avaliação por heurística, que é um exemplo de um tipo de avaliação por inspeção, já que para que seja feita uma avaliação por inspeção, é necessário que siga-se um padrão com o objetivo de detectar as falhas da interface. E a avaliação por heurística traz uma sequência de passos que visam encontrar uma série dos erros mais comuns nas interfaces

2.3.5 Avaliação Por Heurística

É normal assimilar o design à parte estética dos produtos, marcas ou layouts, mas essa não é a totalidade do design. O design é o projeto de algo com um objetivo, tendo um lado estético, mas também o lado funcional. As interfaces digitais se baseiam na construção de meios de comunicação entre uma pessoa e uma máquina.

A avaliação heurística se refere a diversos métodos nos quais uma pessoa que é treinada em IHC e design de interação examina o design que é proposto com a finalidade de avaliar como ele se qualifica perante uma lista de princípios, diretrizes ou heurísticas para o bom design (BENYON, 2010). As heurísticas são uma forma de estratégia cognitiva

que avalia essas interfaces com usuários e diminui a complexidade dos processos de tomada de decisão tornando os julgamentos mais simples e imediatos. Ou seja, essas heurísticas acabam sendo um conjunto de regras que são baseadas na experiência e no planejamento de um determinado projeto, os quais muitas vezes não oferecem garantia teórica como em metodologias algorítmicas e pode ser considerada como a solução ótima ou provavelmente boa para um determinado problema.

As heurísticas são o norte para tornar uma interface mais intuitiva para o usuário e são muito importantes para qualquer profissional de experiência de usuário. É imprescindível que se entenda que desenvolvimento e design estão fortemente relacionados, já que uma interface mal projetada resultará em uma má experiência de uso, portanto mostra-se necessário que o design seja considerado antes, durante e depois do desenvolvimento de um projeto (PRATES; BARBOSA, 2015).

O ideal é que várias pessoas com experiência no design de sistemas interativos façam a avaliação da interface. Nielsen Landauer (1993) recomendam de três a cinco avaliadores, que podem identificar em torno de 75% a 95% dos problemas, onde cada avaliador pode encontrar em torno de 35% dos problemas (GONçALVES, 2013), esse valor pode variar de acordo com a experiência de cada um dos avaliadores. Cada especialista anota os problemas e as heurísticas relevantes, sugerindo soluções onde for possível, além disso, é útil também que haja uma avaliação de gravidade, que diga o impacto provável que esse problema tem seguindo a recomendação de Dumas e Fox (2007) na sua revisão abrangente dos testes de usabilidade (PRATES; BARBOSA, 2015). Esses avaliadores trabalham sozinhos, e somente depois combinam seus resultados.

Pensar em UI (user interface) design é o mesmo que pensar no projeto de uma interface que não deixe o usuário inseguro e que deixe claro qual será o resultado de cada ação sua, de uma maneira que garanta que ele possa realizar todas as tarefas de forma simples e eficiente. Ou seja, é projetar uma interface que faça com que o usuário não precise de um manual de instruções.Existem vários conjuntos de heurísticas dentre os quais escolher, tanto para uso geral quando para domínios de aplicações específicas, e existem dez heurísticas criadas por Jakob Nielsen, cientista da computação, que ajudam a projetar uma boa interface e por consequência uma ótima experiência de uso (SANTOS, 2008).

As heurísticas são as regras ou os caminhos que guiam o usuário a uma solução concreta e eficiente perante um problema do qual não se sabe a resposta. Com isso Jakob Nielsen inventou o método denominado "Avaliação Heurística", que foi baseado em 294 tipos de erros de usabilidade que ele encontrava comumente em suas análises (ROCHA; BARANASKAS,). É feito da seguinte maneira, 3 a 5 especialistas em IHC fazem a inspeção de uma solução de interface, verificando quais heurísticas a interface viola, onde e porquê, baseados nas dez heurísticas que Nielsen criou. As principais vantagens desse método de avaliação são o baixo custo e os resultados rápidos.

Começa-se pela preparação, onde é feita a proposta do design que será avaliada, há hipóteses sobre o perfil de usuários e características de suas tarefas. Nessa etapa, são analisados os esboços e wireframes de tela, a funcionalidade do protótipo, tendo ele ou não o acabamento gráfico. Depois que essa preparação é feita, o próximo passo é a execução, onde as avaliações são relacionadas a telas, e analisa-se se a tela, ou parte dela, está de acordo com a heurística. Se estiver, passa para a próxima heurística, se não estiver, é feita a correção dela. Depois de todas as heurísticas serem analisadas e corrigidas, é feita a consolidação, onde todos os problemas encontrados por todos os inspetores são unidos e é feita a discussão sobre as divergências, e baseado nessas discussões é feito um relatório final com as conclusões de todos os avaliadores em uma só. E por fim, há a conclusão do projeto, onde há a seleção dos problemas que serão corrigidos, já que é necessário mostrar para o cliente os problemas, para que seja vista a gravidade do problema, e o custo benefício de deixá-lo assim e de corrigir.

Cada uma dessas heurísticas é fundamental para que os erros comuns sejam encontrados na interface, as análises são feitas em uma heurística de cada vez, até finalizar todas.

- Status do sistema claramente visível: Os usuários devem estar sempre informados sobre o que está acontecendo, o sistema deve fornecer *feedback* adequado, dentro de um tempo razoável

- Correspondência entre sistema e mundo real: O sistema deve apresentar palavras e expressões que sejam cotidianas e de fácil entendimento do usuário, na linguagem do usuário, não nos termos do sistema. As informações devem aparecer uma ordem natural e lógica, e uma linguagem simples de entendimento para o usuário.

- Liberdade e controle para o usuário: Se o usuário escolher uma função por engano, ele deve ter a opção de desfazer sua ação facilmente, deve estar bem visível como se sai do estado indesejado sem a obrigação de percorrer grandes caminhos. É a liberdade que o usuário precisa de uma saída de emergência facilmente visível.

- Consistência e padronização: Deve haver um padrão nas palavras, o usuário não deve ter que se preocupar se duas palavras similares designam a mesma função, deve haver um padrão.

- Prevenção de erros: Um projeto cuidadoso já consegue prever qual erro o usuário irá cometer, e o ajuda a evitar que o problema ocorra, as instruções devem ajudar o usuário a realizar a sua tarefa sem problema algum.

- Reconhecer ao invés de decorar: Os objetos e as ações têm que ser visíveis, o usuário não precisa decorar um ícone ou objeto em nenhuma parte da aplicação, deve sempre

ter a visibilidade do que acontecerá se for clicado naquele ícone, ou seja, não faça o usuário ter que pensar.

- Flexibilidade e eficiência no uso: O uso de aceleradores (atalhos para alguma função) apesar de serem imperceptíveis aos usuários novatos, tornam a interação de usuários experientes muito mais rápida. É uma função que serve igualmente bem ambos os tipos de usuários.

- Design minimalista e estético: Informações desnecessárias e irrelevantes não devem existir na interface, cada palavra de informação que é colocada sem necessidade no diálogo acaba reduzindo a visibilidade relativa de informações relevantes.

- Apoio para o usuário reconhecer, diagnosticar e consertar erros: As mensagens de erro têm que ser expressas em linguagem simples de entendimento, não na linguagem de erros do sistema. Deve-se indicar precisamente o problema e sugerir uma solução que seria a mais eficaz, ou seja, as mensagens de erros devem ajudar o usuário a entender o que houve.

- Ajuda online e documentação: O melhor caso é um sistema que não precise de documentação pela sua facilidade, mas informações documentais de alta qualidade devem ser encontradas. Essa ajuda com manual deve existir, e devem ser focadas na tarefa do usuário, com enumeração de passos concretos que serão realizados. Devem haver informações de ajuda e facilmente encontradas e compreendidas.

3 TRABALHOS RELACIONADOS

O primeiro trabalho (ZANATTA, 2015) relacionado a esse é o "Programação de computadores para crianças - Metodologia do Code Club Brasil" de 2015, que visava estudar a metodologia do Code Club Brasil para crianças a partir de 9 anos de escolas públicas e privada aprenderem programação. Foi feito um estudo de caso: crianças foram submetidas a um questionário sobre conceitos de lógica de programação antes e depois de serem submetidas as aulas dos voluntários, pelo método do Code Club Brasil, onde os alunos aprendem com a ferramenta scratch, e a conclusão foi que a metodologia que o Code Club Brasil é adequada para a idade, porém que cada criança tem o seu ritmo de aprendizagem, e o seu nível de facilidade para aprender tal assunto.

O segundo trabalho (NASCIMENTO, 2015) relacionado a esse é o "Introdução ao Ensino de Lógica de Programação para Crianças do Ensino Fundamental com a ferramenta Scratch" de 2015, que objetivava o contato das crianças com a lógica de programação, sendo assim também um estudo de caso, envolvendo crianças de uma escola pública local utilizando o software Scratch. Foram aplicados conceitos de algoritmos e programação de softwares baseados em situações reais e cotidianas. Em seguida, os instrutores fizeram um aclaramento sobre os conceitos e modo de usar o Scratch, e por fim, os alunos presenciaram uma aula prática afim de treinar seus conhecimentos, e a conclusão foi que os alunos ficaram muito interessados e animados com as aulas que tiveram.

O terceiro trabalho (LUCRECIO, 2016) relacionado a esse é o "Comparação e aplicação de diferentes ferramentas para ensino de programação para crianças" de 2016, que tinha como objetivo também um estudo de caso, porém nesse trabalho foram usada duas ferramentas: Scratch e VisuAlg afim de determinar qual seria mais eficaz no aprendizado dos alunos, e no fim foi concluído que os alunos que aprenderam Scratch responderam mais questões e corretamente do que os que aprenderam VisuAlg. Foi feito com alunos do ensino fundamental de uma escola pública, e um questionário foi aplicado após as aulas. A conclusão dos alunos em relação a experiência foi entusiasmada e muito positiva.

Enquanto os três primeiros trabalhos tiveram seu foco em um estudo de caso, e provaram que alunos do ensino fundamental aprenderem programação é algo realizável e do interesse deles, este visa focar na importância da inclusão do estudo da lógica de programação para todas as crianças, e de linguagens de programação para os jovens, e ainda, de gerar um modelo que diferencie as maneiras de aprendizado adequadas respeitando a diferença etária, e que mostre, ainda, quais são os modos mais adequados e eficazes para cada idade aprender a programar. Fundamentado nos trabalhos analisados, é pertinente afirmar que o ensino/aprendizado de lógica de programação e de programação em si é um assunto atual - trabalhos relacionados feitos nos últimos 3 anos - e relevante, pois objetiva que alunos, principalmente do ensino fundamental, estejam aptos a fazerem códigos

e programas, ensinados por diversos métodos em busca do modelo mais eficaz segundo a faixa etária determinada. Não é uma disciplina simples, nem fácil, porém é necessária, e o propósito é achar o meio mais incomplexo para melhorar o ensino de programação para as crianças.

4 METODOLOGIA

O trabalho que é empreendido e apresentado neste projeto utiliza uma metodologia de desenvolvimento experimental, contando com o apoio de uma equipe multidisciplinar da escola. A equipe executora é composta por docentes com formação e atuação na área de Ciência da Computação, Letras e Pedagogia. De uma maneira geral, todo o planejamento das aulas, a previsão e a preparação dos ambientes e ações é realizado por toda a equipe envolvida.

Para analisar a viabilidade do uso do Code Club nas escolas públicas do Tocantins, é utilizada a abordagem de avaliação de Interfaces Homem Máquina por observação e por inspeção explanadas no capítulo 2.3 da fundamentação teórica. A seguir, a metodologia para cada uma dessas duas avaliações é melhor explanada. Como o objetivo de responder às questões e obter um resultado mais preciso, foi feita a combinação de paradigmas e técnicas de avaliação, conforme sugere Preece et al. (2005).

4.1 Avaliação por Observação

A avaliação por observação se torna primordial nesse caso, visto que é a aplicação na prática do sistema em um ambiente real. A Figura 4.1 mostra o diagrama com a sequência de passos do processo para que esta avaliação possa acontecer. Primeiramente deve ser realizada a seleção dos alunos que irão participar das aulas, em seguida é definida uma data para a aula introdutória que será dada antes de principiar a metodologia em si, essa aula introdutória visa explanar para as crianças como será o projeto e aplicar os primeiros questionários para a avaliação. Após, será dado início as aulas seguindo a metodologia do Code Club, permitindo a avaliação por observação. Na última aula é aplicado o questionário final (que apesar de ser o mesmo questionário do inicial, tem a ordem trocada) para comparação, e por fim, os dados que foram obtidos são analisados. A seguir cada um desses passos será melhor explanado.

O objetivo principal desse projeto é o mesmo da metodologia Code Club: colocar em prática o ensino de programação para as crianças de uma forma lúdica que as estimule a adquirir esses conhecimentos em lógica de programação. Assim para a etapa das aulas práticas em laboratório é usado inteiramente e exclusivamente a metodologia do Code Club: o que é lecionado em cada aula, quais são os exercícios que as crianças devem fazer e quais são as propostas extra-curriculares, por exemplo. O programa fornece até mesmo folders encorajadores para exposição na escola. O Code Club conta também com aulas para os voluntários, de uma maneira que saibam o caminho do que tem que ser apresentado para os alunos, criando um padrão nas aulas do Code Club do mundo inteiro. No site do Code Club, há a parte exclusiva para voluntários, e lá se encontram todos os passos-

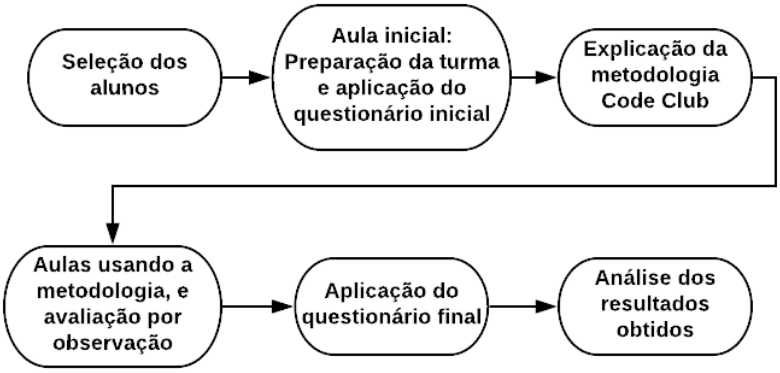


Figura 4.1 – Diagrama de sequência do processo das aulas do Code Club

Tabela 4.1 – Dados relevantes do projeto

Nível de ensino: Ensino fundamental
Séries: 4º, 5º e 6º ano
Ambiente de Programação: Scratch
Turno: Matutino, contra-turno das aulas
Objetivos: Ensinar lógica de programação, códigos e programação em si para os alunos.
Forma de avaliação e validação: Análise qualitativa e quantitativa, estudo de caso e questionários.

a-passos da criação de um clube, e contém todo o material que será utilizado, inclusive os pdfs que descrevem aula por aula. As aulas seguem uma ordem de dois módulos de scratch, dois módulos de HTML e CSS e dois módulos de python, como mostra a figura 4.2. As imagens 4.3, 4.4 e 4.5 exemplificam a página dos voluntários dentro do primeiro módulo de cada etapa. É possível notar que cada módulo que é lecionado conta com 6 projetos dentro deles, onde cada projeto desses conta com um pdf explicativo da aula, materiais do projeto e notas para o líder do clube. No apêndice 1, há um exemplo de um arquivo em texto oficial da metodologia na sua versão resumida do primeiro módulo de scratch. Esse arquivo é exclusivo da área de voluntários, são por esses arquivos e materiais que as aulas são baseadas. A tabela 4.1 mostra as informações mais relevantes sobre o início do projeto.

Na metodologia proposta por este trabalho, não há um critério técnico para a escolha dos alunos, a escola faz a seleção dos alunos interessados em participar do projeto e que têm a possibilidade de ir no contra-turno do horário escolar.

Apos a seleção dos alunos, para dar continuidade a esse projeto, as seguintes atividades devem ser realizadas:

- Análise de todo o material disponibilizado pelo Code Club para entendimento do

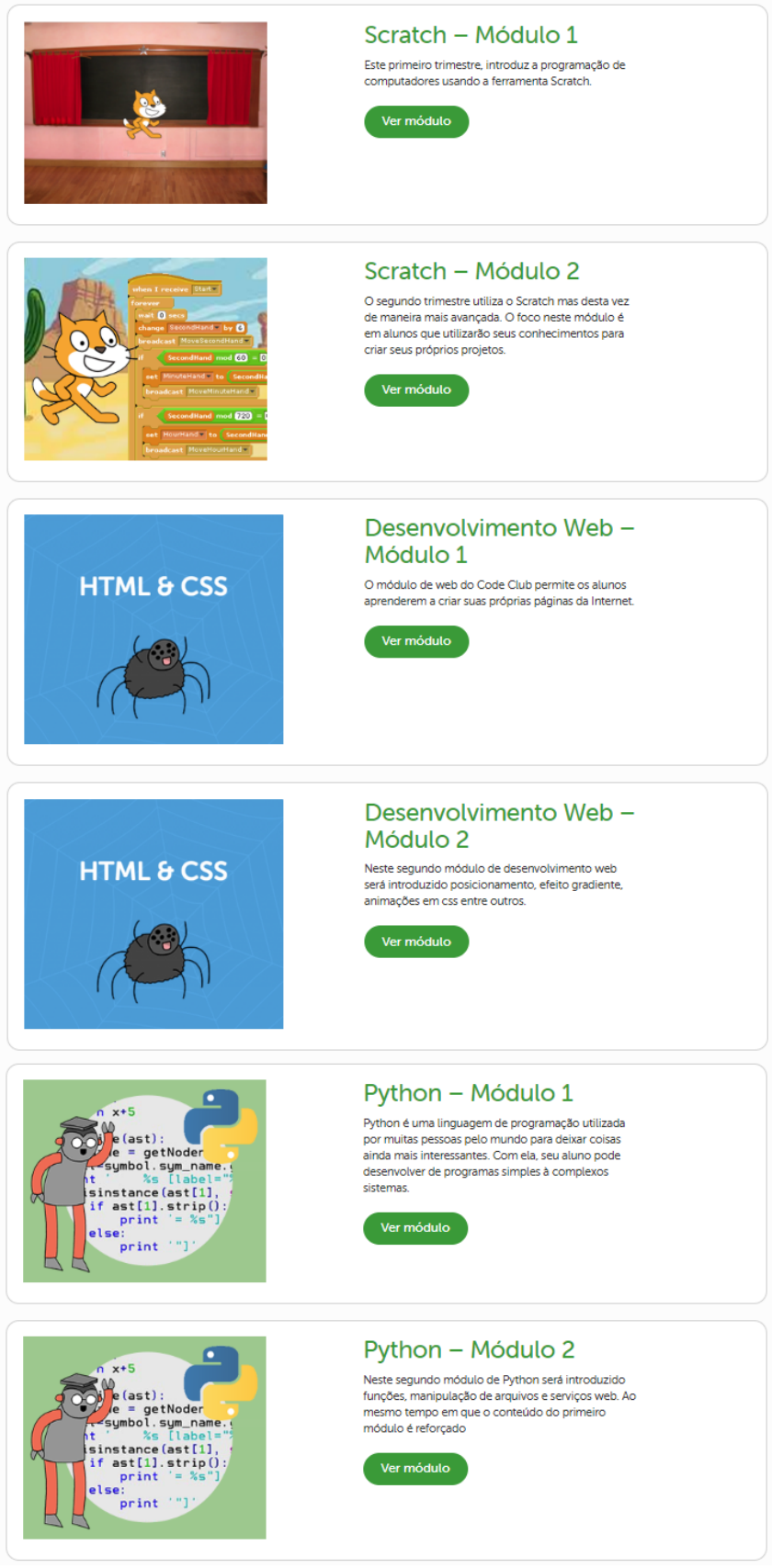


Figura 4.2 – Tela de captura dos módulos do Code Club

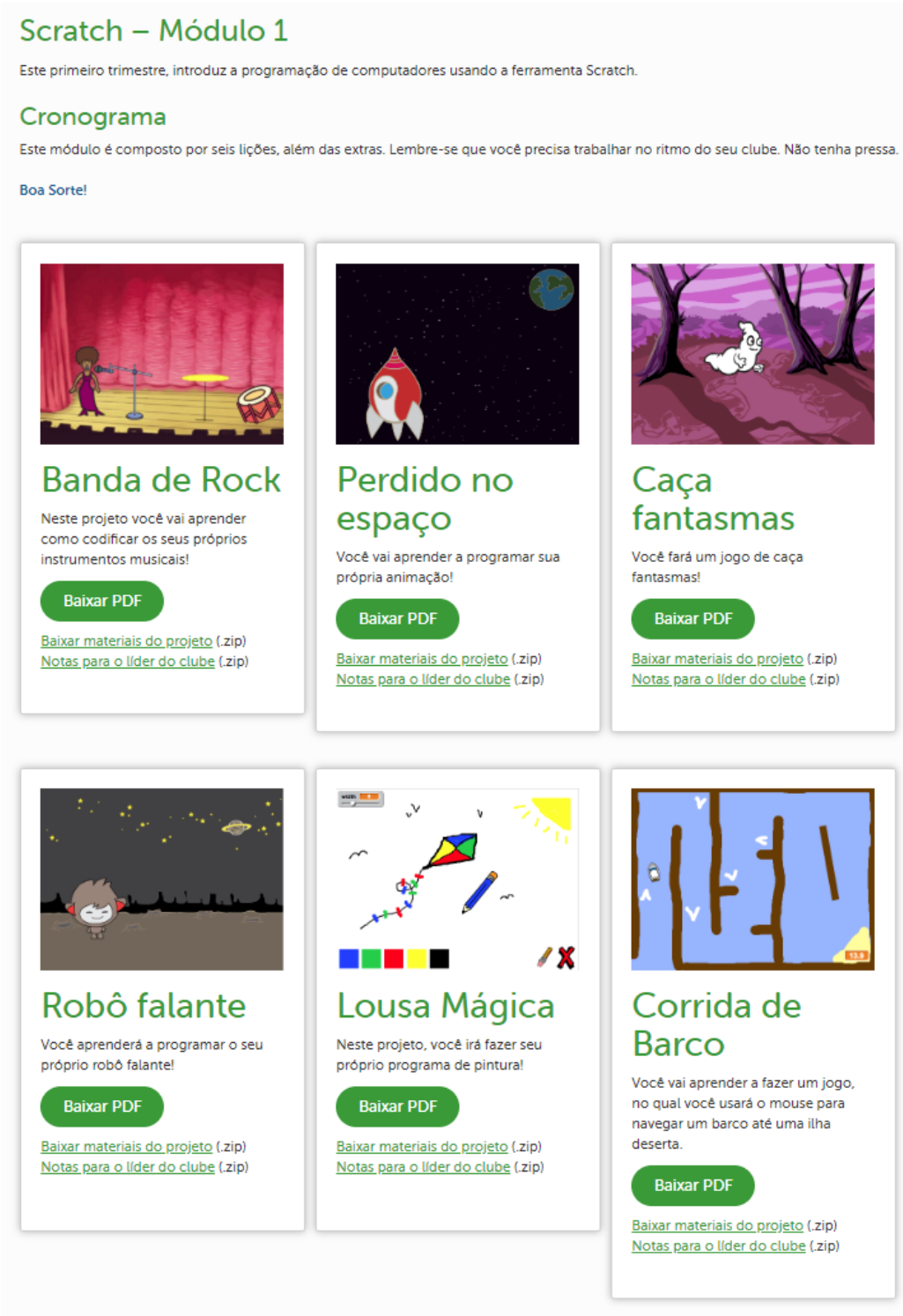


Figura 4.3 – Tela de captura da página dos voluntários do Code Club, na seção Módulo 1 do scratch

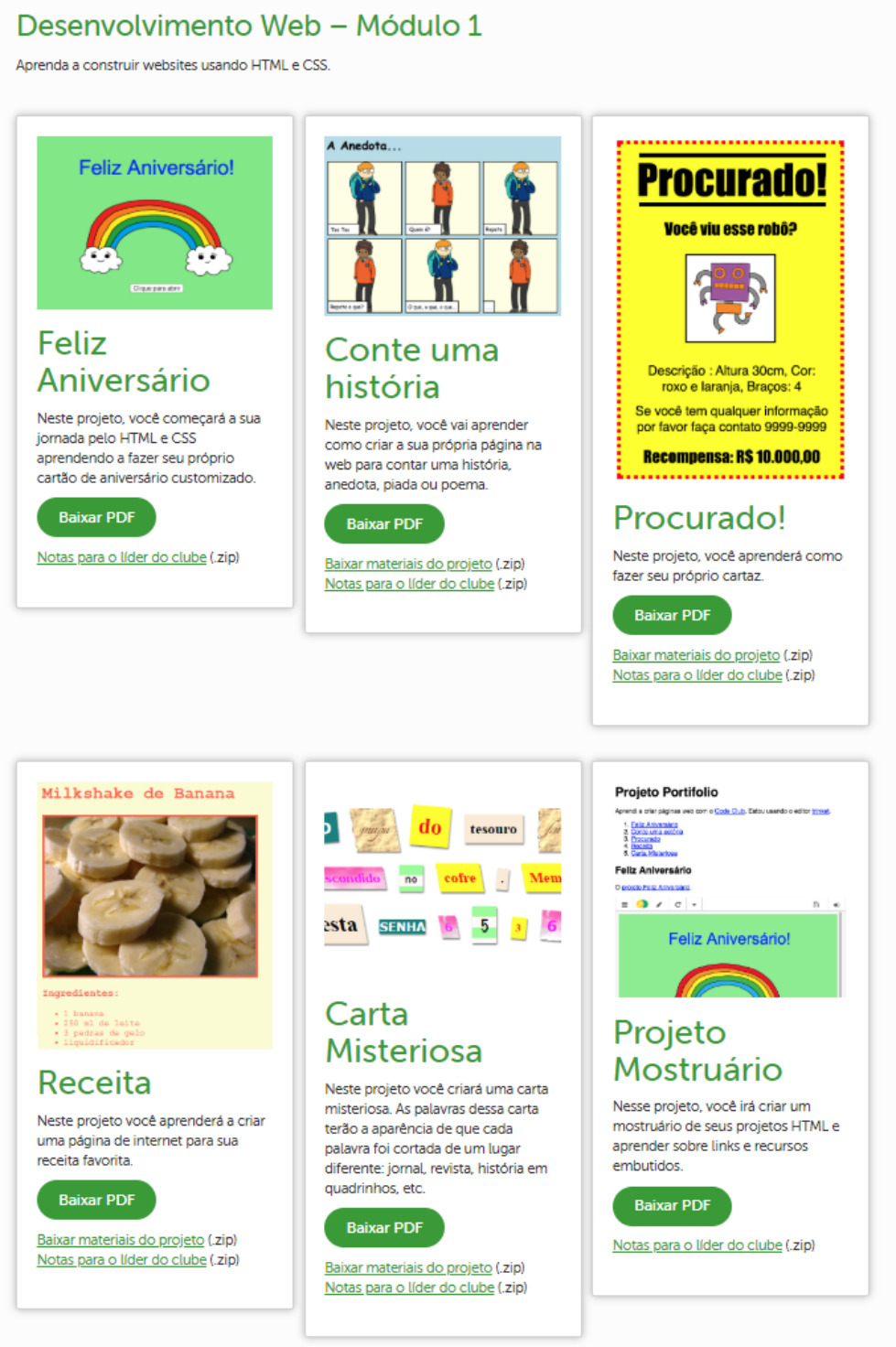


Figura 4.4 – Tela de captura da página dos voluntários do Code Club, na seção Módulo 1 do HTML e CSS

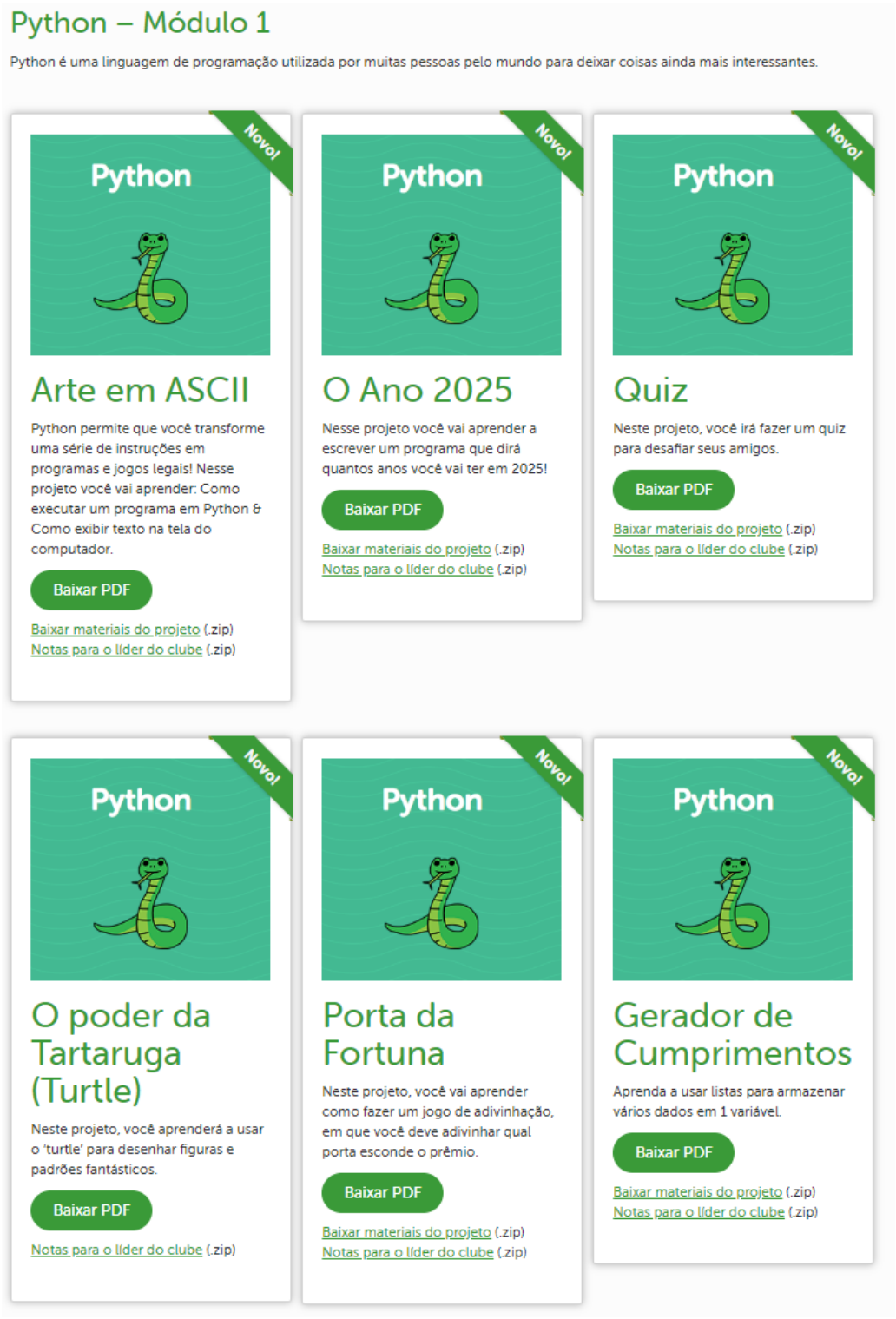


Figura 4.5 – Tela de captura da página dos voluntários do Code Club, na seção Módulo 1 de python

projeto inteiro antes do seu início. O material do Code Club conta com aulas para os instrutores, o material de cada aula especificamente, os exercícios que serão trabalhados em sala de aula em cada módulo e também os encaminhados para serem feitos em casa.

- Pesquisa sobre programação para crianças no Brasil e no exterior, e também, softwares que auxiliam no processo de aprendizagem de programação para crianças.
- Criação de três questionários para os alunos: um em relação a motivação para o projeto, um em relação a experiência no final, e outro em relação ao conteúdo a ser lecionado.
- Início da validação do método estudado, por meio de estudo de caso realizado com alunos do Ensino Fundamental.
- Aplicação das aulas do Code Club com o scratch.
- Aplicação dos questionários com os alunos.
- Obtenção de um registro de opiniões dos voluntários envolvidos e o questionário sobre a experiência para os alunos.
- A coleta dos dados dos questionários, uma comparação entre os dados obtidos e apresentar os resultados.

Conforme explicitado na figura 4.1, antes do início da aplicação da metodologia do Code Club, há uma aula de apresentação do projeto, discussão sobre o Code Club e seus ramos, é explanado o que é visto nas aulas, e qual é o objetivo final. São aplicados também dois questionários: um questionário visa entender o porquê os alunos se interessaram em participar do projeto, se eles já haviam tido algum contato com algoritmos ou programação e se gostavam de programação. O outro, para o método de avaliação da eficácia da metodologia no aprendizado de programação, é um questionário que foi aplicado que visa identificar fatores de aprendizagem e percepção, ou seja, ele tem a finalidade de checar o quanto os alunos sabem antes das aulas, e quanto o projeto é proveitoso, já que o mesmo questionário é aplicado no final das aula.

Esse questionário foi elaborado com dez questões, todas de múltipla escolha, onde uma das questões visava avaliar o quanto eles acreditam no fator de que programar os ajudaria em outras disciplinas.

Seguido da aula de apresentação, são iniciadas as aulas do Code Club, a sequência e descrição de cada aula, pode ser vista nas tabelas 2.1, 2.2, 2.3, 2.4, 2.5 e 2.6 que foram explanadas anteriormente. Nas primeiras aulas, antes da inclusão do scratch, devem ser ensinadas algumas noções básicas de algoritmos com dobraduras de papel e exemplos

cotidianos, como fazer um bolo, por exemplo. Na tabela 4.2 é mostrado o primeiro módulo do scratch detalhadamente.

O Code Club fornece o método necessário para que esse ensino ocorra, no site dos voluntários há seis módulos de aulas - dois de scratch, dois de html e css e dois de python. Cada módulo desse conta com:

- Um arquivo de catorze páginas explicando passo-a-passo o que os alunos vão fazer, conta com desenhos que ajudam na compreensão e lista de atividades durante os capítulos. Se trata de um arquivo de texto que os alunos podem rever em casa e refazer tudo que fizeram na aula, já que ele tem todas as explicações do módulo, dicas de como fazer melhor, e conta com desafios que são baseados no que eles já fizeram, porém de uma maneira mais complexa.

- Uma pasta com os materiais do projeto.

- Nota para líderes de Clubes, que tem o objetivo de ajudar os voluntários a estarem melhor preparados para lecionarem as aulas. Ele mostra os recursos que são utilizados na aula, os objetivos de aprendizagem do módulo, os desafios, e no fim apresenta perguntas frequentes de outros voluntários.

O Code Club fornece também materiais de divulgação: imagens de logo, imagens de apresentação, folders encorajadores, proposta pedagógica, termo de responsabilidade e cessão de direito de imagens dos participantes, e também, pôsters sobre programação, entre eles, pôsters de repetição, decisão, sequência, variáveis e input. A figura 4.6 exemplifica como são esses pôsters mostrando o pôster *input*.

Ao fim do projeto, é aplicado novamente o mesmo questionário que é aplicado no início, com o objetivo de determinar se a abordagem das aulas do Code Club auxiliam de algum modo, e se surtem efeito e melhoria na capacidade de assimilação de conceitos básicos de sequências de comandos, códigos, e programação para as crianças. O outro questionário que é aplicado no final é um questionário onde os alunos davam notas de zero a dez em diversos quesitos referentes as aulas e a metodologia. A tabela 4.3 mostra quais foram essas perguntas. Esses questionários devem ser analisados e estudados para verificar a viabilidade do uso do Code Club nas escolas do Tocantins e serão apresentados na seção de resultados.

Tabela 4.2 – Aulas detalhadas do módulo 1 de scratch

Aula	**Objetivos**
Aula 01	Apresentação do voluntário e da metodologia Code Club. Noções básicas de lógica de programação no dia-a-dia. Aplicação do questionário de motivação. Aplicação do questionário inicial. Introdução ao scratch.
Lição	Criação de um jogo simples.
Aula 02	Começar a trabalhar no que foi visto na aula 01, são abordados temas como: troca de cenário, seguir o mouse e o uso de variáveis (usando pontuação).
Lição	Criação de um jogo de pega-pega entre um gato e um rato, no qual você controla o rato com o mouse na tentativa de fugir do gato.
Aula 03	Definir uma variável, laços de repetições e contagem de pontos.
Lição	Criar uma bruxa que apareça aleatoriamente e suma quando for clicada, de uma maneira que haja um contador para a quantidade de vezes que a bruxa foi acertada.
Aula 04	Responder a cliques do mouse, alterar aparência de objetos, emitir sons de objetos, transmitir e recepcionar eventos.
Lição	Criação de um foguete, e fazer ele explodir de diversas maneiras.
Aula 05	Fazer laços de repetição e pará-los.
Lição	Criação de um ponto onde diversos objetos aparecem aleatoriamente na mesma posição. Fazer a condição de parada quando clicado.
Aula 06	Movimentar e controlar personagens, utilizando uma visão do sistemas de coordenadas, de uma maneira que detecte colisões.
Lição	Criação de um jogo para orientar um peixe que quer comer todas as presas do mar.
Aula 07	Aula sobre eventos, variáveis, animação, emitir sons de objetos, condições de parada e modificação de recursos.
Lição	Montagem de um jogo de corrida no deserto de um leão e um papagaio. Na linha de chegada, mostrar quem foi o vencedor e reiniciar o jogo.
Aula 08	Definir placar, mudar trajes e aparência, definir respostas aleatórias, detectar um clique e verificar se um objeto foi clicado.
Lição	Acertar qual é o objeto entre os que aparecem que está distorcido.
Aula 09	Aula sobre condições de parada, comparação numérica e contas simples de matemática.
Lição	Criação de uma ferramenta de pintura para criar desenhos. Ela permitirá escolher cor do lápis, limpar a tela, usar carimbos, etc.
Aula 10	Aula para incentivar a criatividade, o planejamento e a correção de erros.
Lição	Proposta de criação de algum jogo feito pelo aluno.

Figura 4.6 – Pôster de divulgação: Input

Tabela 4.3 – Perguntas feitas no questionário final de satisfação

1) A maneira como as aulas são dadas.
2) A dificuldade do uso do scratch.
3) O conteúdo das aulas.
4) Os tipos dos exercícios feitos em sala.
5) A dificuldade dos desafios do fim das aulas.
6) A sua vontade de continuar indo para as aulas durante o projeto.
7) O nível de dificuldade do projeto inteiro.
8) O resultado final dos seus projetos.
9) A sua vontade de aprender mais sobre programação depois do projeto.
10) O quanto o projeto inteiro foi proveitoso para você.

4.2 Avaliação por Heurísticas

A avaliação por heurística e as heurísticas de Nielsen foram citadas e explanadas na fundamentação teórica, na sessão 2.3.5 e são utilizadas na metodologia desse trabalho da seguinte maneira: é escolhida uma das etapas do jogo, as etapas iniciais do primeiro módulo do scratch, e foram feitas capturas de tela com a finalidade de averiguar, por meio das heurísticas de Nielsen, se a interface está de acordo com cada uma das heurísticas apresentadas. Esses quesitos que são analisados são dez ao total:

1. Status do sistema claramente visível
2. Correspondência entre sistema e mundo real
3. Liberdade e controle para usuário
4. Consistência e padronização
5. Prevenção de erros
6. Reconhecer, ao invés de decorar
7. Flexibilidade e eficiência de uso
8. Design minimalista e estético
9. Apoio para o usuário reconhecer, diagnosticar e consertar erros
10. Ajuda online e documentação

As telas que foram selecionadas passam por todas essas heurísticas, uma tela de cada vez, por uma heurística de cada vez. O objetivo do uso da avaliação por heurísticas é a análise da interface e encontrar erros para que seja possível a melhoria da interface.

Os erros que são buscados nessas telas são:

- Se há um *feedback* adequado dentro de um tempo razoável.
- Se o usuário entende facilmente as palavras e expressões.
- Se o usuário consegue sair facilmente de um lugar indesejado.
- Se há um padrão nas palavras, não causando confusão ao usuário sobre se aquele botão faz isso mesmo.
- Se o sistema ajuda o usuário a evitar erros.
- Se os objetos e as ações são facimenite visíveis.
- Se há maneiras de facilitar o uso dos mais experientes sem atrapalhar o uso dos iniciantes.
- Se informações desnecessárias aparecem na tela, poluindo-a.
- Se as mensagens de erros são sucintas, diretas e de fácil compreensão.
- Se há um manual de ajuda com passos enumerados.

4.3 Avaliação por Inspeção

Na avaliacao preditiva por inspeção segue-se os passos mostrados na figura 4.7, onde é feito todo um estudo sobre as heurísticas de Nielsen, e a importância da validação de cada uma. Esse entendimento das heurísticas torna possível que quando as capturas de tela estiverem prontas para avaliação, elas serão avaliadas de uma forma fiel a Nielsen. Para a preparação dos testes, são usadas as heurísticas também de uma maneira que torne possível a diminuição de erros, ou seja, prepara-se o material a ser utilizado tentando não cometer erros que serão procurados depois. Após isso foi feita a organização do material para que as aulas pudessem ser dadas. Ao fim das aulas, conta-se com diversas capturas de telas feitas durante o projeto, e é feita a seleção dessas telas para poderem ser analisadas. Foram selecionadas seis telas para passarem por essa avaliação por inspeção: todas elas são as do módulo 1 de scratch, situadas no início do projeto. Assim foram escolhidas as telas para que a avaliação por inspeção pudesse ser feita.

A heurística funciona da segunda maneira: na primeira imagem é analisada a primeira heurística, se ela não satisfazer a heurística, são feitas as anotações para posteriores correções, e tenta-se a próxima heurística. O processo se repete até se ter o conhecimento de todas as imagens em relação à todas as heurísticas. Por exemplo, a imagem 1 falhou na heurística de prevenção de erros e a imagem 2 falhou na heurística de desing minimalista e estético. Ao fim, são juntados todos as falhas para que possa ser feito o relatório final que analisa essas falhas, a quantidade delas e o grau de severidade delas, e é analisado se a interface é aprovada ou reprovada.

Para essa avaliação foi selecionada a tela inicial do primeiro módulo da primeira aula de scratch. Como exigência para esse método, os profissionais de Interfaces Homem Máquina foram a autora deste trabalho sob a supervisão do Prof. Dr. Eduardo Ribeiro. Cada um dos itens/requisitos avaliados foram explicitados na seção de fundamentação teórica.

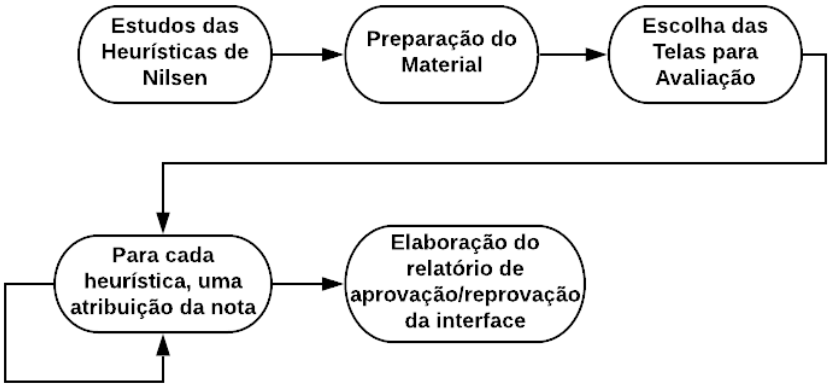


Figura 4.7 – Diagrama de sequência das heurísticas

5 RESULTADOS

Conforme explicitado na metodologia, abaixo estão os resultados obtidos pela avaliação por observação *in-loco* bem como a avaliação preditiva feita por inspeção de uma das atividades utilizadas. A avaliação foi feita por meio de observações e baseada na análise dos resultados produzidos pelos alunos. Na avaliação por obervação houve algumas particularidades intrínsecas dos imprevistos de uma atividade como essa. Tais particularidades também serão melhor explicitadas, porém, as mesmas não atrapalharam na avaliação em geral.

5.1 Avaliação por Observação

Para a avaliação por observação, a coordenação da Escola Estadual Vale do Sol fez a seleção de dez alunos que se mostram interessados em participar do projeto e que possuiam a possibilidade de ir no contra-turno do horário escolar. Os alunos são do 4º ao 6º ano, todos entre 11 e 13 anos. Na escola selecionada para esse estudo, há muitas crianças com distorção idade-série, que é o aluno com mais de 2 anos de atraso escolar, por isso para atender a idade requisitada pelo Code Club, os alunos selecionados deverão ser de séries mais avançadas. Nessa relação faixa etária, obtêm-se o seguinte histograma: 2 alunos com 10 anos, 1 aluno com 11 anos, 3 alunos com 12 anos e 4 alunos com 13 anos. Entre esses alunos contamos com 6 meninos e 4 meninas no início das aulas.

A escola municipal conta somente com dois notebooks que são da coordenação, o que dificulta o trabalho, visto que os alunos são acostumados a usarem apenas os celulares para todo tipo de trabalho escolar. Essa dificuldade causa um desestímulo nas crianças, por quererem testar o que aprendem, mas a dificuldade é contornada com dois novos notebooks que a escola empresta para as aulas, e o uso de uma smart tv que a escola disponibiliza, o que ajuda a manter as crianças indo para a escola para o projeto, mas devido à essa escassez de computadores, tornou-se mais difícil e demorada a execução das aulas. Por exemplo, era dada uma instrução para as crianças fazerem uma ação no scratch, duas crianças faziam, e alternavam as crianças no mesmo computador, em abas diferentes na internet, ou seja, levou três vezes mais tempo do que levaria se cada um tivesse acesso à um notebook.

Esse fator dos notebooks serem compartilhados levou às crianças a trabalharem em duplas e trios em um mesmo notebook, e isso gerou - além da demora para finalizar os exercícios - desavenças entre as crianças, porque foi visível que alguns alunos conseguem trabalhar adequadamente em dupla, enquanto outros são mais individualistas. O trabalho em dupla foi um desafio nesse projeto, pois muitas crianças não aceitam, além disso, crianças com mais dificuldade de aprendizado relataram estarem “atrapalhando” os colegas

que faziam junto, e já estavam mais avançados nos exercícios.

Como foi dito anteriormente, houve uma aula inaugural antes do início da aplicação da metodologia, e foi um dos momentos desafiadores desse projeto, pois nesse início foi difícil explicar o que era a programação de uma forma que eles conseguissem visualizar de uma forma próxima a realidade, como se eles pudessem programar. Tentando melhorar a definição de lógica de programação na cabeça das crianças, foram feitos diversos exercícios de "passo-a-passo" que são usados diariamente, como as etapas para se fazer um bolo, por exemplo. Quando foram apresentados exemplos corriqueiros, tornou-se mais fácil fazer comparações, mas em sua grande maioria, os pupilos não sabiam muito sobre programação, lógica, linguagem ou ambiente de programação, scratch, html, css ou python.

Durante as aulas, há exercícios a serem feitos, e no final de cada aula há um desafio utilizando o que foi lecionado durante a aula, porém com um nível de dificuldade um pouco maior. O fato da divisão dos notebooks sempre atrasava as aulas, e elas acabavam sempre depois do previsto. Dos 10 alunos, havia 3 que mostraram mais dificuldade do que os outros para concluir até mesmo os exercícios que eram feitos em conjunto, porém, mesmo demorando mais, fazendo mais perguntas e precisando de mais ajuda, todos os alunos em todas as aulas finalizaram os exercícios e os desafios por completo. Houve um caso específico de um aluno que se mostrou extremamente interessado em descobrir a programação, e chegou a assistir video-aulas em casa, e pedir ajuda com certas dúvidas que restaram. Esse aluno estudou em casa, fez vários exercícios duas vezes, e melhorou consideravelmente do primeiro para o segundo questionário.

Como os projetos do Code Clube no scratch são lúdicos e envolvem criatividade, quando se comparava, por exemplo, o mesmo projeto de dois alunos, era possível ver a diferença que alguns gostavam mais do que outros de explorar o scratch, a fim de melhorar seu projeto. Um dos primeiros módulos do scratch é a banda de rock, onde eles montam um cenário, colocam um cantor, escolhem a música, o que vai tocar quando clicar, a maneira como o desenho vai mudar quando for clicado para tocar e quais instrumentos haveria na cena. Esses são exemplos de tarefas que eles deviam fazer no dia desse projeto, porém ao olhar o projeto de um aluno, se vê apenas o básico: se foi ensinado usando o exemplo do tambor usando o primeiro toque da lista, o aluno somente reproduz igual no seu projeto - usa o mesmo instrumento e o mesmo toque. Porém, nesse mesmo projeto, houve um aluno que mudou completamente o cenário, colocou diversos instrumentos, e diversas respostas ao som (quando um instrumento tocava, por exemplo, saía "faíscas" dele, como se fazendo o desenho do som. Nesse projeto havia ainda uma troca de roupa da cantora, e diversas músicas tocavam, era realmente um projeto inteiro de uma banda de rock.

Foi possível concluir, com situações como a que foi exemplificada que alguns alunos tinham mais facilidade do que outros nos exercícios propostos, e também que alguns acabaram gostando mais do que outros e queriam ir além. A maioria dos alunos passaram

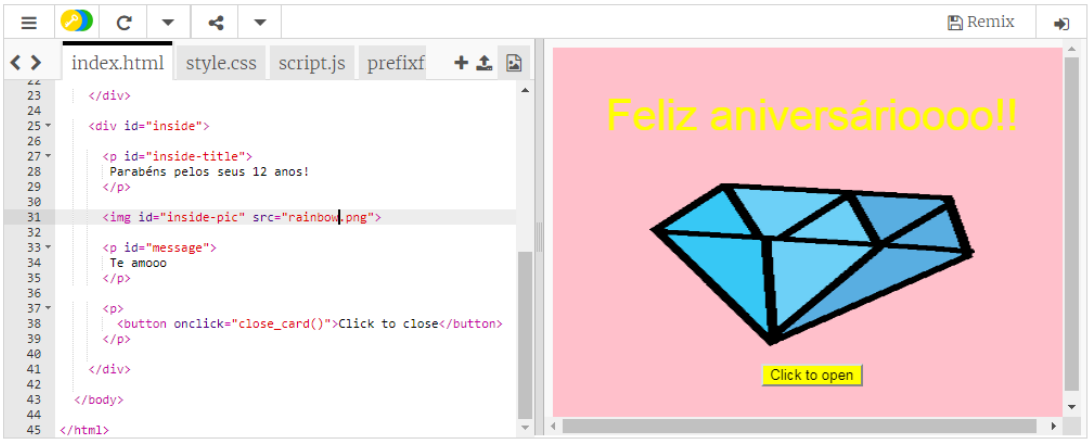


Figura 5.1 – Tela 1 do projeto feliz Aniversário de um aluno

Figura 5.2 – Tela 2 do projeto feliz Aniversário de um aluno

a reconhecer que o software pode controlar operações do computador de forma interativa. Mas mesmo assim, foi possível também ver já na segunda aula o tamanho desinteresse de duas alunas: ambas não voltaram a partir da terceira aula. A taxa de desistência foi de 1/5, e todos os desistentes eram do sexo feminino. Mesmo a aceitação não sendo unânime, acredita-se, por meio desta experiência, que seja possível perceber que o ensino da computação pode ser integrado ao currículo do Ensino Fundamental de forma harmônica e interdisciplinar.

Na avaliação por observação, foram analisadas todas as telas finais de todos os projetos de todos os alunos. As figuras 5.1, 5.2 e 5.3 exemplificam um projeto de um aluno, do módulo de html onde eles deviam fazer um convite de aniversário. É um convite simples, mas foi feito inteiramente pelo aluno, como a aula previa.

Na avaliação por observação, foram aplicados questionários com as crianças, um deles visa medir a aprendizagem dos alunos, o mesmo questionário foi aplicado na aula de apresentação e na aula final, para que houvesse uma comparação. É possível ver

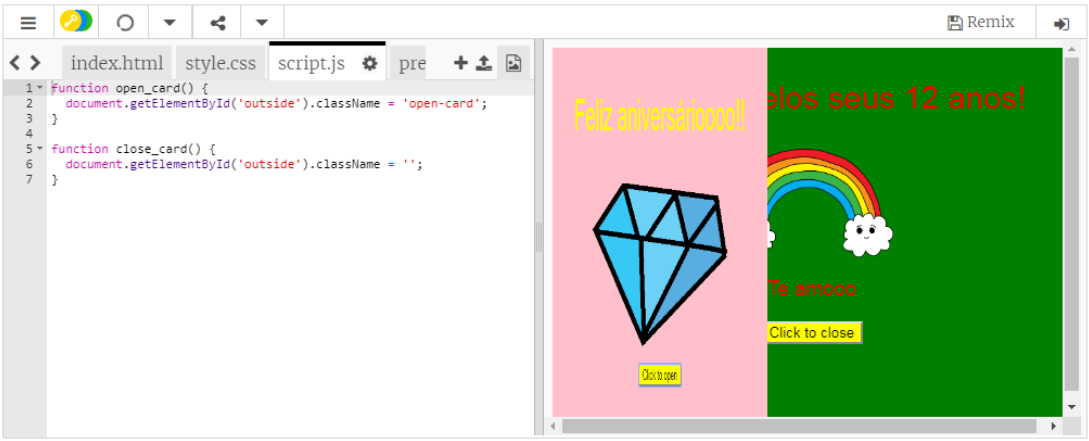


Figura 5.3 – Tela 3 do projeto feliz Aniversário de um aluno

por meio dos gráficos das figuras 5.4, 5.5, 5.6, 5.7, 5.8, 5.9, 5.10, 5.11, 5.12 e 5.13 que quando as crianças fizeram a avaliação pela primeira vez, antes das aulas, a taxa de acerto foi muito baixa. Tem-se que levar em consideração o fato de que é uma escola pública em uma região afastada do centro de Palmas, e o projeto lidava com crianças menos favorecidas e com menos oportunidades de estudo. Mas ao comparar a quantidade de perguntas que estavam certas na primeira vez em que o questionário foi aplicado com a segunda vez, é possível ver que apesar da quantidade de alunos terem diminuído devido as desistências que ocorreram, ainda houve aumento nas respostas certas, como mostra o gráfico da figura 5.14. E mesmo com essas desistências, quando é comparado a quantidade de acertos que houveram questão por questão, percebe-se que apenas uma questão teve queda na quantidade de acertos do primeiro para o segundo questionário. Todas as outras questões ou mantiveram, ou aumentaram o número de acertos, como se vê na tabela 5.1. Na tabela 5.2, tem-se a versão mais detalhada dessa diferença usando porcentagens das questões que foram feitas corretamente, e ainda mostra a que se referia cada questão.

De um total de 10 alunos no pré-teste apenas 3 (30%) conseguiram explicar corretamente como conseguiram chegar na solução dos problemas, no entanto, a maioria 7 (70%) não soube explicar por meio de uma sequência lógica de passos como conseguiram chegar a um resultado. Esse fator é levado em consideração devido ao fato das questões serem de múltipla escolha. Quanto ao pós-teste, é possível notar uma melhora sutil, pois 5 (62,5%) dos alunos explicaram adequadamente os passos que o levaram a solução das questões.

Outro ponto que vale ressaltar sobre os resultados observados é que as questões que envolviam lógica de programação, tiveram uma taxa de erro muito grande no primeiro questionário, e uma taxa de acertos muito grande no segundo questionário. Acredita-se que as aulas tenham ajudado nessa diferença visto que quando os alunos compreendem a lógica de programação, isso deixa o pensamento computacional deles mais desenvolvido,

Tabela 5.1 – Quantidade de alunos que responderam corretamente no início e no fim

Aula inicial (Total: 10)		Aula final (Total: 8)	
Questão	**Alunos que responderam corretamente**	**Questão**	**Alunos que responderam corretamente**
1	2	**1**	2
2	4	**2**	5
3	5	**3**	8
4	2	**4**	4
5	5	**5**	8
6	3	**6**	7
7	8	**7**	7
8	7	**8**	8
9	0	**9**	6
10	5	**10**	5

Tabela 5.2 – Objetivos das questões do questionário e porcentagens de acerto

Objetivos das questões	**Questão**	**Acertos Pré-Teste**	**Acertos Pós-Teste**	**Diferença**
Efetuar uma sequência de passos	1	20%	25%	+5%
Identificar uma sequência correta de passos	2	40%	62,5%	+22,5%
Interpretar e executar instruções com variáveis e estrutura de repetição	3	50%	100%	+50%
Identificar e executar operações com operadores lógicos e relacionais	4	20%	50%	+30%
Efetuar operações de adição e multiplicação	5	50%	100%	+50%
Resolver equação de 1 grau	6	30%	87,5%	+57,5%
Realizar equação de 1 grau	7	80%	87,5%	+7,5%
Aplicar raciocínio lógico na resolução do problema	8	70%	100%	+30%
Aplicar raciocínio lógico na resolução do problema	9	0%	75%	+75%
Aplicar raciocínio lógico na resolução do problema	10	50%	62,5%	+12,5%

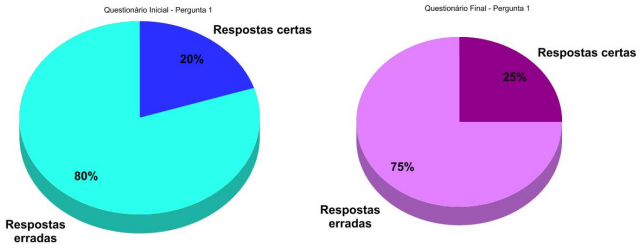


Figura 5.4 – Questionário inicial e final - Pergunta 1

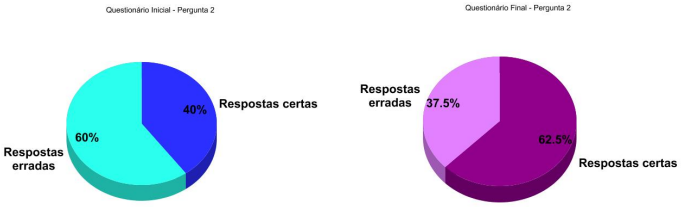


Figura 5.5 – Questionário inicial e final - Pergunta 2

sendo capazes de encontrar resolução para problemas de forma mais prática e rápida.

Outro questionário que foi feito, foi o questionário final, que se tratava de notas de 0 a 10 em vários quesitos, onde zero era extremamente negativo e dez era extremamente positivo. Todos os envolvidos da escola que ajudaram também participaram desse questionário final, que anteriormente já foram mostradas as perguntas que havia nele na tabela 4.3 para que fossem dadas as notas. Além dessas perguntas da tabela, havia também uma parte onde eles podiam deixar comentários sobre o projeto, seguem trechos de alguns comentários dos alunos nas respostas:

"Foi muito legal participar do Code Club, o scratch é muito legal de mexer, e a aula não era enjoada, foi legal aprender, foi bom pra mim".

"Eu nunca tinha mexido com nada de programação antes, e começou bem do zero, e isso me ajudou muito pra eu poder conseguir".

"Eu só entendi que eu tinha mesmo aprendido alguma coisa diferente quando eu fiz a prova de novo, vi que eu dava conta de responder as perguntas que respondi antes e

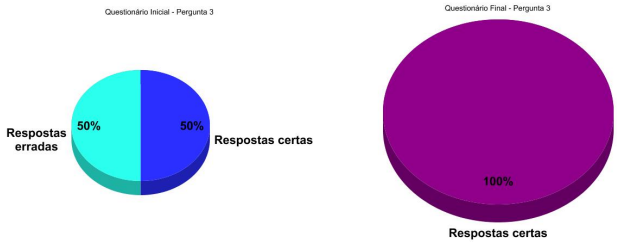


Figura 5.6 – Questionário inicial e final - Pergunta 3

Figura 5.7 – Questionário inicial e final - Pergunta 4

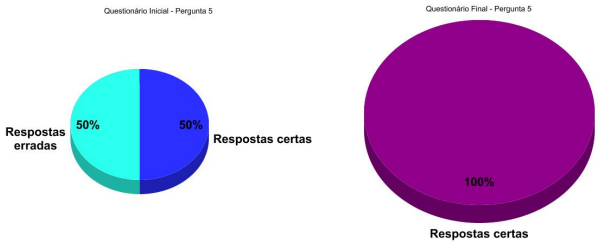


Figura 5.8 – Questionário inicial e final - Pergunta 5

Figura 5.9 – Questionário inicial e final - Pergunta 6

Figura 5.10 – Questionário inicial e final - Pergunta 7

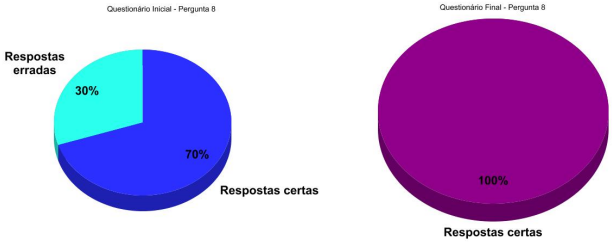


Figura 5.11 – Questionário inicial e final - Pergunta 8

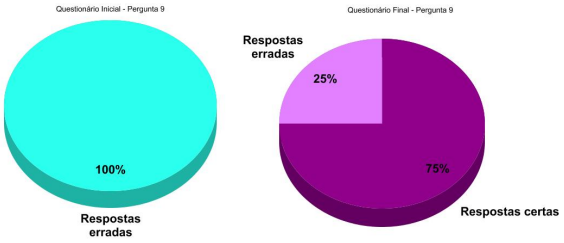


Figura 5.12 – Questionário inicial e final - Pergunta 9

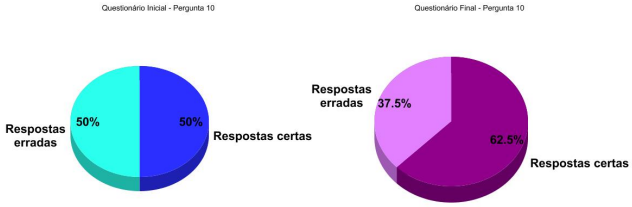


Figura 5.13 – Questionário inicial e final - Pergunta 10

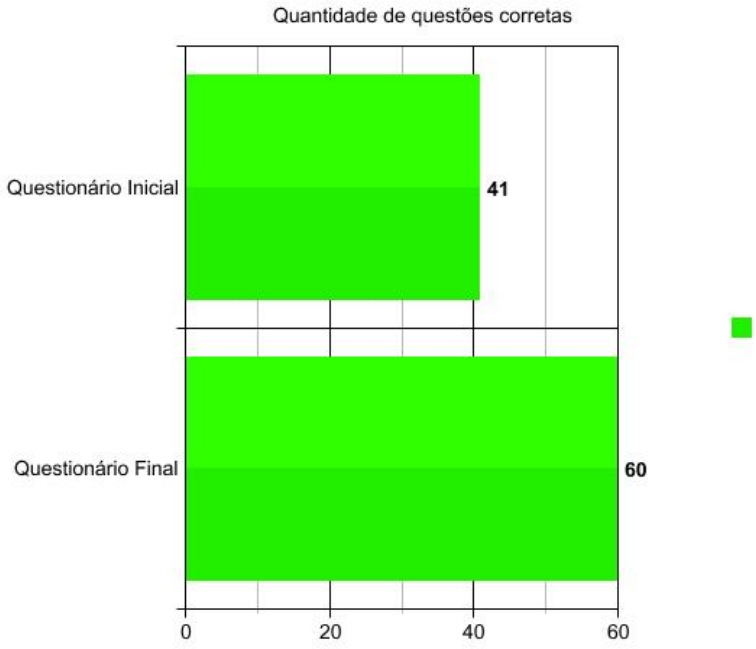


Figura 5.14 – Quantidade de questões corretas no questionário de aprendizagem inicial e final

não tinha conseguido".

"Quando eu ouvi 'projeto de programação' na escola, fiquei curiosa, mas achei muito difícil e achei que não ia conseguir acompanhar. No começo, pensei em desistir, mas aí fui tentando só mais uma aula, e quando eu vi, queria ir até o fim, o scratch é muito legal e se estudar um pouco, não é tão difícil".

"Pra mim foi muito difícil, mas foi muito legal, o scratch tem tantos desenhos que deixou tudo mais fácil pra eu aprender a entender o que eu estava fazendo".

"O Code Club mostrou pra gente que a internet não é só pra rede social, que podemos usar ela de outros jeitos, eu adorei aprender de programação, e não quero parar".

"Na primeira aula, quando eu ouvi que a gente ia programar, já queria desistir, porque parecia muito difícil, mas na outra aula eu entendi que a gente ia usar o scratch e que ele não é difícil, e foi muito legal aprender um pouco de programação".

E aqui seguem trechos da equipe e dos professores:

"Esse projeto que a UFT proporcionou pra nossa escola foi muito gratificante, ver nossas crianças aprendendo mais sobre isso, esse ensino de programação abre a cabeça das crianças".

"As crianças aqui tem uma idade diferente das séries, ficamos com medo de não ser proveitoso, mas o método é muito simples e objetivo, e conseguiu entreter até os alunos mais desinteressados".

"Foi uma experiência diferente para nossa escola, o interesse dos alunos foi muito grande, e a maneira que as aulas são dadas conseguiu aguçar a vontade de eles quererem fazer mais".

No geral, os comentários que foram deixados pelos alunos e pelos envolvidos no projeto, foram agradecendo e falando que gostaram do projeto, mas houveram comentários negativos e críticas construtivas em 25% das respostas do total. Essas respostas, em sua maioria, falavam sobre a falta de infra-estrutura para receber um projeto assim em uma escola sem computadores, e reclamações sobre o horário da aula ser cedo demais (era as 8 horas da manhã).

5.2 Avaliação por Inspeção

Como foi visto anteriormente, só uma interface bonita não é suficiente, precisa-se de que ela atenda alguns critérios que visam facilitar a vida do usuário, é importante que o usuário não tenha que pensar muito para utilizar a aplicação. Com isso foi feita a avaliação por inspeção nesse projeto, para que seja mostrado quais são as heurísticas de uma boa interface que estão sendo atendidas. Para a avaliação por inspeção foram escolhidas quatro telas para que pudessem ser feitas as avaliações das suas interfaces. As telas escolhidas foram:

1) Scratch - Módulo 1 - Projeto 1

Figura 5.15 – Tela do Scratch Módulo 1 - Banda de Rock

2) Scratch - Módulo 1 - Projeto 2
3) Scratch - Módulo 1 - Projeto 3
4) Scratch - Módulo 1 - Projeto 4

Todas as telas estão relacionadas ao primeiro módulo do scratch. O scratch é uma linguagem de programação visual que possibilita a criação de jogos e animações pela própria criança, por meio de combinações de blocos lógicos que se "arrastam". Todas as aulas do módulo 1 são feitas no Scratch e cada um dos projetos têm um objetivo.

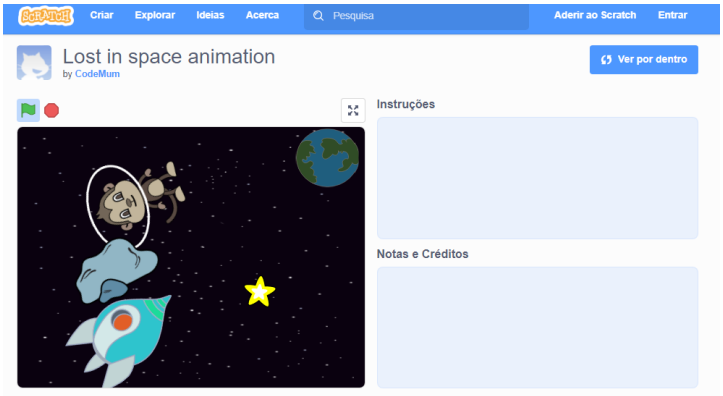


Figura 5.16 – Tela do Scratch Módulo 1 - Perdido no Espaço

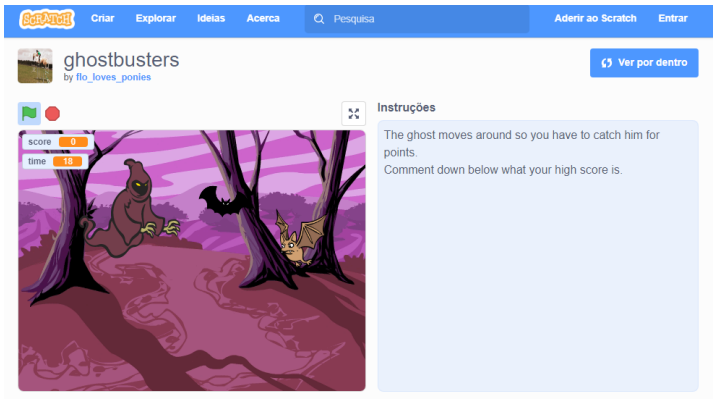


Figura 5.17 – Tela do Scratch Módulo 1 - Caça Fantasmas

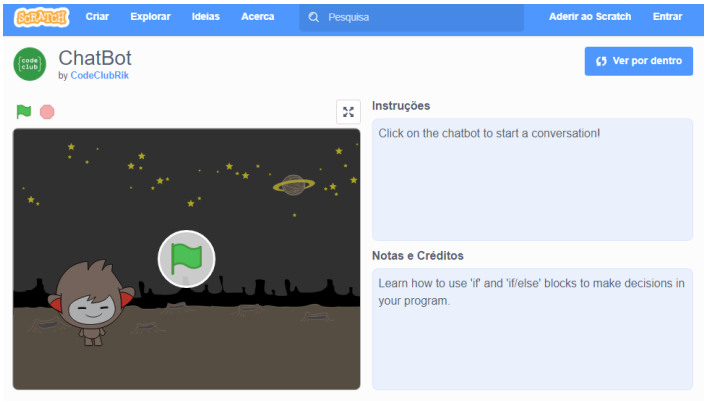


Figura 5.18 – Tela do Scratch Módulo 1 - Robô Falante

6 CONCLUSÕES E TRABALHOS FUTUROS

6.1 Limitações

Para que esse projeto pudesse acontecer, foi preciso se adequar a realidade da escola que não dispunha de um notebook para cada aluno. Eles são acostumados a fazer qualquer atividade fora da sala de aula, no próprio celular, e o fato de eles terem que dividir notebook, trabalharem juntos, esperarem sua vez, com certeza foi uma limitação no trabalho, se houvesse mais computadores, talvez eles teriam tido mais tempo de testar as funções que aprendiam.

Outra limitação que é possível citar que houve nesse projeto foi o fato de ser somente um voluntário com conhecimento de programação para os dez alunos. Em uma hora de aula apenas, torna-se muito difícil atender a tantos alunos e focar na dificuldade de cada um, achar erros, e dar a atenção que cada um deles realmente queria. Em outros clubes há diversos voluntários, e mostra-se muito mais proveitoso para o aprendizado do aluno.

O fato das aulas serem no contra-turno do horário escolar dos alunos atrapalhou também, pois as aulas eram muito cedo e era corriqueiro os alunos chegarem atrasados, às vezes atrasavam mais do que meia hora (em uma aula de uma hora de duração), e isso prejudicou o aprendizado deles, pois quando perdem o começo da aula, era preciso reiniciar tudo com ele, pra poder conseguir acompanhar, e acabava atrapalhando os outros também.

6.2 Conclusões

Ao final da aplicação do método, é possível concluir que o trabalhou atinge seus objetivos propostos por meio da metodologia Code Club: a aplicação na escola pública mostrou que apesar das diversas limitações, se o método for seguido da maneira que ele foi planejado, e o voluntário estiver disposto a ensinar realmente da maneira como foi pré-definido pela metodologia, ao fim da aula todos os alunos concluem os exercícios que são propostos, deixando-os preparados para o módulo seguinte, já que há um "gancho" entre uma aula e outra. Alguns alunos levam mais tempo do que os outros, alguns alunos têm mais dificuldade que os outros, mas todos os que não desistiram, concluíram todas as tarefas propostas.

A metodologia ressalta que ela foi desenvolvida para ser de uma forma lúdica e descontraída. E os exercícios são propostos de uma maneira que as crianças realmente parecem ter vontade de fazer o que foi proposto, já que no começo são tarefas que elas julgam fáceis. A lógica de programação que eles aprendem no scratch, é feita com dese-

nhos, encaixes de quebra cabeça, palavras no português com comandos, assim, torna-se mais compreensível para elas entenderem como funciona a lógica de programação de um laço de repetição, por exemplo. Portanto, com esse projeto conclui-se que por meio dessa metodologia o aprendizado da lógica de programação foi facilitado sim, devido ao fato do método ser lúdico e interessante para as crianças, assim, houve o interesse das crianças e as fez analisar a melhor maneira pra montar uma boa cena para o resultado final.

Acredita-se que o objetivo principal deste projeto - fazer com que os alunos estudassem lógica de programação - foi atingido, uma vez que os alunos conseguiram desenvolver sozinhos todas as atividades propostas, alguns com mais dificuldades e outros com menos, mas todos concluíram.

6.3 Trabalhos Futuros

Para trabalhos futuros, deseja-se fazer o mesmo teste, porém com uma turma maior de alunos, visando que as dificuldades, desistências, acertos e erros sejam melhor comparados. Também deseja-se fazer o teste em outras escolas públicas do Tocantins, mas tentar também fazer a aplicação da metodologia com alunos da rede privada, para fazer um comparativo se há uma diferença com alunos que tem, no geral, melhores condições. Outro ponto para trabalhos futuros é tentar fazer a aplicação do Code Club com crianças que não tenham a distorção série-idade, a fim de analisar se o desempenho seria melhor. Poderia ser abordado em um trabalho futuro também uma preparação dos alunos para participarem na OBI- Olimpíada Brasileira de Informática, visto que durante as aulas, o assunto veio à discussão várias vezes, e os alunos mostraram grande interesse em participarem futuramente, e um preparatório poderia melhorar os resultados deles na OBI.

REFERÊNCIAS

ABELSON, H.; SUSSMAN, G. J.; SUSSMAN, J. **Structure and Interpretation of Computer Programs. MIT Press, Cambridge, MA.** 1985. <http://web.mit.edu/alexmv/6.037/sicp.pdf>. Último acesso em 30/04/2018.

ALBUQUERQUE, I. A. L. **Viagem pelo cérebro para pais e filhos.** [S.l.: s.n.], 2011.

BARR, V.; STEPHENSON, C. **Bringing Computational Thinking to K-12: What is Involved and What is the Role of the Computer Science Education Community?** 2011.

BENYON, D. **Interação Humano-Computador**. 2nd. ed. [S.l.]: Pearson, 2010.

BERG, C. H. **Ferramentas para avaliação de interface humano computador em sites acessíveis.** 2016. Último acesso em 18/07/2019.

BLUMENFELD, P. C. et al. **Motivating Project-Based Learning: Sustaining the Doing, Supporting the Learning IEEE ACM**. 2011. Último acesso em 10/03/2018.

BORNELI, J. **Austrália começa substituir disciplinas por programação nas escolas**. 2015. <https://conteudo.startse.com.br/mundo/juniorboneli/australia-comeca-substituir-disciplinas-de-historia-e-geografia-por-aulas-de-programacao/>. Último acesso em 15/03/2019.

BRENNAN, K.; BALCH, C.; CHUNG, M. **Creative Computing**. 2011. <http://scratched.gse.harvard.edu/guide/files/CreativeComputing20140806.pdf>. Último acesso em 19/05/2018.

CASTRO, C. S.; VILARIM, G. de O. **Licenciatura em Computação no cenário nacional: embates, institucionalização e o nascimento de um novo curso**. 2013.

CLUB, C. **Code Club (Brasil)**. 2017.

COMPUTER Science: A curriculum for schools. 2012. <https://www.computingatschool.org.uk/data/uploads/ComputingCurric.pdf>. Último acesso em 22/05/2019.

CROSS, R.; GARDNER, T. **Simple Coding for total beginners: Book of Scratch**. 2018. <https://www.raspberrypi.org/magpi-issues/CC_Book_of_Scratch_v1.pdf>. Último acesso em 16/07/2019.

DENNING, P. J. **Great Principles of Computing**. 2003. <http://delivery.acm.org/10.1145/950000/948400/p15-denning.pdf?ip=200.129.179.187&id=948400&acc=ACTIVE%20SERVICE&key=344E943C9DC262BB%2E4013100E04B2BC4D%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&__acm__=1527510039_f297371d512977b263ed034b2d160629>. Último acesso em 15/05/2018.

FOUNDATION, R. P. **A worldwide network of volunteer-led coding clubs for children aged 9-13**. 2017.

FURBER, S. **Shut down or restart? The way forward for computing in UK schools**. 2012.

FóZ, A. **NEUROCIÊNCIA NA EDUCAÇÃO: Desafios e conhecimentos**. 2015.

GAL-EZER, J. et al. **A High-School Program in Computer Science IEEE Computer Society Press Los Alamitos, CA, USA ACM**. 1995. <https://www.openu.ac.il/personal_sites/download/galezer/high-school-program.pdf>. Último acesso em 20/03/2018.

GARNER, R. **Welcome to Code Club: UK programme that teaches children computer coding goes global**. 2013.

GILLANI, B. **Cognitive Theory and The Design of Education to Work Connection**. 2013. <http://delivery.acm.org/10.1145/2510000/2501938/a7-gillani.pdf?ip=200.129.179.187&id=2501938&acc=ACTIVE%20SERVICE&key=344E943C9DC262BB%2E4013100E04B2BC4D%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&__acm__=1527519791_2b045594af81c7bb9e9ac6776d19615f>. Último acesso em 10/05/2018.

GONçALVES, A. T. B. N. andHerlanio L. **Método de Avaliação de Comunicabilidade da Engenharia Semiótica: um estudo de caso em um sistema Web**. 2013. Último acesso em 20/07/2019.

GOV.UK. **The national curriculum**. 2017.

GROSS, A. **25 states now require computer science for high school graduation**. 2015. <https://www.educationdive.com/news/25-states-now-require-computer-science-for-high-school-graduation/391113/>. Último acesso em 23/05/2018.

JONES, S. P. **Computer Science as a School Subject**. 2013.

LUCRECIO, A. I. **Comparação e aplicação de diferentes ferramentas para ensino de programação para crianças**. 2016. <https://repositorio.ufsc.br/xmlui/bitstream/handle/123456789/173903/TCC-FINAL.pdf?sequence=1&isAllowed=y>. Último acesso em 13/06/2019.

MALONEY, J. et al. **The Scratch Programming Language and Environment**. 2010. <http://web.media.mit.edu/~jmaloney/papers/ScratchLangAndEnvironment.pdf>. Massachusetts Institute of Technology.

NASCIMENTO, C. da S. **Introdução ao ensino de lógica de programação para crianças do ensino fundamental com a ferramenta scratch**. 2015. <https://ufrr.br/liead/index.php?option=com_phocadownload&view=category&download=113:introducao-ao-ensino-de-logica-de-programacao-para-criancas-do-ensino-fundamental-com-a-ferram id=21:polo-rorainopolis&Itemid=309>. Último acesso em 01/05/2019.

PRATES, R. O.; BARBOSA, S. D. J. **Avaliação de Interfaces de Usuário – Conceitos e Métodos**. 2015. Último acesso em 18/07/2019.

PRETZ, K. **Computer Science Classes for Kids Becoming Mandatory**. 2014. <http://theinstitute.ieee.org/career-and-education/education/computer-science-classes-for-kids-becoming-mandatory>. Último acesso em 22/05/2018.

PROJECT, T. L. **Super Scratch Programming Adventure!** [S.l.: s.n.], 2013. Learn to program by making cool games.

ROCHA, H. V. da; BARANASKAS, M. C. C.

SAELI, M. et al. **Teaching Programming in Secondary School: A Pedagogical Content Knowledge Perspective**. 2010. <https://www.mii.lt/informatics_in_education/pdf/INFE177.pdf>. Último acesso em 28/05/2018.

SANTOS, A. P. O. dos. **Metodologias e Ferramentas para Avaliação da Qualidade de Sistemas Web de Código Aberto com Respeito à Usabilidade**. 2008. Último acesso em 20/06/2019.

SCHMIDT, E. **James Mac Taggart Lecture**. 2011. <https://www.youtube.com/watch?v=hSzEFsfc9Ao>. Último acesso em 14/05/2018.

SCRATCH Cards. 2012. <https://scratch.mit.edu/info/cards/>. Último acesso em 10/05/2018.

SCRATCH: Create stories, games, and animations. Share with others around the world. 2017. <https://scratch.mit.edu//>. Último acesso em 25/05/2018.

SMITH, M. **Computer Science For All**. 2016.

SMITH, N.; SUTCLIFFE, C.; SANDVIK, L. **Code Club: Bringing Programming to UK Primary Schools through Scratch**. 2014.

TUCKER, A. **A Model Curriculum for K–12 Computer Science: Final Report of the ACM K–12 Task Force Curriculum Committee**. 2003. <http://delivery.acm.org/10.1145/2600000/2593247/a1-tucker.pdf?ip=200.129.179.187&id=2593247&acc=NO%20RULES&key=344E943C9DC262BB%2E4013100E04B2BC4D%2E4D4702B0C3E38B35%2E4D4702B0C3E38B35&__acm__=1527510194_e06b04f02efca6c1d807323c6a0416fb>. Último acesso em 20/05/2018.

WHEATLEY, M. **Chicago votes to make Computer Science a mandatory subject at all public schools**. 2016. <https://www.google.com.br/search?q=Chicago+votes+to+make+Computer+Science+a+mandatory+subject+at+all+public+schools&oq=Chicago+votes+to+make+Computer+Science+a+mandatory+subject+at+all+public+schools&aqs=chrome..69i57.295j0j7&sourceid=chrome&ie=UTF-8>. Último acesso em 19/05/2018.

WING, J. M. **Computational Thinking - Communications of the ACM**. 2006. <https://cacm.acm.org/magazines/2006/3/5977-computational-thinking/fulltext>. Último acesso em 24/05/2018.

ZANATTA, A. C. **Programação de computadores para crianças - Metodologia do Code Club Brasil**. 2015. <https://repositorio.ufsc.br/xmlui/bitstream/handle/123456789/158762/TCC_Andrei_TIC_2015.pdf?sequence=1&isAllowed=y>. Último acesso em 16/06/2019.

A APÊNDICE 1

Scratch 1

Banda de Rock

code club

All Code Clubs must be registered. By registering your club we can measure our impact, and we can continue to provide free resources that help children learn to code. You can register your club at codeclubworld.org.

Introdução

Neste projeto você vai aprender como codificar os seus próprios instrumentos musicais!

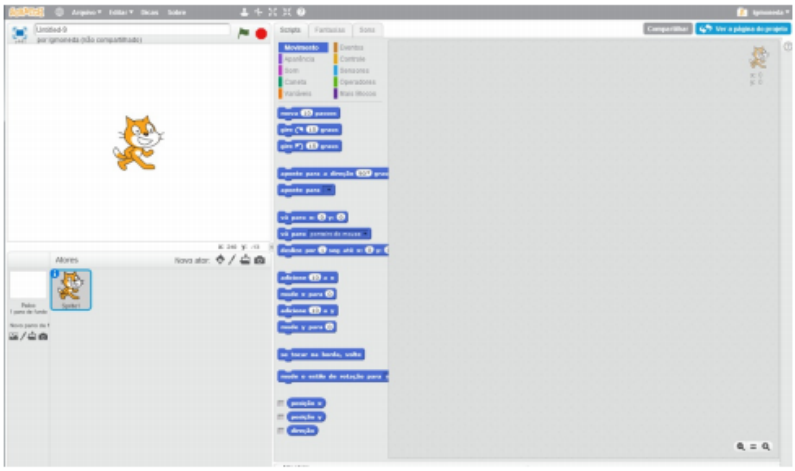
Lista de atividade	Siga estas INSTRUÇÕES uma a uma
Teste seu projeto	Clique na bandeira verde para TESTAR
Salve seu projeto	Certifique-se de SALVAR seu trabalho

Passo 1: Atores (Sprites)

Antes que você possa começar a animar, é preciso adicionar uma 'coisa' para animar. No Scratch, estas "coisas" são chamados sprites ou fantasias.

Lista de atividades

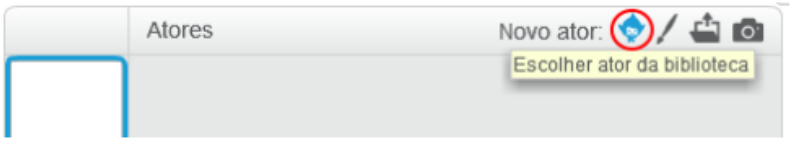
- Primeiro, abra o editor do Scratch. Você pode encontrar o editor Scratch on-line em jumpto.cc/scratch-new. A aparência é como esta:



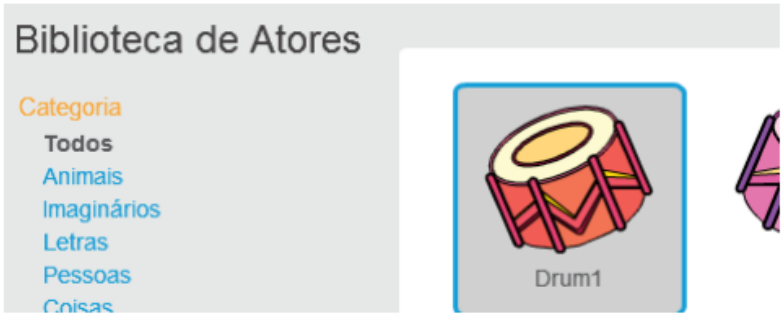
- O sprite que você pode ver (um gato), é o mascote do Scratch. Vamos nos livrar dele, clicando com o botão direito e, em seguida, clicando em 'apagar'.

- Em seguida, clique 'Escolher ator da biblioteca' para abrir uma

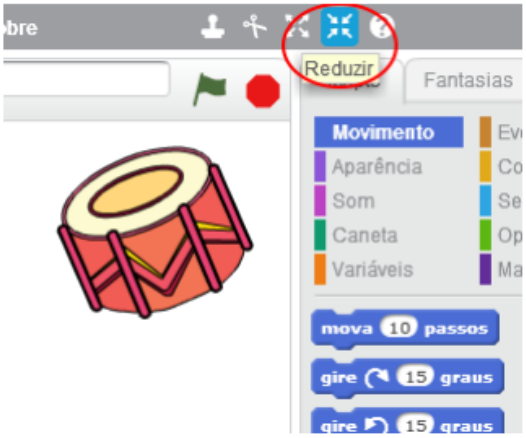
lista de todos os sprites de Scratch.



- Role para baixo até ver um tambor. Clique no tambor e em 'OK' para adicioná-lo ao seu projeto.

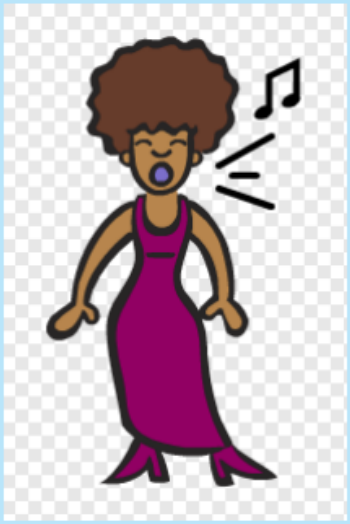


- Clique no ícone 'reduzir', e em seguida, clique no tambor algumas vezes para reduzir o seu tamanho.



Desafio: Mudando a Fantasia de seu cantor

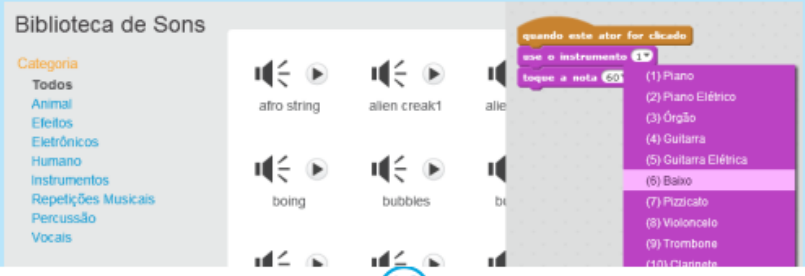
Você pode fazer com que seu cantor pareça que está cantando quando você clicar nele? Se você precisar de ajuda, você pode usar as instruções para a criação de um tambor acima.



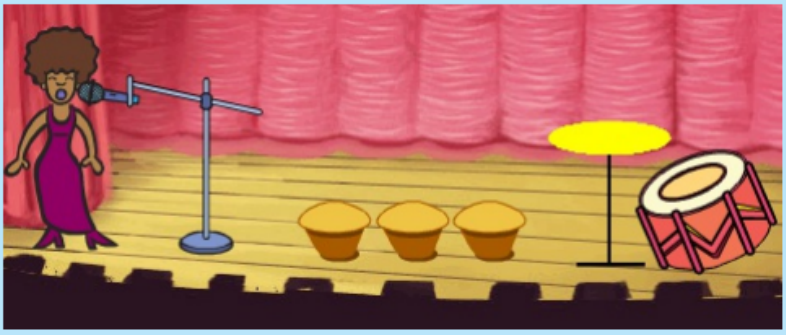
Lembre-se de testar se o novo código funciona!

Desafio: Faça a sua própria banda

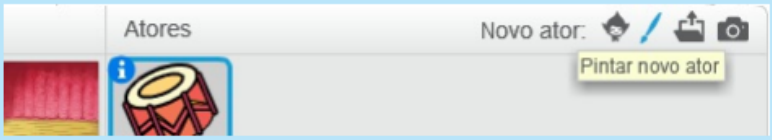
Use o que você aprendeu neste projeto para fazer a sua própria banda! Você pode criar qualquer instrumentos que você gosta, mas ouça os sons e instrumentos para ter algumas idéias.



Os seus instrumentos não tem que fazer sentido. Por exemplo, você poderia fazer um piano feito de bolinhos!



Bem como a utilização de fantasias existentes, você também pode criar suas próprias fantasias.



Se você tiver um microfone você pode gravar os seus próprios sons, ou mesmo usar uma webcam para fazer seus instrumentos!

Salve seu projeto