UNIVERSIDADE PAULISTA - UNIP
CIÊNCIA DA COMPUTAÇÃO

JEAN PAULO ATHANAZIO DE MEI
D4457G3
THIAGO DA SILVA RIBEIRO
N176059

APLICAÇÃO DE TÉCNICAS DE INTELIGÊNCIA ARTIFICIAL PARA A INCLUSÃO DE DEFICIENTES VISUAIS NO UNIVERSO DOS JOGOS DIGITAIS.

TRABALHO DE CONCLUSÃO DE CURSO

SÃO JOSE DO RIO PRETO
2020

JEAN PAULO ATHANAZIO DE MEI
D4457G3
THIAGO DA SILVA RIBEIRO
N176059

APLICAÇÃO DE TÉCNICAS DE INTELIGÊNCIA ARTIFICIAL PARA A INCLUSÃO DE DEFICIENTES VISUAIS NO UNIVERSO DOS JOGOS DIGITAIS.

Trabalho de conclusão de curso apresentado como parte dos requisitos para obtenção do título de graduação em Ciência da Computação, curso que aborda áreas da Computação em geral, apresentado a Universidade Paulista UNIP – câmpus JK de São José do Rio Preto.

Orientador: Cícero Marcelo de Oliveira
Universidade Paulista - UNIP

SÃO JOSE DO RIO PRETO
2020

Eu Jean Paulo dedico, por sempre estarem comigo no pior e no melhor momento de minha vida a minha irmã e finada mãe: Taina Viviane Leme Athanazio Silva e Maria Helena Leme Athanazio, foi uma dádiva estar com vocês.

Eu Thiago Ribeiro dedico este trabalho aos meus pais, pois é graças ao seu esforço que hoje posso concluir o meu curso.

AGRADECIMENTOS

Primeiramente, eu, Jean Paulo, quero agradecer a Deus pois sem ele não estaria aqui. Tenho um déficit de atenção que por muitos anos foi severo, mas Ele me agraciou com a cura parcial desse problema que tanto me afligia e me dando um índice de inteligência acima da média. Só estou aqui por que Ele permitiu.

Em Segundo lugar, quero agradecer a minha familía que tanto vem me apoiando nesses últimos anos, desde de a época que estive na escola, ela me guia e me conforta quando estou passando por dificuldades.

Quero agradecer também ao sistema de voucher federal que permitiu com que eu fizesse quatro anos de ciência da computação em universidade privada sem precisar pagar mensalidade, e que iniciativas assim cresçam cada vez mais favorecendo esse enorme público.

Descobri uma paixão que não conhecia, a paixão pela programação, quero agradecer a todos os meus professores por ter iluminado meus caminhos com a luz do conhecimento e das ideias! Somente elas podem iluminar a escuridão presente no nosso mundo.

Eu, Thiago Ribeiro, quero agradecer a **UNIP** e aos meus professores, que me passaram seus conhecimentos e experiência na área de computação.

“ideais e somente ideais
podem iluminar a escuridão.”
Ludwig Von Mises

RESUMO

O Resumo é um elemento obrigatório em tese, dissertação, monografia e TCC, constituído de uma seqüência de frases concisas e objetivas, fornecendo uma visão rápida e clara do conteúdo do estudo. O texto deverá conter no máximo 500 palavras e ser antecedido pela referência do estudo. Também, não deve conter citações. O resumo deve ser redigido em parágrafo único, espaçamento simples e seguido das palavras representativas do conteúdo do estudo, isto é, palavras-chave, em número de três a cinco, separadas entre si por ponto e finalizadas também por ponto. Usar o verbo na terceira pessoa do singular, com linguagem impessoal, bem como fazer uso, preferencialmente, da voz ativa. Texto contendo um único parágrafo.

Palavras-chave: Palavra. Segunda Palavra. Outra palavra.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE TABELAS

LISTA DE SÍMBOLOS

LISTA DE ALGORITMOS

SUMÁRIO

1 INTRODUÇÃO

Durante o curso de ciência da computação são transmitidos conceitos interessantes como: “agentes inteligentes”, “algoritmos genéticos”, “inteligência computacional” e “redes neurais”. Também foi possível por meio de uma APS desenvolver um jogo do zero, situações que inspiraram ao tema deste TCC. Com a ajuda de jogos e inteligência artificial é possível aumentar a inclusão dos deficientes visuais aos estudos acadêmicos de todos os tipos e, para o escopo deste trabalho, o tema abordado será o ensino de história através de jogos de texto ditados.

1.1 Objetivos

1.1.1 Gerais

- Realizar levantamento bibliográfico relacionado com a inclusão de deficientes visuais em jogos digitais;
- Analisar ferramentas que unidas, possibilitem o desenvolvimento de jogos digitais voltados para o público com deficiências visuais;
- Fortalecer e expandir conceitos de inteligência computacional.
- Contribuir para trabalhos de pesquisa que queiram usar o modelo de conversação automatizada “GPT-3”.
- Atrair índividuos que não são da area da técnologia a conhecer assuntos novos da computação atráves de temas comuns a eles abordado nesse trabalho.

1.1.2 Específicos

Desenvolver um jogo com o intuito de ajudar deficientes visuais e neurológicos a aprender a diciplina escolar de História, a idéia central é que seja possível ele jogar uma faze de um jogo escrito onde será possível interagir em momentos históricos, para o escopo desse trabalho somente a primeira faze será abordada que será sobre a pré história.

O principal objetivo é mostrar que é possível criar aplicativos para o ensino de qualquer tipo de matéria visando a inclusão dos deficientes e aumentar.

1.2 JUSTIFICATIVA

Na atualidade, o mundo está carente de ferramentas gratuitas e pagas que forneçam educação de qualidade para pessoas deficientes ou que possuem défict de atenção, tendo em vista que tais pessoas sentem dificuldade para absorver o conteúdo.1

Além disso, informações sobre tecnologia da informação no geral, em especial criação e implementação do front end e back end e a inteligência artificial ainda são temas muito carentes

de informações ao publico leigo, podendo ser unidas em um projeto ligando as palavras chaves “Jogos” e “Deficientes visuais”, podendo atrair um público até então sem acesso a essa junção.

Pretende-se também mostrar como foi o caminho de desenvolvimento com ajuda do versionamento de código, utilizando das técnlogias **GIT** e **GIT HUB**.

Segundo (**citação**), **GIT** é um sistema de controle de versões distribuído que pode ser usado para registrar o histórico de edições de qualquer tipo de arquivo, já **GIT HUB** é uma plataforma de hospedagem de código-fonte e arquivos com controle de versão usando o **GIT**).

1.3 METODOLOGIA

Será desenvolvido um sistema web que juntamente com a tecnologia do GPT-3 irá emular um jogo de conversação em linguagem natural, o qual terá como tema um professor de história particular que tenta ensinar seu aluno que possuí défict de atenção por meio do RPG de mesa.

1.4 ORGANIZAÇÃO DO TRABALHO

Os capítulos desta pesquisa estão organizados da seguinte forma:

- Capítulo 1 — Fundamentação teórica. O conceito de ... são descritos no Capítulo 1.
- Capítulo 2 — Desenvolvimento.
- Capítulo 3 — Testes.
- Capítulo 4 — Resultados e conclusão.

2 FUNDAMENTAÇÃO TEORICA

Neste capítulo serão apresentados conceitos fundamentais para o trabalho, permitindo que os leitores entendam de fato o como esse sistema foi desenvolvido e quais são as melhorias possíveis.

Dessa forma, procura-se responder aos seguintes questionamentos:
O que é o ramo da computação?
O que é o GIT?
O que é o GITHUB?
O que é UIX e Experiencia do usuário?
O que é o HMTL?
O que é o CSS?
O que é a lógica de programação?
O que é um algoritmo?
O que é o JavaScript Vanilla?
O que é Phyton?
O que é aprendizado de máquina?
O que é uma Rede Neural?
O que é o Tensor Flow?

2.1 O ramo da computação e seus desafios.

Em termos gerais, a computação pode ser definida como uma atividade realizada para processar as informações por meio de computadores. Esta atividade inclui o desenvolvimento de software e hardware também.

A utilização da tecnologia de computador para completar a tarefa é conhecida como computação.

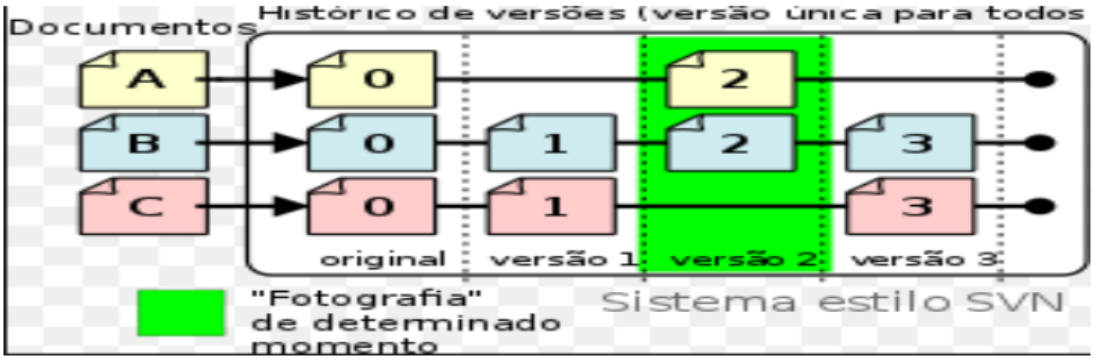
Em um sentido moderno, usamos computadores para uma infinidade de finalidades - uma das quais é a crescente variedade de opções de entretenimento. No entanto, o hardware de computação moderno foi desenvolvido a partir de aparelhos muito maiores e muito mais básicos, nascidos principalmente da necessidade. O dispositivo computacional original era o ábaco, que foi supostamente inventado na antiga Babilônia por volta de 2.300 a.C. Algum tempo depois (por volta de 100 a.C.), o mecanismo de Antikythera foi usado para prever movimentos astronômicos com décadas de antecedência. Leia mais sobre a história da computação - Uma breve história da computação.

(MIKE, 2020)

É com isto que lida a teoria da computação, subcampo da ciência da computação e da

2.2.2 CONTROLE DE VERSÃO COM O GIT

Figura 1 – Ilustração de o funcionamento do versionamento



matemática.

2.2 O GIT

2.2.1 VANTAGENS DE USAR O GIT

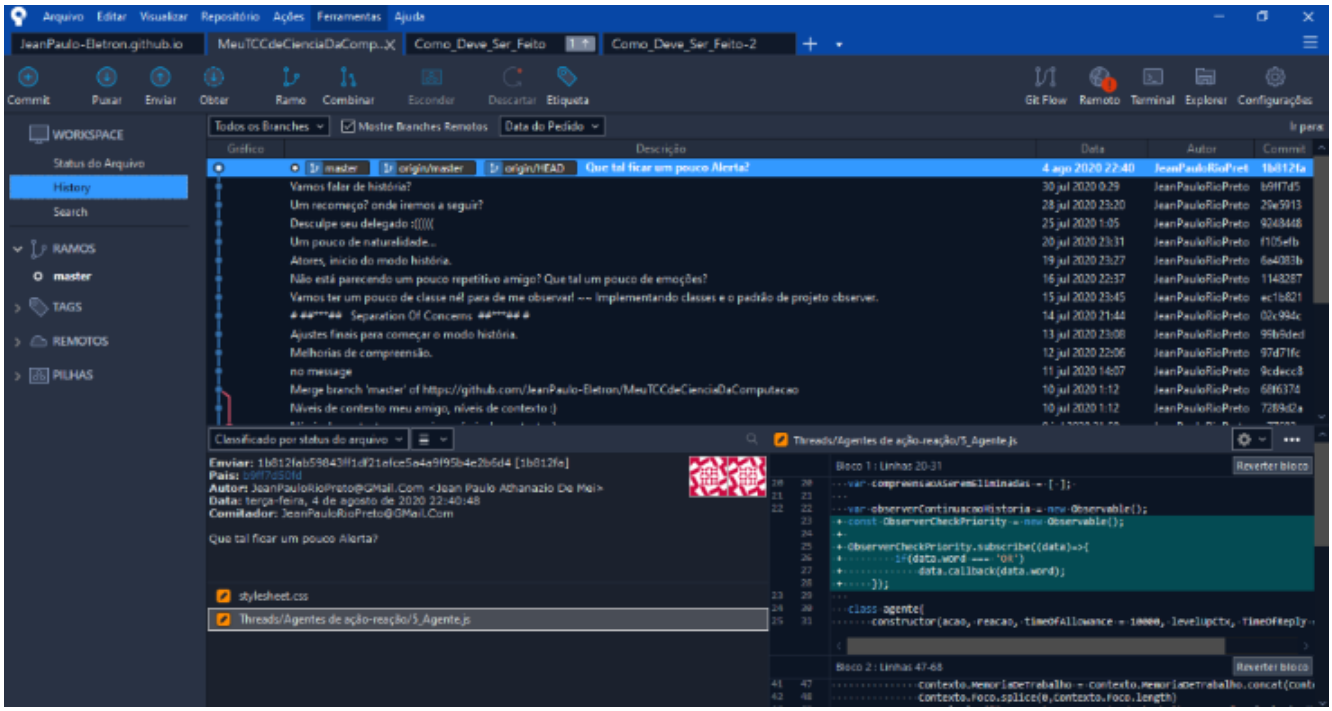
Imagine que você esteja escrevendo um texto importante, escrevendo e apagando várias e várias vezes. Agora imagine que você tenha alterado seu texto demais e queira que ele volte a ser o que era antes, então você queria ter uma máquina do tempo, como a **TARDIS** da série cientifica **Doctor Who** e com ela você possa viajar no tempo e reescrever o texto a um ponto do passado. Isso é o que o **GIT** tenta fazer, ou seja, da mesma forma que a Tardis, ele grava todas as alterações dos arquivos de texto rastreados por ele, sendo possível realizar "**Commits**" que nada mais é que marcos dessas alterações, como se fosse uma data importante que algo relevante aconteceu, sendo possível voltar a esse ponto depois.

O **GIT** é um sistema de controle de versão distribuído gratuitamente e de código aberto. Escrito para gerenciar o desenvolvimento do sistema operacional **Linux** de código aberto, ele permite que grandes equipes de programadores trabalhem de forma independente em suas próprias cópias do código, rastreiem alterações com granularidade linha por linha, mesclem essas alterações de volta no repositório principal e reverta as alterações conforme o necessário. (BITBUCKET, 2020)

Para utiliza-lo, deve-se criar um repositório, clonar esse repositório e dar um **pull** (puxar) e fazer suas alterações no arquivo, devendo-se fazer um **commit** dessas alterações para marcar um revisão, depois de realizar todos os **commits** necessários com todas as revisões necessárias o usuário deve fazer o **push** (enviar) para o repositório para que as alterações e revisões vá para o repositório.(GITHOWTO, 2020)

Para tanto, conta-se com a ajuda de um programa com uma interface visual de controle de versão (também conhecido como controle de revisão), o **scv** (sistema de controle de versão). ele mostra as linhas adicionadas com um destaque em verde e as linhas removidas com um destaque em vermelho, caso altere uma linha já existente ele coloca consecutivamente a remoção

Figura 2 – Imagem do SourceTree em execução



Para este TCC o SourceTree foi o principal programa utilizado para o versionamento, juntamente com o GITHUB DESKTOP.

da linha antiga e a adição da linha nova. (SOURCETREE, 2020)

Vantagens de usá-lo:

- Trata-se de um sistema de controle de versão distribuído com um repositório do GitHub e cópias locais nas máquinas. Dessa forma, torna-se possível trabalhar com o projeto simultaneamente com outro profissional.
- Será possível saber o que foi modificado em cada revisão e, desa forma, possibilita visualizar a evolução do projeto e diagnosticar bugs mais facilmente ou desfazer as alterações que ocasionaram o erro.
- Refazer parte do código de modo correto, eliminando os erros anteriormente citados.
- Saber quem realizou as alterações, bem como sugerir a correção necessária.
- Em situações que dois programadores alterem o mesmo arquivo o sistema de controle de versão junta essas alterações através do merge e, caso alterem a mesma linha, o programador informa a ferramenta apropriada qual a linha correta. (SILVA, 2020)

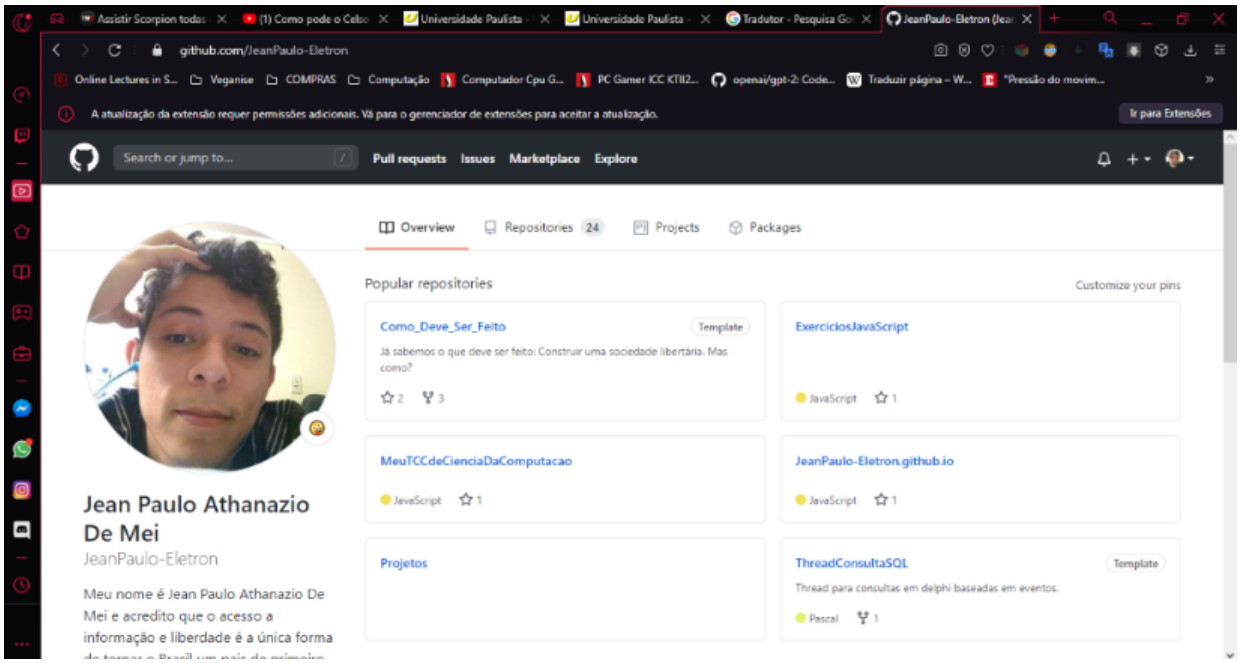
2.3 O GITHUB

O **GitHub** é uma plataforma que fornece repositórios para armazenar arquivos e alterações nesses arquivos:

Muitos projetos **Open Source** foram criados a partir dele e, com esses projetos, os programadores podem contribuir para a comunidade e ainda ter visibilidade para o mercado, já que muitas empresas procuram programadores através de seus projetos no **GitHub** e suas contribuições em

projetos **Open Source**.

Figura 3 – Imagem do GITHUB



2.3.1 README

É um arquivo que é exibido seu conteúdo de forma sumária ao entrar no repositório.

2.4 EXPERIÊNCIA DO USUÁRIO

Experiência do usuário (UX), trata da interação do usuário com um determinado produto, sistema ou serviço cujo resultado gera uma percepção positiva ou negativa. Segundo Norman, UX envolve não somente aspectos relacionados ao design (hardware, software, interface, usabilidade, facilidade de busca etc), mas também destaca os aspectos afetivos e experienciais, significativos e valiosos de interação humano-computador e propriedade do produto. A experiência do usuário é de natureza subjetiva, pois é sobre a percepção e pensamento individual no que diz respeito ao sistema. Ela é também dinâmica, pois é constantemente modificada ao longo do tempo, devido à evolução das circunstâncias e inovações. (RAMOS et al., 2016)

Um conceito importante no design UX como os usuários formam experiências. Quando o usuário encontra um produto, forma uma impressão momentânea, que evolui ao longo do tempo. (RAMOS et al., 2016)

Neste processo, a percepção, ação, motivação e cognição do usuário se integram para formar uma história memorável: chamada “experiência do usuário”. Esse processo provoca respostas emocionais, que determinam em grande parte se a experiência será considerada positiva ou negativa. (RAMOS et al., 2016)

2.5 HTML

Antes de falar sobre HTML será necessário trazer à tona o conceito de camadas de desenvolvimento:

2.5.1 CAMADAS DE DESENVOLVIMENTO

Figura 4 – Imagem do SourceTree em execução

Para este TCC será utilizado o SourceTree.

Separation of concerns(Separação de conceitos): antes de explicar o que são as camadas de desenvolvimento é importante conhecer o conceito de "separação de conceitos", ela se refere a um princípio de projeto que permite a separação de um sistema em diferentes módulos. Ao se combinar esses módulos tem-se o software, que pode ser uma página da web um hardware ou softwares desktop. Essa separação é fortemente recomendável quando o software envolve uma complexidade muito alta e atende diferentes departamentos em uma única solução, com diversas regras de negócio diferentes, como módulo contábil, financeiro, rh, custos, controle de estoque, vendas... Se ele foi programado de modo desacoplado(ou seja a alteração no código de um módulo não influencia no outro), definimos o software como modularizado, já que é constituído de diferentes partes. Cada módulo pode ser alocado para uma equipe diferente de desenvolvimento, capacitada nas respectivas regras de negócio. Assim, além de facilitar a manutenção, cada módulo poderá ser atualizado independentemente dos outros módulos. O delphi, por exemplo, propicia a modularização do software através de arquivos bpl para simplificar a atualização. Programadores web praticam a separação de conceitos indiretamente. (CELESTINO, 2020)

Para construir uma página, normalmente quatro linguagens são utilizadas, cada uma com um propósito em particular:

- HTML
- CSS
- JAVASCRIPT
- BACKEND(PHYTON)

nas quais poderemos intitulá-las como agentes de "conceitos".
O html, é utilizado para descrever, dar significado, organizar e formatar o conteúdo de uma página, ou seja cada item/objeto contido nela estará lá, uma analogia seria com uma receita de bolo, você escreve a ordem correta de cada um dos ingredientes serem colocados (na página), assim o navegador consegue construir o site, porém ele não consegue modificar propriedades de um elemento/objeto da página depois de criado, e também não consegue fornecer um estilo próprio aquele objeto, essas atribuições são dadas ao css e javascript. (TORRES, 2018)

A linguagem css, por sua vez, edita a apresentação do conteúdo gerado pela linguagem html, fornecendo estilos aos componentes/objetos da página através do seu nome(id) ou através do nome do grupo ao qual aquele objeto pertence(classe), ou seja deixar ela bonita visualmente, fazendo outra analogia seria como colocar uma roupagem em cima do objeto feito pelo html. (SILVA, 2007a)

O javascript é empregado para executar scripts do lado do cliente, ou seja alterar como os objetos já existentes se comportam ou tem de propriedade e criar objetos novos ao longo do tempo, dar comportamento a página, comportando-se de forma assíncrona para alterar a exibição do conteúdo sem comunicação com o servidor. (o jquery, por exemplo, fornece funções ajax, que permitem a alteração de um componente da página sem a necessidade de recarregá-la). (PÉREZ, 2019)

O back-end ("retaguarda") de uma página, é responsável por manipular informações do banco de dados e executar operações de tratamento e envio de informações, com um servidor central ou distribuído, no escopo desse trabalho serão feitas pela linguagem phyton. (SOUTO, 2019)

2.5.2 ESTRUTURA DO HTML5

Html (abreviação para a expressão inglesa hypertext markup language, que significa linguagem de marcação de hipertexto) é uma linguagem de marcação utilizada na construção de páginas na web. (SILVA, 2007b)

Documentos html podem ser interpretados por navegadores. a tecnologia é fruto da junção entre os padrões hytime e sgml. (SILVA, 2007b)

Importante trazer a definição de hytime, que segundo (CARR et al., 1994), é um padrão para a representação estruturada de hipermídia e conteúdo baseado em tempo.

Um documento é visto como um conjunto de eventos concorrentes dependentes de tempo (como áudio, vídeo, etc.), conectados por hiperligações (em inglês: hyperlink e link). O padrão é independente de outros padrões de processamento de texto em geral. (CARR et al., 1994)

Sgml é um padrão de formatação de textos. Não foi desenvolvido para hipertexto, mas tornou-se conveniente para transformar documentos em hiper-objetos e para descrever as ligações. (CARR et al., 1994)

Marcas (tags): todo documento html possui marcadores (do inglês: tags), palavras entre parênteses angulares(símbolos de maior e de menor “>” “<”), esses marcadores são os comandos de formatação da linguagem. um elemento é formado por um nome de marcador (tag), atributos, valores e filhos (que podem ser outros elementos ou texto). os atributos modificam os resultados padrões dos elementos e os valores caracterizam essa mudança. exemplo de um elemento simples (não possui filhos):

```
<hr />
Exemplo de um elemento composto (possui filhos):

<a href="http://pt.wikipedia.org/">Wikipédia</a>
```

Exemplo de um código HTML:

Figura 5 – Imagem de um código HTML

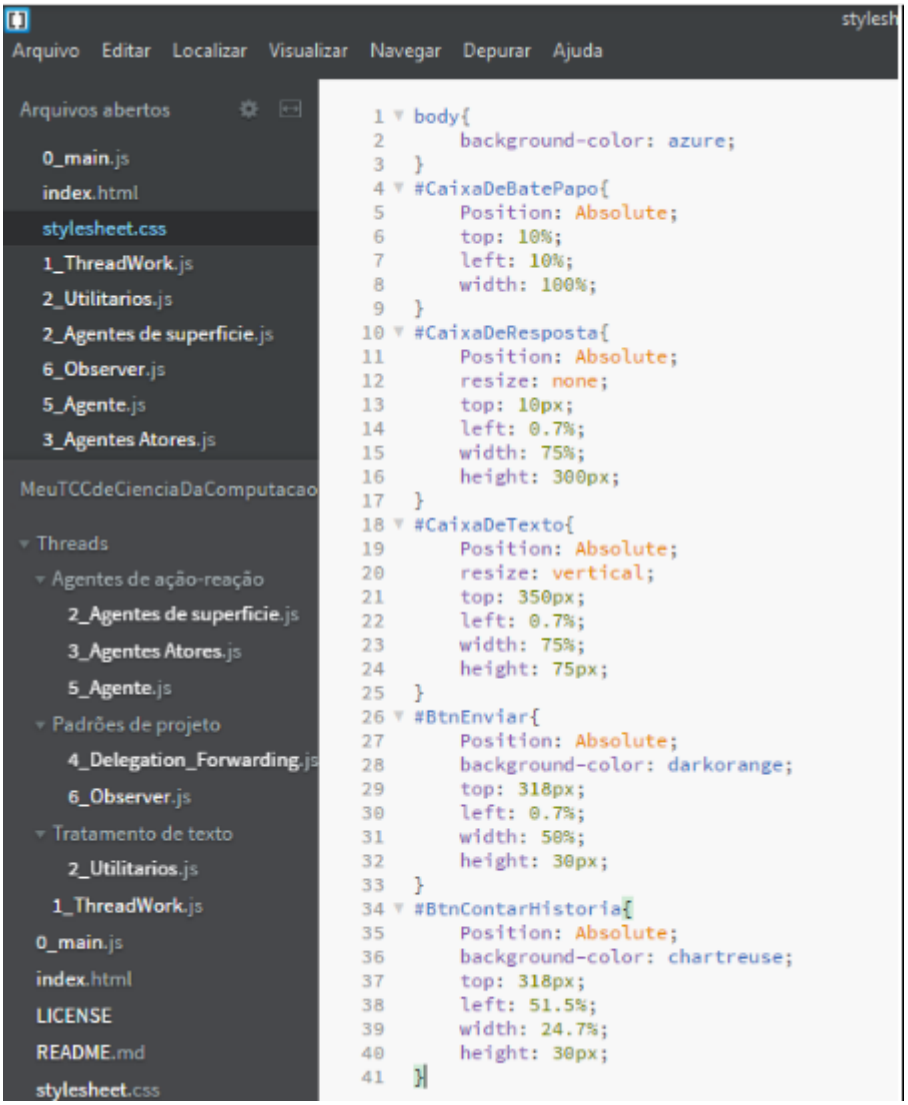
Esse código é um código HTML justamente desse TCC, uma versão mais antiga de 12/10/2020, mas que já dá uma ideia de como funciona o HTML.

2.6 CSS

O QUE É CSS? Cascading Style Sheets (CSS) é um mecanismo simples para adicionar estilo (por exemplo, fontes, cores, espaçamento) a documentos da Web.6

Desta forma é possível deixar o visual da página mais agradável visualmente, tendo uma identidade visual própria, sem ele todas as páginas seriam preto e branco, sem cores... Exemplo de um código CSS:

Figura 6 – Imagem de um código CSS



2.7 LÓGICA DE PROGRAMAÇÃO

O computador não consegue interpretar diretamente um texto escrito em linguagem natural, devido as ambiguidades presente na nossa linguagem, então para que ele possa entender o que tem que ser feito é necessário escrever para ele o que tem que ser feito de forma muito didática, descrevendo com muitos detalhes.

A lógica de programação envolve operações lógicas que são realizadas em dados concretos, elas funcionam a partir de princípios lógicos e o resultado da execução dos algoritmos(sequência de passos[que ele entende o significado] informados ao computador para que ele possa cumprir um determinado objetivo) feitos através dela podem ser quantificados.

A distinção importante aqui é que a lógica de programação, e a lógica em geral, é fundamentalmente definida contra outros tipos de programação que não são construídos em lógica rígida ou estados e resultados quantificáveis.

Por exemplo, a lógica modal por sua natureza é comparada às operações quânticas teóricas que não fornecem um estado de conjunto específico ao qual os computadores podem aplicar a lógica.

A lógica de programação em geral repousa sobre uma base de lógica computacional que é compartilhada por humanos e máquinas, que é o que exploramos à medida que continuamos a interagir com novas tecnologias. Com isso em mente, pode-se desenvolver definições mais específicas de uma lógica de programação relacionada à base de um trecho de código.

2.8 ALGORITMOS

Todo programador para conversar com o computador e dizer a ele o que fazer usa o "código", mas para ele escrever e montar esse código é necessário conhecer o conceito de algoritmo.

Um algoritmo pode ser definido como uma sequência de raciocínios, instruções ou operações para alcançar um objetivo, sendo necessário que os passos sejam finitos e operados sistematicamente. Ele é fundamental para o ramo da computação, é através dele que os sistemas são criados, possibilitando ter acesso a dados que antes não eram possíveis só no papel e caneta(por serem quantidades colossais de dados a serem analisados a mão), os algoritmos são geralmente criados de forma a ser independente de uma linguagem, ou seja, um mesmo algoritmo pode ser implementado em mais de uma linguagem de programação... (MEDINA; FERTING, 2006)

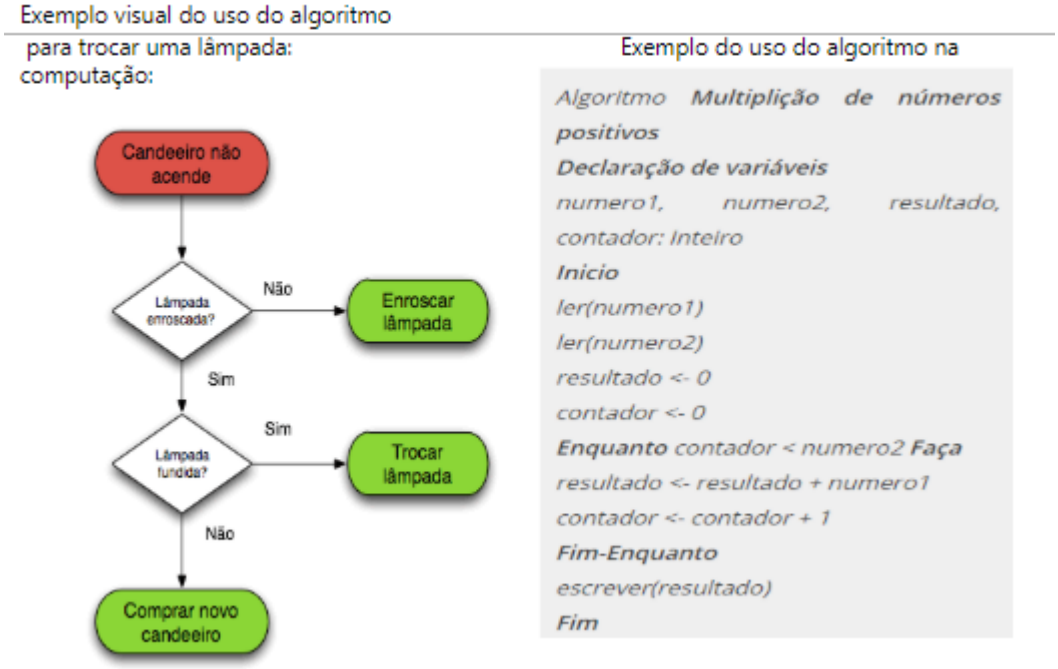
Porém eles são muito importantes fora da computação, alguns exemplos de algoritmos que podemos citar são: receitas culinárias, manual de instrução de aparelhos e funções matemáticas. (MEDINA; FERTING, 2006)

Uma receita culinária, por exemplo. Nela pode se encontrar os ingredientes necessários (dados de entrada), o passo a passo para realizar a receita (processamento ou instruções lógicas) e atinge um resultado (o prato finalizado). Um algoritmo, portanto, conta com a entrada (input) e saída (output) de informações mediadas pelas instruções, é fundamental compreender que o algoritmo se justifica no resultado que ele almeja alcançar, logo, deve ter um objetivo específico. Uma sequência de instruções simples pode se tornar mais complexa conforme a necessidade de considerar outras situações e se o algoritmo não for bem dimensionado em relação a quantidade e complexidade dos dados aos quais ele for exposto, ele pode travar ou não fornecer a resposta esperada, por conta disso, o algoritmo tende a crescer e ficar mais complexo com o passar do tempo para englobar todos os cenários possíveis. Ao montar a estrutura do algoritmo deve se ter em mente que para funcionar da forma esperada ele precisa seguir uma lógica sistemática. Por exemplo, se você está fazendo um bolo, mas "pula" a etapa de inserir farinha, no final, você não terá mais um bolo. Com o código, é a mesma coisa, sendo necessário ler linha por linha para que

ele atinja o objetivo final. As estruturas de um algoritmo são:

- variáveis: são as informações de entrada inseridas que determinam aonde o algoritmo poderá ir. As mais comuns são texto, inteiro, lógico e real;
- comandos de repetição: consiste no uso de "se" e "enquanto", para que o algoritmo saiba o que fazer quando determinados processos ocorrerem e o que fazer se eles mudarem. (MEDINA; FERTING, 2006)

Figura 7 – Exemplos de algoritmos



Disponível em: https://dicasdeprogramacao.com.br/o-que-e-algoritmo/

2.9 PHYTON

Essa será a linguagem utilizada no BackEnd, comumente usado para desenvolver algoritmos e aplicações de inteligência artificial.

2.10 INTELIGÊNCIA ARTIFICIAL

A Inteligência Artificial (IA) é um avanço tecnológico que permite que sistemas simulem uma inteligência similar à humana — indo além da programação de comandos específicos e bem definidos para tomar decisões de forma autônoma, baseadas em padrões de comportamento promissores obtidos através do refinamento de um algoritmo que varre enormes bancos de dados e métricas muito bem estipuladas do quão bom está esse comportamento.

Há algumas décadas, se estuda os denominados "agentes inteligentes", que percebem seu ambiente, entendem como podem operar e qual a melhor forma, essa foi a principal forma de inteligência computacional antes do novo milénio.

2.11 REDES NEURAIS E APRENDIZADO DE MÁQUINA

As redes neurais artificiais tem como objetivo emular o comportamento biológico do cérebro humano e nos processadores de um computador, o cérebro humano pode ser comparado a um computador altamente complexo, não linear e paralelo, por meio de seus neurônio ele consegue processar as informações (e realizar de forma muito mais rápida certas tarefas que um computador digital faz de forma satisfatória, como, reconhecimento de padrões, percepção, controle motor...), por tanto os algoritmos de Redes Neurais tem por objetivo emular "a aprendizagem", ou seja interagir com o ambiente ao qual ele foi exposto, colocar métricas de desempenho reforçando os bons comportamentos e punindo os ruins, alterando as conexões e pesos de sua rede neural com o intuito armazenar o comportamento adquirido e adquirir novos comportamentos que podem ser melhores... O procedimento usado para que a aplicação aprenda é chamado "algoritmo de aprendizagem", entre seus pontos fortes estão o fato que a estrutura interna de uma rede neural é altamente distribuída e descentralizada, capacidade de ser linear e não linear (ao passar pelos pesos eles se encontram em um neurônio ao qual esses valores passam por uma função matemática de ativação, que pode ser linear ou não), é possível fazer o mapeamento da entrada e saída, assim sendo é possível criar modelos estatísticos visando melhorar o comportamento da rede como um todo, além disso ela tem a capacidade de adaptar seus pesos para se comportar corretamente com as alterações do ambiente, se o ambiente for não estacionário(ou seja muda muito ao longo do tempo), a rede pode ser projetada para adequar seus pesos em tempo real com o objetivo de aumentar sua performance para novamente um nível satisfatório, ela pode também classificar o nível de crença na resposta dada, assim caso ela se encontrar com padrões ambíguos ela pode escolher qual entre eles é o mais próximo de sua crença de ser o padrão correto, pela sua forma de implementação com vários neurônios um afetando o outro a informação contextual também é armazenada, pelo fato de ter muitos neurônios ela também é tolerante a falhas de hardware, ela pode ser implementada de maneira hierárquica, a mesma notação matemática pode ser utilizado em todos os domínios de redes neurais, e também possuem uma analogia neurobiológica, se baseiam no sistema mais complexo, mais poderoso e tolerante a falhas que existe. (HAYKIN, 2007)

2.12 OPENAI, GPT-2 E GPT-3

OpenAI é uma instituição sem fins lucrativos de pesquisa em inteligência artificial, que tem como objetivo promover e desenvolver IA amigável, de tal forma a beneficiar a humanidade como um todo. O GPT-2 é uma IA de conversação em linguagem natural de código aberto, o GPT-3 é a evolução do GPT-2 por meio de uma API. (BROCKMANMIRA MURATIPETER, 2020)

2.13 TENSOR FLOW

TensorFlow é uma biblioteca de software de código aberto para cálculo numérico usando gráficos computacionais.Foi originalmente desenvolvido pela equipe do Google Brain da Organização de Pesquisa de Inteligência de Máquina do Google para aprendizado de máquina e pesquisa de rede neural profunda (Deep Learning), mas a biblioteca é suficientemente versátil e também pode ser aplicada a muitos outros campos. Foi fornecido como código aberto em 2015 e atingiu a versão 1.0 em fevereiro de 2017. Seu desenvolvimento e adoção são muito rápidos e há muitos colaboradores externos. O TensorFlow se tornou a biblioteca padrão para aprendizado profundo e desenvolvimento de aplicativos de inteligência artificial. (ACADEMY, 2018)

2.14 TENSOR FLOW

Placeholder é uma expressão do inglês: place (lugar) + hold (segurar/reservar). Em outras palavras, este é um local reservado para fins específicos. (HAHN, 2018)

3 REVISÃO BIBLIOGRÁFICA(TRABALHOS PARECIDOS)

4 DESENVOLVIMENTO

5 ANÁLISE

6 CONCLUSÕES

6.1 Apêndice - Código

Referências

ACADEMY, D. S. **O QUE É O TENSORFLOW MACHINE INTELLIGENCE PLATFORM?** 2018. Disponível em: <http://datascienceacademy.com.br/blog/o-que-e-o-tensorflow-machine-intelligence-platform/#:~:text=TensorFlow%20é%20uma%20biblioteca%20de,computaç~ao%20numérica%20usando%20grafos%20computacionais.&text=O%20TensorFlow%20vem%20se%20tornando,e%20aplicaç~oes%20de%20Inteligência%20Artificial.> Citado na página 14.

BITBUCKET, A. **Why git?** 2020. Disponível em: <https://www.atlassian.com/git/tutorials/why-git#:~:text=One%20of%20the%20biggest%20advantages,every%20change%20to%20your%20codebase.> Citado na página 4.

BROCKMANMIRA MURATIPETER, W. . O. G. **OpenAI API**. 2020. Disponível em: <https://openai.com/blog/openai-api/>. Citado na página 13.

CARR, L. et al. Why use hytime? **Electronic Publishing**, v. 7, n. 3, p. 163–178, 1994. Citado na página 9.

CELESTINO, A. L. **Entendendo a Separação de Conceitos (Separation of Concerns – SoC)**. 2020. Disponível em: <https://www.profissionaisti.com.br/entendendo-a-separacao-de-conceitos-separation-of-concerns-soc/>. Citado na página 7.

GITHOWTO. **Why git?** 2020. Disponível em: <https://githowto.com/pt-BR>. Citado na página 4.

HAHN, R. **8 Ótimos Sites de Placeholder na Web**. 2018. Disponível em: <https://king.host/blog/2018/07/8-otimos-sites-de-placeholder-na-web/>. Citado na página 14.

HAYKIN, S. **Redes neurais: princípios e prática**. [S.l.]: Bookman Editora, 2007. 27–31,75 p. Citado na página 13.

MEDINA, M.; FERTING, C. **Algoritmos e programação: teoria e prática**. [S.l.]: Novatec Editora, 2006. 13–22 p. Citado 2 vezes nas páginas 11 e 12.

MIKE, D. **What is computing?** 2020. Disponível em: <https://www.quora.com/What-is-computing>. Citado na página 3.

PÉREZ, J. E. **introduccion a JavaScript**. 2019. 5–9 p. Citado na página 8.

RAMOS, M. et al. Design de serviços e experiência do usuário (ux): uma análise do relacionamento das áreas. **DAPesquisa**, v. 11, n. 16, p. 105–123, 2016. Citado na página 6.

SILVA, J. **What is a Git Merge Conflict?** 2020. Disponível em: <https://blog.axosoft.com/learn-git-merge-conflict/#:~:text=A%20merge%20conflict%20is%20an,merge%20commits%20without%20your%20help>. Citado na página 5.

SILVA, M. S. **Construindo sites com CSS e (X) HTML: sites controlados por folhas de estilo em cascata**. [S.l.]: Novatec Editora, 2007. 49–56 p. Citado na página 8.

SILVA, M. S. **Construindo sites com CSS e (X) HTML: sites controlados por folhas de estilo em cascata**. [S.l.]: Novatec Editora, 2007. 37–38 p. Citado na página 8.

SOURCETREE. **Get started with Sourcetree**. 2020. Disponível em: <https://confluence.atlassian.com/get-started-with-sourcetree>. Citado na página 5.

SOUTO, M. **O que é front-end e back-end?** 2019. Disponível em: <https://www.alura.com.br/artigos/o-que-e-front-end-e-back-end>. Citado na página 8.

TORRES, V. M. Html e seus componentes. **Revista Ada Lovelace**, v. 2, p. 99–101, 2018. Citado na página 8.

Apêndices

APÊNDICE A – Nome do apêndice

Lembre-se que a diferença entre apêndice e anexo diz respeito à autoria do texto e/ou material ali colocado.

Caso o material ou texto suplementar ou complementar seja de sua autoria, então ele deverá ser colocado como um apêndice. Porém, caso a autoria seja de terceiros, então o material ou texto deverá ser colocado como anexo.

Caso seja conveniente, podem ser criados outros apêndices para o seu trabalho acadêmico. Basta recortar e colar este trecho neste mesmo documento. Lembre-se de alterar o "label" do apêndice.

Não é aconselhável colocar tudo que é complementar em um único apêndice. Organize os apêndices de modo que, em cada um deles, haja um único tipo de conteúdo. Isso facilita a leitura e compreensão para o leitor do trabalho.

APÊNDICE B – Nome do outro apêndice

conteúdo do novo apêndice

Anexos

ANEXO A – Nome do anexo

Lembre-se que a diferença entre apêndice e anexo diz respeito à autoria do texto e/ou material ali colocado.

Caso o material ou texto suplementar ou complementar seja de sua autoria, então ele deverá ser colocado como um apêndice. Porém, caso a autoria seja de terceiros, então o material ou texto deverá ser colocado como anexo.

Caso seja conveniente, podem ser criados outros anexos para o seu trabalho acadêmico. Basta recortar e colar este trecho neste mesmo documento. Lembre-se de alterar o "label" do anexo.

Organize seus anexos de modo a que, em cada um deles, haja um único tipo de conteúdo. Isso facilita a leitura e compreensão para o leitor do trabalho. É para ele que você escreve.

ANEXO B – Nome do outro anexo

conteúdo do outro anexo