João Pedro Toledo Cuyabano

Desenvolvedor Web (Full Stack), Programador Python e Java

Graduando Análise e Desenvolvimento de Sistemas na Fatec de São José dos Campos

E-mail: jp-toledo@hotmail.com

Telefone: +55 (12) 981131792

Endereço: Rua Jairo Hilário Moreira, 39 – Borda da Mata - Caçapava – SP – CEP: 12284-570

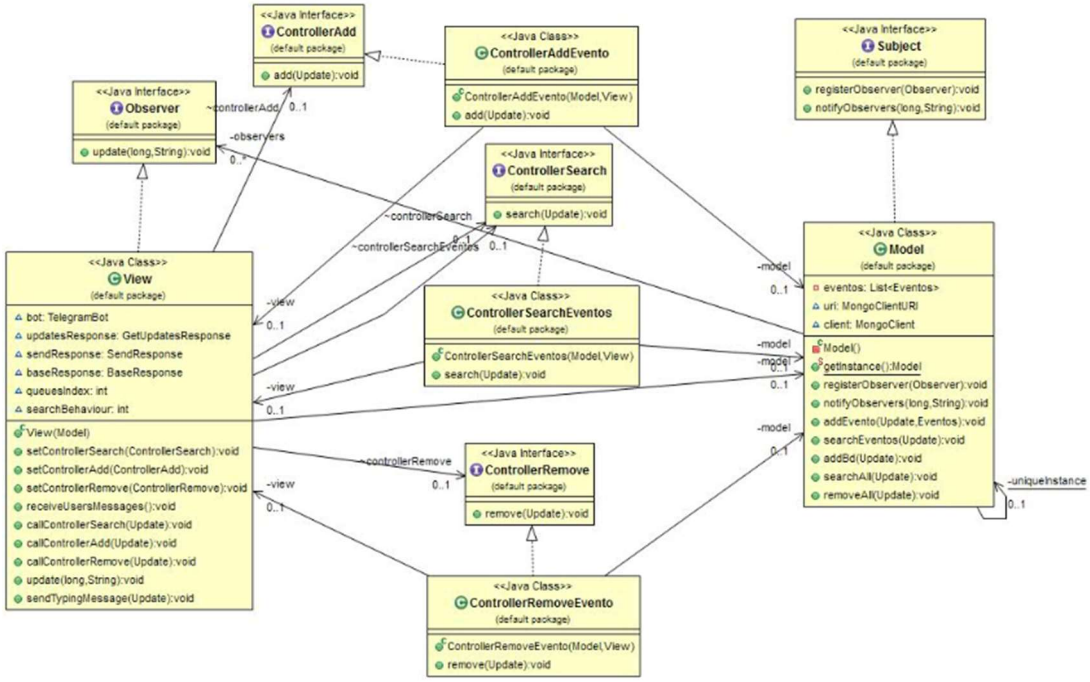
SUMÁRIO

ALGUNS PROJETOS REALIZADOS

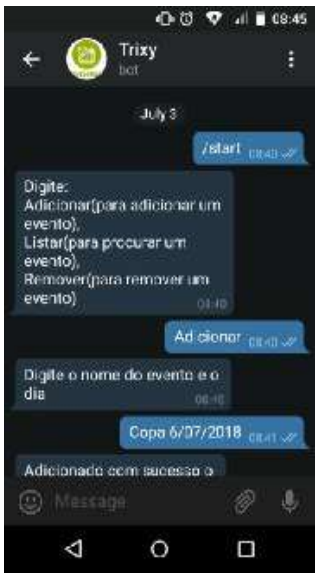
Chatbots no Telegram

- ChatBot para lembrete de eventos.

Utilizada a linguagem Java e o Padrão de Projeto MVC (Model View Controller)
Diagrama de Classe:



Exemplo de funcionamento:



- ChatBot para pesquisa de filmes e séries com o intuito de saber a sinopse e a opinião do público em rating.

Exemplo de código:

```
@app.route('/api/v1/filmes', methods=['GET'])
def filmes():
    html_doc = urllib.request.urlopen("http://www.adorocinema.com/filmes/numero-cinemas/").read()
    soup = BeautifulSoup(html_doc, "html.parser")

    data = []
    for dataBox in soup.find_all("div",class_="data_box"):
        nomeObj = dataBox.find("h2", class_="tt_18 d_inline").find("a", class_="no_underline")
        imgObj = dataBox.find(class_="img_side_content")
        sinopseObj = dataBox.find("div", class_="content").find("p")
        dataObj = dataBox.find("ul", class_="list_item_p2v tab_col_first").find("div", class_="oflow_a")

        data.append({ 'nome': nomeObj.text.strip(),
                  'poster' : imgObj.img['src'].strip(),
                  'sinopse' : sinopseObj.text.strip(),
                  'data' :  dataObj.text.strip()})

    # response = app.response_class(
    #     response=json.dumps(data),
    #     status=200,
    #     mimetype='application/json'
    # )
    # return response

    return jsonify({'filmes': data})
```

```
app = Flask(__name__)

@app.route('/api/v1/filmes', methods=['GET'])
def filmes():
    html_doc = urlopen("http://www.adorocinema.com/filmes/numero-cinemas/").read()
    soup = BeautifulSoup(html_doc, "html.parser")


    data = []
    for dataBox in soup.find_all("div", class_="card card-entity card-entity-list cf hred"):
        nomeObj = dataBox.find("h2", class_="meta-title")
        imgObj = dataBox.find(class_="thumbnail ")
        sinopseObj = dataBox.find("div", class_="synopsis")
        dataObj = dataBox.find(class_="meta-body").find(class_="meta-body-item meta-body-info")

        data.append({ 'nome': nomeObj.text.strip(),
                      'poster' : imgObj.img['data-src'].strip(),
                      'sinopse' : sinopseObj.text.strip(),
                      'data' :  dataObj.text[1:23].strip().replace('/',' ')})

    return jsonify({'filmes': data})
```

Desktop

- Jogo da Forca desenvolvido em Python, utilizando dicionário.

Exemplo de código – Função de lógica do jogo:

```
def condição(palavra):
    import random
    global letras
    global secreta
    secreta = random.choice(palavra)
    global corretas
    global erradas
    corretas = ''
    erradas = ''
    totalcorretas = ''
    letras = ''
    print('ESTE É O JOGO DA FORCA!')
    print('É bem fácil! Chute letras até acertar a palavra.\nAviso: Só poderá errar 6 vezes.')
    print('Lembre-se a palavra escolhida pode ter acento.\nCaso isso aconteça você deverá colocar a letra correta ACENTUADA.')
    if len(palavra) < 5:
        random.choice(palavra)
    else:
        desenha(secreta,corretas,erradas,totalcorretas,letras,palavra)
```

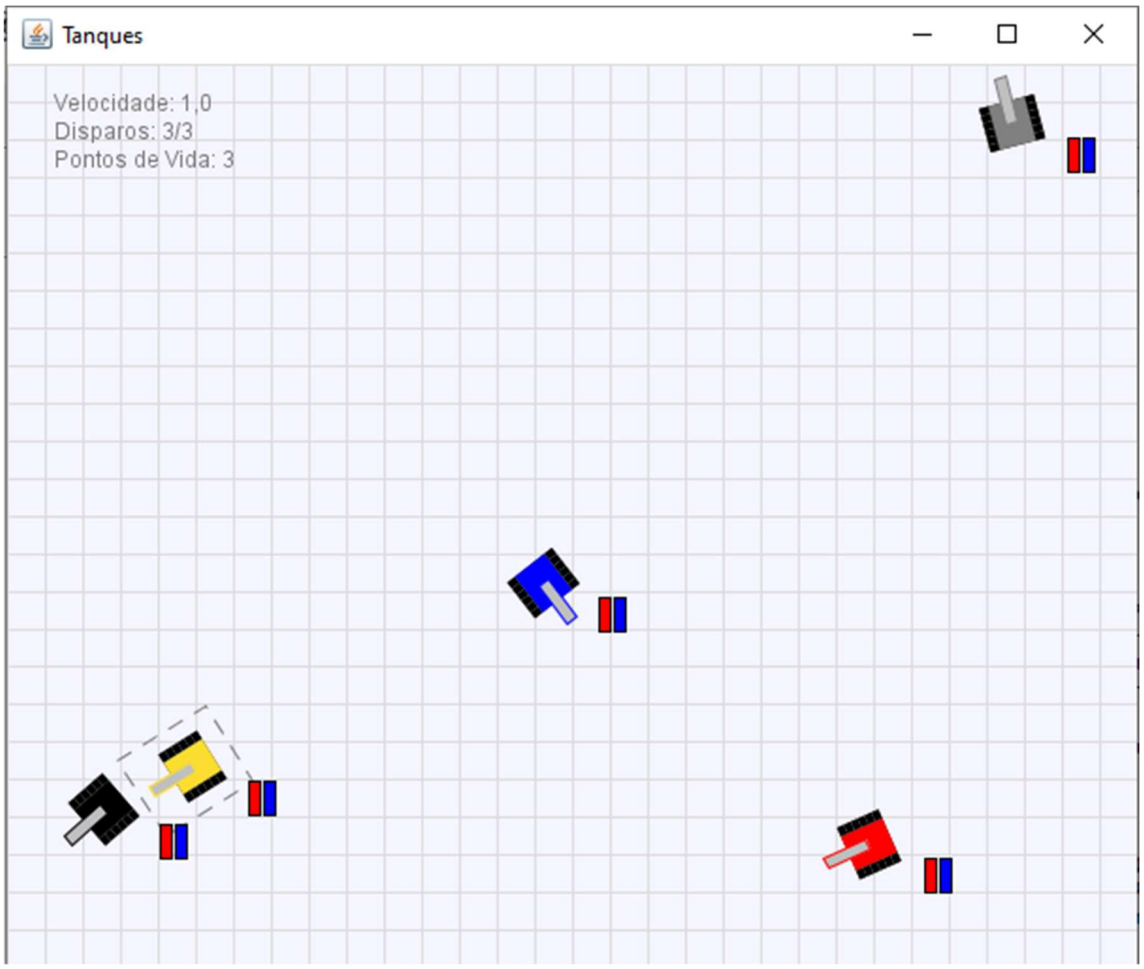
Função de chute:

```
def chute (secreta,corretas,erradas,totalcorretas,letras,palavra):
    global letra
    letra = str(input('\nDigite a letra: '))
    letra = letra.upper().lower()
    if letra in letras:
        print('Você já digitou essa letra. Digite outra.')
        return chute(secreta,corretas,erradas,totalcorretas,letras,palavra)
    else:
        if len(letra) != 1:
            print('Somente 1 letra')
            return chute(secreta,corretas,erradas,totalcorretas,letras,palavra)
        else:
            if letra.isalpha():
                coloca_letra (secreta,corretas,erradas,totalcorretas,letras,palavra)
            else:
                print('Só pode ser usada letras!')
                chute(secreta,corretas,erradas,totalcorretas,letras,palavra)
```

- Jogo de Tanque multiplayer local (LAN).

Utilizada linguagem Java, servidor e cliente feitos por Socket.

Jogo rodando no computador.



- Paint, aplicação para desenhar e apagar.

Utilizada linguagem Java para elaboração do jogo.

Classe Borracha:

```
public class Eraser implements Shape {
	private Color color;
	private int size;
	private List&lt;Point&gt; eraserList;

	protected Eraser(Point start, int size, Color color) {
		this.color = color;
		this.size = size;
		this.eraserList = new LinkedList&lt;Point&gt;();
		ajustPreferedSize(start);
		this.eraserList.add(start);
	}
	public void addPoint(Point p) {
		ajustPreferedSize(p);
		this.eraserList.add(p);
	}
	public Color getColor() {
		return color;
	}
	public int getSize() {
		return size;
	}
	public List&lt;Point&gt; getEraserList() {
		return eraserList;
	}
	private void ajustPreferedSize(Point p) {
		p.move((int) (p.getX() - Math.ceil(((double) size / 2.))), (int) (p.getY() - Math.ceil(((double) size / 2.))));
	}
}
```

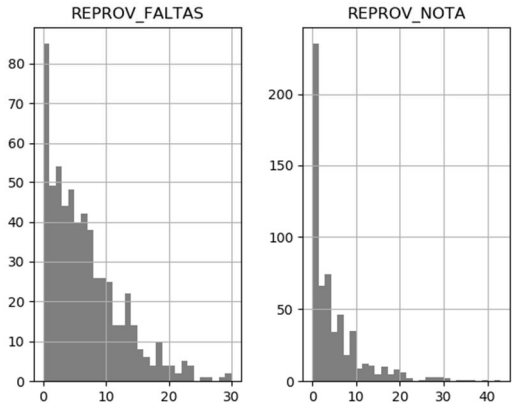
Analises de Dados

Trabalho Analise de dados Públicos com Python utilizando a biblioteca Pandas.

- Neste trabalho foi utilizado a tabela de levantamento histórico das turmas de ADS da Fatec de São José dos Campos, de 2014 a 2018.

```
In [7]: #Aqui temos um grafico, que mostra a comparação de alunos reporvados por faltas e por notas.
        %matplotlib notebook
        df[['REPROV_FALTAS','REPROV_NOTA']].hist(bins=30,alpha=0.5,color='Black')
```

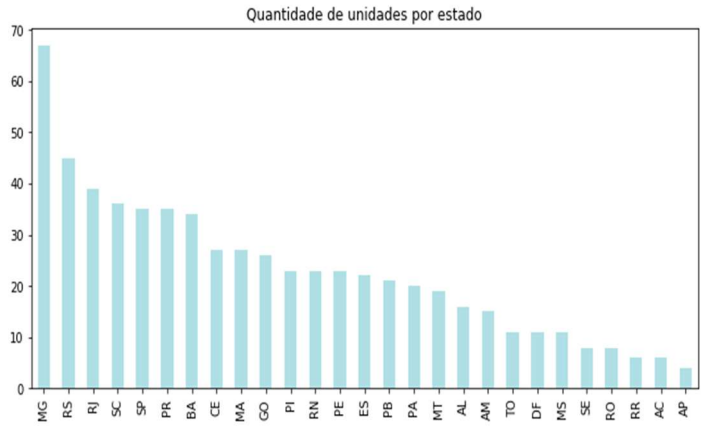
```
<IPython.core.display.Javascript object>
```



- Análise de Dados do Pronatec, utilizando Jupyter Notebook

```
In [9]: df['SIGLA_UF_UNIDADE'].value_counts().plot(kind ='bar', figsize = (11,5), grid = False, color = 'powderblue')
        plt.title('Quantidade de unidades por estado')
```

```
Out[9]: Text(0.5, 1.0, 'Quantidade de unidades por estado')
```



Inteligência Artificial

- Trabalho utilizando a linguagem de programação Python, juntamente com biblioteca **scikit-learn**, E **PyKnown.**

Sistema Especialista - sistemas que tem como objetivo simular o raciocínio de um profissional em alguma área de conhecimento específica. A arquitetura mais comum de um sistema especialista é a que envolve regras de produção na sua base de conhecimento.

Objetivo - Desenvolver um sistema especialista, a fim de guiar os jovens a se qual área científica ele mais se enquadra.

Gráfico de resposta do sistema especialista

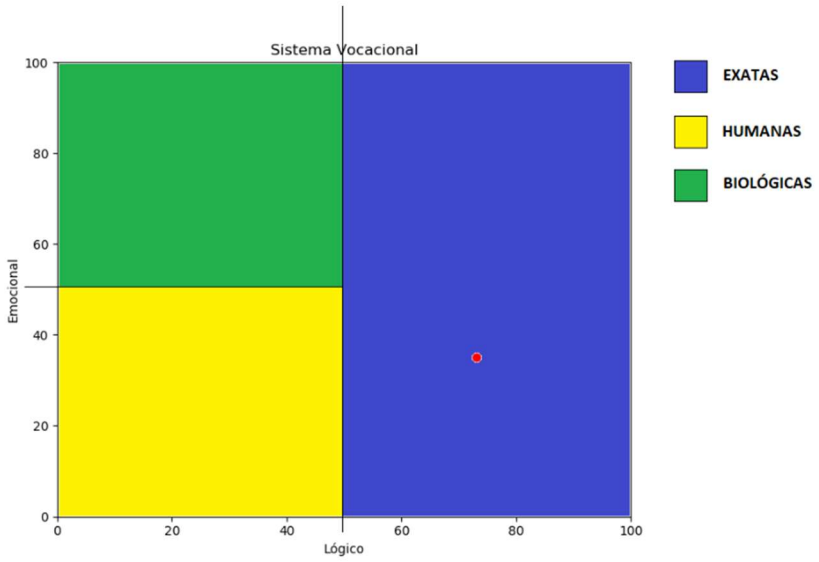


Tabela de Pesos

A seguir, a tabela de pesos mostra o peso de cada questão e como ela ira influenciar na reposta final do usuario:

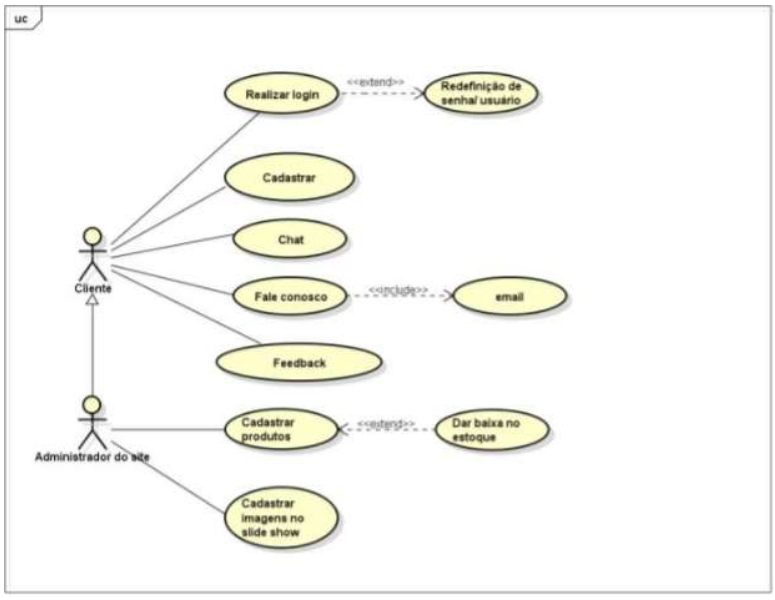
PESOS				
humanas	IF = TRUE		ELSE = FALSE	
	x (exatas)	y (emoção)	X (EXATAS)	Y(EMOÇÃO)
gosta de ler	2	1	6	2
criativo	3	2	7	3
escrever	3	3	8	4
questionador	3	3	7	3
exatas				
	x	y		
cálculos	9	1	1	1
prática	7	3	3	2
tecnologia	7	3	3	2
raciocínio lógico	10	0	0	2
Biológicas				
	x	y		
animais	3	7	3	3
medicina/veterinária	3	7	3	3
meio ambiente	2	8	3	3
preocupa pessoas	3	7	3	3

Desenvolvimento Web

- Site desenvolvido para E-commerce de bijuterias e quadros.

Produzido em HTML, CSS e PHP no front-end e Java no back-end, com Bando de Dados MySQL.

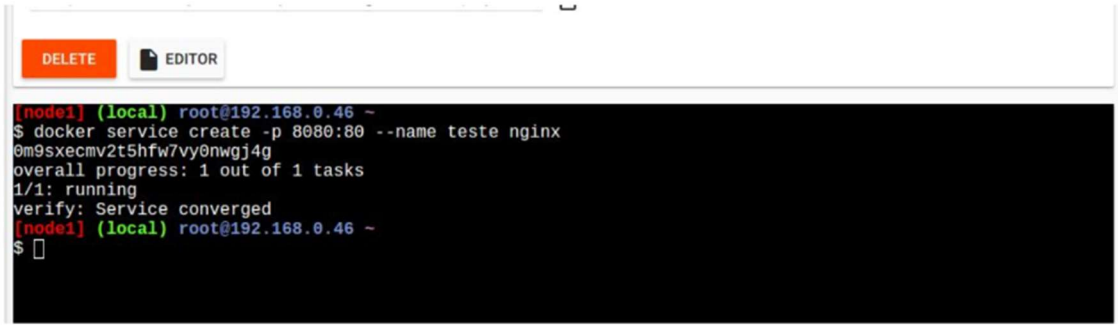
Diagrama de Casos de Uso:



Redes de Computadores

Criação de Cluster com Docker Swarm na plataforma **Play with Docker.**

Criação do Serviço



Serviço no browser, utilizando o endereço IP e porta do serviço.

Streaming

- Projeto com o objetivo de analisar e tratar dados inteligentes.

Utilização da linguagem Java, Amazon s3 para armazenagem e recuperação de dados, Amazon Kinesis para criação de streams, e DynamoDb para o Banco de Dados.

Criação de Stream:

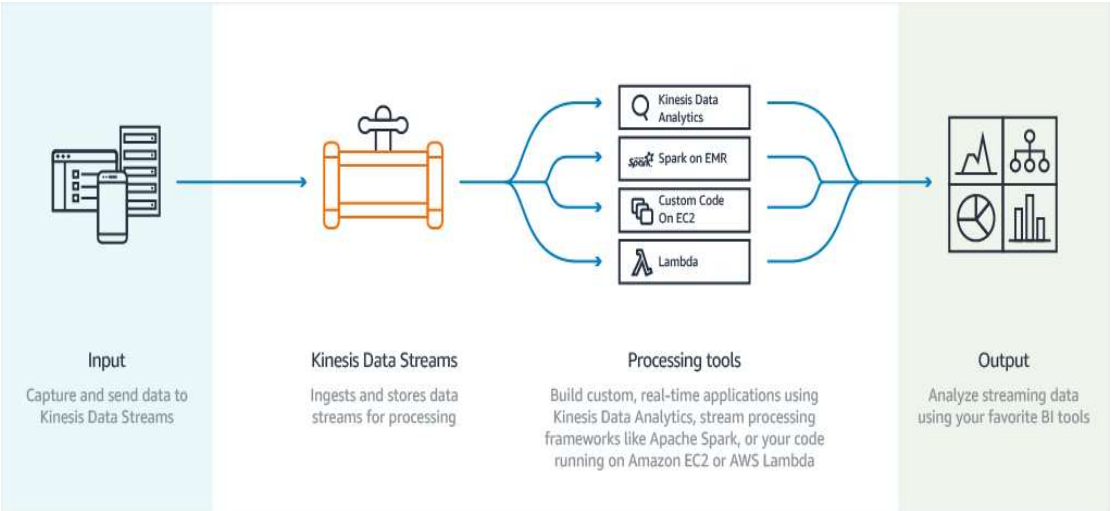
```
client.createStream( createStreamRequest );
DescribeStreamRequest describeStreamRequest = new DescribeStreamRequest();
describeStreamRequest.setStreamName(amazon-kinesis);

long startTime = System.currentTimeMillis();
long endTime = startTime + ( 10 * 60 * 1000 );
while ( System.currentTimeMillis() < endTime ) {
  try {
    Thread.sleep(20 * 1000);
  }
  catch ( Exception e ) {}

  try {
    DescribeStreamResult describeStreamResponse = client.describeStream( describeStreamRequest );
    String streamStatus = describeStreamResponse.getStreamDescription().getStreamStatus();
    if ( streamStatus.equals( "ATIVO" ) ) {
      break;
    }

    try {
      Thread.sleep( 1000 );
    }
    catch ( Exception e ) {}
  }
  catch ( ResourceNotFoundException e ) {}
}
if ( System.currentTimeMillis() >= endTime ) {
  throw new RuntimeException( "Stream " + amazon-kinesis + " não esta ativo" );
}
```

Arquitetura do funcionamento:



Certificação

- MongoDB

MongoDB University

Course Completion Confirmation

SEPTEMBER 2018

This confirms

JOÃO PEDRO TOLEDO CUYABANO

successfully completed

M101P: MongoDB for Developers

a course of study offered by MongoDB, Inc.

Shannon Bradshaw
VP, Education
MongoDB, Inc.