CURSO SUPERIOR TECNÓLOGO EM ANÁLISE DESENVOLVIMENTO DE SISTEMAS

ARMANDO RODRIGUES PROGENIO NETO

38511502

PORTIFÓLIO

RELATÓRIO DE AULA PRÁTICA

ARMANDO RODRIGUES PROGENIO NETO

38511502

PORTIFÓLIO:

RELATÓRIO DE AULA PRÁTICA

Trabalho textual apresentado como requisito parcial para obtenção de média semestral na disciplina de LINGUGEM DE PROGRAMAÇÃO do Curso Superior Tecnólogo em Análise e Desenvolvimento de Software.

Orientador: Prof. Anderson Emidio de Macedo Goncalves

SUMÁRIO

1. INTRODUÇÃO

Python é uma linguagem de programação muito conhecida pela sua curva de aprendizado extremamente suave, por se tratar de uma linguagem de programação orientada a objeto e ter uma sintaxe extremamente clara e fácil de aprender, se tornou uma das linguagens de programação mais conhecidas e usadas do mercado dos últimos tempos. Abrangendo desde ao aprendizado de máquina até a análise de dados e programação web, Python tem se tornado uma linguagem extremamente útil no mercado, com uma comunidade extremamente ativa, sendo assim constantemente atualizada em módulos e técnicas de usos para facilitar o desenvolvimento o usando-a. Há quem diga, por se tratar de uma linguagem interpretada e não compilada como a linguagem C, por exemplo, que se trata de uma linguagem um tanto quanto lenta, porém com técnicas de uso corretos e o uso de ferramentas que a linguagem oferece, pode se ter uma otimização quanto a velocidade, trazendo assim uma eficiência muito maior, e por consequência disso, trazendo benefícios em métodos de desenvolvimento como o ágil.

2. DESENVOLVIMENTO

Foi proposto a construção de um programa que retornasse o IMC de um determinado paciente, com as devidas informações fornecidas pela organização mundial da saúde, para se saber o estado atual do paciente para que se possa tomar as medidas cabíveis quanto ao bem-estar do paciente.

- **Cálculo IMC:**

  O cálculo foi feito com a criação de uma função dentro de uma classe em um arquivo separado para se ter uma melhor organização do código e ser um projeto muito mais fácil de ser dados manutenção

- **Funções:**

  Existem módulos como o módulo de Funções, que é um módulo para realizar alguns tratamentos de strings, assim verificando se o dado inserido pode ser convertido para float ou não, até mesmo opções de voltar ou sair do programa sem que erros interrompam o funcionamento correto do programa com aviso de que os dados inseridos foram incorretos e até mesmo com exemplos de como os dados possam ser inseridos para se ter uma melhor experiência do usuário.

- **Leitura de dados:**
  - A leitura de dados pedindo primeiramente o peso do paciente e fornecendo exemplos como, insira o peso o formato [45.65 ou 45,65] para que o usuário não se sinta confuso sobre o que fazer e até mesmo dando suporte como o de finalizar a sessão usando o comando [“Sair”] também se faz útil para se obter uma melhor experiência do usuário, assim entregando o melhor controle possível.
  - O mesmo citado acima se aplica a altura, com a adição de uma opção extra como a de inserir a opção de voltar com o comando [“Volta”] também se torna uma opção eficaz em caso de o usuário ter inserido dados errado e querer fazer a correção que precisar.
  - Após os dados inseridos e considerados incorretos, assim se vai para o cálculo usando a função certa para se obter o melhor resultado com uma melhor formatação e clareza possível, lembrando que uma limpeza automática também foi inserida para que o prompt de comando apartir de determinado momento identifique em que tipo de sistema operacional esteja inserido, assim realizando a limpeza necessária e retornando a melhor usabilidade possível ao usuário.

- Por fim se fez necessária a opção de continuar usando o programa para o maior conforto do usuário, com comando de resposta usan do apenas [“Sim” ou “Não”], essa verificação é útil, pois para usar novamente o programa o usuário não precisa reiniciar novamente o programa, assim se obtendo a melhor experiência possível.
- Para se saber mais sobre o projeto, uma documentação com instruções de uso e versão incluindo passo a passo para a instalação foi criado um repositório no GitHub que está disponível online para contribuições e visualização em https://github.com/oneto133/ProjetoIMC.git

3. MÉTODOS E TESTES

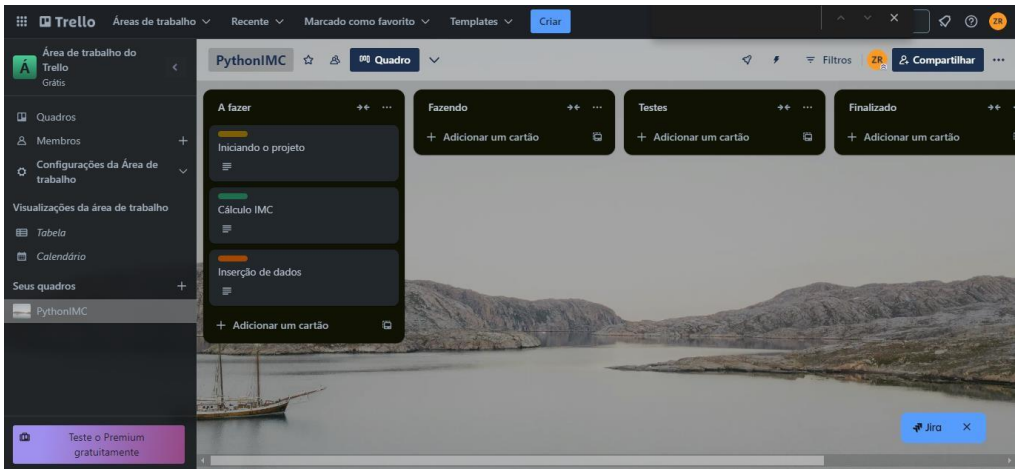
O uso de softwares como o Cloud Shell, Google colab, VS Code, Pycharm foi extremamente precisa para garantir o funcionamento e a manutenção em diferentes softwares, assim se tendo uma versatilidade do código, para que seja um projeto que possa usado por diferentes plataformas sem comprometer a integridade e eficácia do código

- Testes unitários foram criados usando a plataforma Cloud shell para se obter o melhor do código, assim garantindo a isenção de erros, o programa também foi apresentado a pessoas não técnicas e não inseridas no projeto para ter um feedback e desenvolver um produto focado no cliente.

- Foi feito o uso da metodologia ágil e do método incremental pois, conforme partes código estavam sendo concluídas, requisitos novos estavam surgindo para assim se aumentar o número de funcionalidades que o projeto iria ter, se tendo um programa mais simples e dinâmico.

- A documentação foi feita em docstrings dentro do próprio projeto com o objetivo de se ter um programa simples e fácil de manutenção, além de um arquivo Readme ter sido desenvolvido no GitHub para se ter diferentes maneiras de entender sobre o projeto, complementando com o controle de versão Git para se ter a possibilidade de retornar a versões anteriores em casos de erros.

4. RESULTADOS



Dando início ao projeto com o quadro Kanban.

```python
#Definindo a função para realizar o calculo IMC
class Imc():

  def __init__(self, peso, altura):
    self.peso = peso
    self.altura = altura


  def Indice_De_Massa(self):
    self.imc = self.peso / (self.altura * self.altura)
    print(f'{self.imc:.2f}')



peso = 60
altura = str('1,70')
alturare = altura.replace(',', '.')
Imc(peso, float(alturare)).Indice_De_Massa()
```

Iniciando o código do cálculo de massa.

```python
def Tratar_Peso_Altura(self, valor, max, tam, nome):
  valor = str(valor)
  pos = valor.find(".")
  if int(valor[:pos]) > max:
    return f"505 {nome} maior que o permitido"

  else:
    if len(valor[pos:])> tam:
      valor = valor.replace(".", "")
      valor = valor[:6]
      valor = float(valor[:pos] + "." + valor[pos:pos+3])
      return valor

    else:
      peso = float(valor)
      return valor
```

Código que tratará os dados obtidos pelo campo altura e peso.
O valor inicialmente recebido é convertido para um número flutuante antes do tratamento...

```python
# Criando a classe principal
class MeuAplicativo():
    def __init__(self):
        self.Hora_Atual = Funções.Funcao().hora()
        Mensagem_de_Alerta = Funções.Funcao().Alerta_Mensagem()
        self.horaatual = self.Hora_Atual
        self.horacalculo = self.Hora_Atual
        self.mensagem = Mensagem_de_Alerta
        self.continua = True
        return self.App()

    def App(self):
        return self.Peso()
```

Iniciando o programa.

```
Boa Noite! 20:40:20
Insira um peso apenas com números.
Exemplo:
   127,65 ou 65.78
Se precisar sair do programa, a qualquer momento digite SAIR.
Digite o peso: 65
Digite uma altura: 1.56
Peso: 65.0
Altura: 1.56
Carregando resultado...

O peso informado foi 65.00kg
A altura informada foi 1.56m
O Indice de massa corporal é: 26.71
O paciente está acima do peso!


=-==-==-==-==-==-==-==-==-==-==-==-=
Digite SIM para continuar ou NÃO para encerrar o programa...
Deseja continuar? [SIM/NÃO]sim
Voltando...
Boa Noite! 20:40:20
Insira um peso apenas com números.
Exemplo:
   127,65 ou 65.78
Se precisar sair do programa, a qualquer momento digite SAIR.
Digite o peso: 89
Digite uma altura: 1.75
Peso: 89.0
Altura: 1.75
Carregando resultado...

O peso informado foi 89.00kg
A altura informada foi 1.75m
O Indice de massa corporal é: 29.06
O paciente está acima do peso!


=-==-==-==-==-==-==-==-==-==-==-==-=
Digite SIM para continuar ou NÃO para encerrar o programa...
Deseja continuar? [SIM/NÃO]não
Até mais
Finalizado com êxito!
```

Executando.

5. CONCLUSÃO:

Utilizando de técnicas como a metodologia ágil, temos um ganho significativo de tempo com relação a entrega do produto final. O projeto obviamente não se resume apenas ao que foi entregue, pois se trata de apenas um protótipo desenvolvido em algumas horas, porém, com as técnicas adotadas como o uso de modularização que é muito eficaz quebrando o código em pedaços menores facilitando manutenção e também o uso de bibliotecas prontas como foi o caso da biblioteca Date para se obter a data atual e a biblioteca Time para se ter uma sen sação de suavidade ao usar o programa com pequenas animações de 2 segundos, temos um produto dinâmico e pronto para uso, mesmo que com funcionalidades limitadas.

O Cloud Shell se mostrou uma ferramenta extremamente eficaz, pois é simples de usar e aparenta ter o uso de uma inteligência artificial por trás auxiliando o desenvolvedor e trazendo também ajuda com as Docstrings sugerindo frases de acordo com o objetivo do código, sendo assim uma aliada na agilidade no processo de documentação.

Tenha acesso ao código completo no repositório público no GitHub https://github.com/oneto133/ProjetoIMC.git.

6. REFERÊNCIAS

A linguagem de programação **Python** possui uma sintaxe clara e concisa, facilitando a leitura e escrita de código *(Python Software Foundation, 2023). Disponível em* Welcome to Python.org. *Acesso em: 07 out. 2024.*

GOOGLE CLOUD. **Cloud Shell**. *Disponível em: https://cloud.google.com/shell/docs. Acesso em: 07 out. 2024.*

O **GitHub** foi utilizado como plataforma para compartilhar e versionar o código fontedeste projeto*. (GITHUB, s.d.). Disponível em:* GitHub