Atividade Avaliativa (GRUPO)

Disciplina: Pensamento Computacional **Data de Entrega:** 31/10/2025

Professor: Augusto Kruger Ortolan

Avaliação Bimestral

(4,0) Projeto Avaliativo

Descrição:
Compartilhar o projeto através do GitHub com o usuário **augustoortolan02@gmail.com** ou o usuário **augusto16ortolan.**

Objetivo:
Desenvolver novas soluções baseadas em linguagens de programação de alto nível, aplicando conhecimentos adquiridos em sala de aula em um projeto prático.

Menu interativo em Python

Desenvolver um programa em Python que funcione no terminal, apresentando um menu interativo para o usuário. O programa deve permitir inserir, visualizar, editar e excluir informações relacionadas a um tema escolhido pelo grupo. Os dados devem ser armazenados em um arquivo (por exemplo, .txt ou .csv) para que o conteúdo permaneça salvo entre execuções. A solução desenvolvida deve ser implementada com uma temática própria e única, seguindo a base criada durante as aulas.

Formação dos Grupos:

- Máximo de 3 integrantes por grupo;
- Cada grupo deverá escolher um tema diferente (não será permitido repetir);
- Caso decidam trocar de tema, deve-se primeiramente ter a aprovação do professor;

Parte 1: Configuração e Controle de Versão (Git)

1. [ ] Desenvolva o sistema utilizando a linguagem Python, com execução no terminal.
2. [ ] Todos os arquivos de dados (ex: .txt, .csv, .json) devem estar organizados em uma pasta chamada dados/.
3. [ ] O repositório do projeto deve conter pelo menos 5 commits, com mensagens descritivas sobre o progresso do sistema (ex: “adiciona menu principal”, “implementa função de cadastro”).
4. [ ] Todos os integrantes do grupo devem realizar pelo menos um commit no projeto.
5. [ ] Caso o aluno desenvolva o trabalho sozinho, deve pedir para um colega da turma realizar um commit no repositório, editando o arquivo README.md e adicionando:

   Testador do Sistema: Nome Completo – RA

6. [ ] Crie um arquivo README.md contendo:
   - Nome completo dos integrantes do grupo

    - RA (Registro Acadêmico) de cada integrante
    - Tema escolhido
    - Breve descrição do sistema
    - Tecnologias utilizadas (ex: Python, manipulação de arquivos, dicionários, funções etc.)
7. [ ] O projeto deve conter um arquivo principal chamado main.py, que executa o menu do sistema.

Parte 2: Funcionalidades do Sistema

1. [ ] O sistema deve apresentar um menu principal, com pelo menos 4 opções (ex: Cadastrar, Listar, Editar, Excluir, Sair).
2. [ ] O programa deve permanecer em execução até que o usuário escolha explicitamente Sair.
3. [ ] Utilize estruturas de condição (if, elif, else) para tratar as opções do menu.
4. [ ] Utilize laços de repetição (while, for) para percorrer dados e manter o menu ativo.
5. [ ] Crie e utilize pelo menos 4 funções (ex: cadastrar_item(), listar_itens(), editar_item(), excluir_item()).
6. [ ] Os dados devem ser armazenados e lidos de um arquivo na pasta dados/ (ex: dados/cadastros.txt).
7. [ ] Cada registro deve conter informações organizadas (pode usar dicionários ou tuplas).
8. [ ] Ao iniciar o programa, o sistema deve ler os dados do arquivo e carregá-los em memória.
9. [ ] Ao cadastrar, editar ou excluir, o sistema deve atualizar o arquivo automaticamente.
10. [ ] O sistema deve registrar logs de execução (ações do usuário, data e hora) em um arquivo chamado log.txt.
11. [ ] O log deve registrar, no mínimo:
    - Ação realizada (ex: "Cadastro de novo produto")
    - Data e hora (formato DD/MM/YYYY HH:MM:SS)
12. [ ] Organize o código em múltiplos arquivos:
    - main.py → responsável pelo menu e execução principal.
    - funcoes.py → conter as funções do sistema.
    - dados/ → pasta onde ficam os arquivos de dados e logs.
13. [ ] O sistema deve apresentar mensagens claras para o usuário (ex: "Cadastro realizado com sucesso!").
14. [ ] Utilize tratamento de erros com try e except para evitar que o programa quebre em casos como:
    - Entrada inválida do usuário (ex: digitar letras quando o programa espera um número);
    - Arquivo inexistente (deve ser criado automaticamente, se não encontrado);
    - Erros ao abrir, ler ou escrever arquivos.
15. [ ] Personalize o sistema com criatividade (ex: mensagens personalizadas, cores no terminal, contagem de registros, etc.).

Sugestões de temas:

- **Sistema de Cadastro de Alunos** – nome, curso e média final.

- **Gerenciador de Tarefas** – tarefa, prioridade e status (pendente/concluída).
- **Controle de Estoque** – produto, quantidade e preço.
- **Agenda de Contatos** – nome, telefone e e-mail.
- **Biblioteca Virtual** – título, autor e status (disponível/emprestado).
- **Controle Financeiro Pessoal** – tipo de operação (entrada/saída), valor e data.
- **Sistema de Votação Simples** – candidatos e total de votos.
- **Registro de Treinos** – exercício, duração e data.
- **Gerenciador de Filmes** – título, gênero e status (assistido/não assistido).
- **Simulador de Loja** – produtos, valores e carrinho de compras.

Fiquem à vontade para escolher algum desses temas e alterar como quiserem.

Critérios de avaliação:

Critério	Descrição	Pontos
Menu e Fluxo do Programa	Menu principal funcional, opções claras, entradas do usuário validadas, laços de repetição corretamente implementados e fluxo de execução sem travamentos.	1,0
Leitura e Escrita em Arquivos	Dados armazenados e recuperados corretamente, log de execução implementado, arquivos criados e atualizados automaticamente, persistência funcional.	1,0
Uso do Git e Documentação (README)	Todos os integrantes realizaram commits, mensagens descritivas, e README completo com nomes, RA, tema, descrição do sistema e tecnologias utilizadas.	0,8
Organização e Estrutura do Código	Código modularizado, funções bem definidas, arquivos separados (main.py, funcoes.py), nomes de variáveis significativos, comentários explicativos quando necessário.	0,7
Tratamento de Erros	Entradas do usuário e operações de arquivo protegidas contra erros, programa não quebra com entradas inválidas, mensagens de erro claras.	0,5