



Projetos

Semana 4 e 5 / Projeto 1

Aluno (a)	Projeto
	A
	B
	C
	D
	E
	F
	G
	H
	I
	J
	A
	B
	C
	D
	E
	F
	G
	H
	I
	J

Projeto A – Sistema de Gestão de Alunos e Projetos de IC

Desenvolver um sistema em linha de comando para gerenciar **alunos**, **projetos** e **participações** em projetos de iniciação científica ou desenvolvimento.

Funcionalidades obrigatórias

- Cadastrar, listar e buscar:
 - Alunos
 - Projetos
- Registrar e listar **participações** de alunos em projetos (com datas de início e fim).
- Impedir cadastros com campos vazios.
- Permitir buscar:
 - Todas as participações de um aluno.
 - Todos os alunos participantes de um projeto.

Classes sugeridas

- Aluno – prontuario, nome, email
- Projeto – codigo (int), titulo, responsavel, lista de participacoes
- Participacao – codigo, data_inicio, data_fim, aluno, projeto

Módulos sugeridos

- main.py – menus e fluxo principal.
- models.py – definição das classes.
- repositórios.py – leitura/escrita de alunos.csv, projetos.csv, participacoes.csv.
- menus.py – funções para menus de alunos, projetos e participações.

Arquivos de dados

- data/alunos.csv
- data/projetos.csv
- data/participacoes.csv

Projeto B – Biblioteca Escolar

Criar um sistema para gerenciar uma **biblioteca escolar**, permitindo controlar livros, usuários e empréstimos.

Funcionalidades obrigatórias

- Cadastrar / listar livros.
- Cadastrar / listar usuários.
- Realizar **empréstimos** (somente se o livro estiver disponível).
- Registrar **devolução**.
- Listar:
 - Empréstimos em aberto.
 - Histórico de empréstimos de um usuário.

Classes sugeridas

- Livro – código, título, autor, ano, disponível (True/False).
- Usuário – id, nome, email.
- Empréstimo – id, livro, usuário, data_emprestimo, data_devolucao, status.

Módulos sugeridos

- main.py – menus (livros, usuários, empréstimos).
- models.py – classes.
- repositório_livros.py, repositório_usuarios.py, repositório_emprestimos.py.

Arquivos de dados

- data/livros.csv
- data/usuarios.csv
- data/emprestimos.csv

Projeto C – Controle de Finanças Pessoais

Construir um sistema que registre **receitas** e **despesas**, calcule o saldo e gere pequenos relatórios.

Funcionalidades obrigatórias

- Registrar receita ou despesa (tipo, categoria, descrição, valor, data).
- Listar todas as transações.
- Mostrar **saldo atual**.
- Mostrar total por **categoria** e/ou por **mês**.

Classes sugeridas

- Transacao – id, tipo ('receita'/'despesa'), categoria, descricao, valor, data.

Módulos sugeridos

- main.py – menu principal.
- models.py – classe Transacao.
- repositorio_transacoes.py – leitura/escrita de transacoes.csv.
- relatorios.py – funções para resumo por categoria/mês.

Arquivos de dados

- data/transacoes.csv

Projeto D – Gerenciador de Tarefas e Hábitos

Desenvolver um gerenciador simples de **tarefas** e **hábitos**, com registro de conclusão e pequenos relatórios.

Funcionalidades obrigatórias

- Cadastrar, listar e marcar **tarefas** como concluídas.
- Cadastrar **hábitos** (ex.: “estudar 30 min por dia”) e registrar execuções.
- Listar:
 - Tarefas pendentes e concluídas.
 - Quantas vezes cada hábito foi cumprido.

Classes sugeridas

- Tarefa – id, titulo, descricao, data_limite, concluida.
- Habito – id, nome, frequencia (diário/semanal), contador_execucoes.

Módulos sugeridos

- main.py – menu (tarefas, hábitos, relatórios).
- models.py – classes.
- repositorio_tarefas.py, repositorio_habitos.py.
- relatorios.py – tarefas atrasadas, desempenho de hábitos.

Arquivos de dados

- data/tarefas.csv
- data/habitos.csv

Projeto E – Reserva de Salas e Laboratórios

Criar um sistema para gerenciar a **reserva de salas** ou laboratórios em uma escola/faculdade.

Funcionalidades obrigatórias

- Cadastrar salas (nome, capacidade, tipo).
- Registrar reservas informando sala, responsável, data e horários.
- Impedir reservas com **conflito de horário** para a mesma sala.
- Listar:
 - Reservas por sala em uma data.
 - Todas as reservas futuras.

Classes sugeridas

- Sala – id, nome, capacidade, tipo.
- Reserva – id, sala, responsavel, data, horario_inicio, horario_fim.

Módulos sugeridos

- main.py – menus (salas, reservas).
- models.py – classes.
- repositorio_salas.py, repositorio_reservas.py.
- validacoes.py – função para verificar conflito de horários.

Arquivos de dados

- data/salas.csv
- data/reservas.csv

Projeto F – Controle de Estoque da Lanchonete Escolar

Implementar um sistema para gerenciar o estoque e as vendas de uma lanchonete escolar.

Funcionalidades obrigatórias

- Cadastrar produtos (nome, preço, quantidade em estoque).
- Listar produtos e seus estoques.
- Registrar vendas (produto, quantidade, data).
- Atualizar estoque automaticamente na venda:
 - Não permitir estoque negativo.
- Relatórios:
 - Produtos com estoque baixo (abaixo de um limite).
 - Total vendido em um dia ou período.

Classes sugeridas

- Produto – código, nome, preço, quantidade_estoque.
- Venda – id, produto, quantidade, data.

Módulos sugeridos

- main.py – menus (produtos, vendas, relatórios).
- models.py – classes.
- repositório_produtos.py, repositório_vendas.py.
- relatórios.py – funções de resumo.

Arquivos de dados

- data/produtos.csv
- data/vendas.csv

Projeto G – Sistema de Cursos e Matrículas

Criar um sistema para gerenciar **cursos**, **alunos** e suas **matrículas** em cursos.

Funcionalidades obrigatórias

- Cadastrar e listar cursos.
- Cadastrar e listar alunos.
- Matricular aluno em curso.
- Listar:
 - Alunos matriculados em um curso.
 - Cursos em que um aluno está matriculado.

Classes sugeridas

- Curso – codigo, nome, carga_horaria.
- Aluno – id, nome, email.
- Matricula – id, aluno, curso, data_matricula, status.

Módulos sugeridos

- main.py – menus (cursos, alunos, matrículas).
- models.py – classes.
- repositorio_cursos.py, repositorio_alunos.py, repositorio_matriculas.py.

Arquivos de dados

- data/cursos.csv
- data/alunos.csv
- data/matriculas.csv

Projeto H – Sistema de Quiz de Múltipla Escolha

Desenvolver um sistema de **quiz/prova de múltipla escolha**, com cadastro de perguntas, montagem de quizzes e registro de resultados.

Funcionalidades obrigatórias

- Cadastrar perguntas com:
 - enunciado
 - 4 alternativas (A, B, C, D, por exemplo)
 - alternativa correta
- Montar quizzes escolhendo um conjunto de perguntas.
- Aplicar o quiz:
 - Mostrar perguntas ao usuário.
 - Receber respostas.
 - Calcular nota final.
- Salvar e listar **resultados** dos quizzes.

Classes sugeridas

- Pergunta – id, enunciado, alternativas, alternativa_correta.
- Quiz – id, titulo, lista_de_perguntas.
- Resultado – id, quiz, nome_participante, nota, data.

Módulos sugeridos

- main.py – menus (perguntas, quizzes, aplicar quiz, resultados).
- models.py – classes.
- repositorio_perguntas.py, repositorio_quizzes.py, repositorio_resultados.py.

Arquivos de dados

- data/perguntas.csv
- data/quizzes.csv
- data/resultados.csv

Projeto I – Sistema de Controle de Turmas e Frequência

Criar um sistema para gerenciar turmas e registrar a frequência dos alunos em cada aula.

Funcionalidades obrigatórias

- Cadastrar e listar turmas (ex.: código, nome da turma, professor).
- Cadastrar e listar alunos.
- Associar alunos a uma turma (matrícula na turma).
- Registrar frequência em uma data específica:
 - Presente / Ausente para cada aluno da turma.
- Listar:
 - Frequência de um aluno em uma turma.
 - Frequência geral da turma em uma data ou período.

Classes sugeridas

- Turma – código, nome, professor, lista_de_alunos.
- Aluno – id, nome, email.
- RegistroFrequencia – id, turma, aluno, data, presente (True/False).

Módulos sugeridos

- main.py – menus (turmas, alunos, frequência).
- models.py – classes.
- repositorio_turmas.py, repositorio_alunos.py, repositorio_frequencias.py.
- relatorios.py – funções para frequência por aluno/turma/período..

Arquivos de dados

- data/turmas.csv
- data/alunos.csv
- data/frequencias.csv

Projeto J – Sistema de Chamados de Suporte de TI

Desenvolver um sistema para registrar e acompanhar chamados de suporte de TI (problemas em computadores, rede, sistemas etc.).

Funcionalidades obrigatórias

- Cadastrar e listar usuários (quem abre chamado).
- Cadastrar e listar técnicos (quem atende chamados).
- Abrir chamado:
 - usuário, título, descrição, prioridade, data de abertura, status inicial ("aberto").
- Atribuir um técnico a um chamado.
- Atualizar status do chamado (aberto, em atendimento, resolvido, cancelado).
- Listar:
 - Chamados abertos.
 - Chamados de um determinado usuário.
 - Chamados atribuídos a um técnico.

Classes sugeridas

- Turma – código, nome, professor, lista_de_alunos.
- Usuario – id, nome, email, setor.
- Tecnico – id, nome, email, especialidade.
- Chamado – id, usuario, tecnico (pode começar como None), titulo, descricao, prioridade, data_abertura, data_fechamento, status.

Módulos sugeridos

- main.py – menus (usuários, técnicos, chamados).
- models.py – classes.
- repositório_usuarios.py, repositório_tecnicos.py, repositório_chamados.py.
- relatorios.py – listagens por status, por usuário, por técnico.

Arquivos de dados

- data/usuarios.csv
- data/tecnicos.csv
- data/chamados.csv