Filhos da PUC - Plataforma de Avaliação de Professores

INTEGRANTE	MATRÍCULA	PROFESSOR
FELIPE EDUARDO NUNES DA SILVA	2320615	FLAVIO HELENO BEVILACQUA E SILVA
LUCAS BRITO OLIVEIRA DA SILVA	2411907	
GABRIEL VALENTE FERREIRA PIRES	2310488	
THEO JEAN-MARIE J LEMAIRE	2520420	

Grupo 2

Problema

- Alunos não têm fácil acesso às avaliações dos professores
- Informações importantes ficam espalhadas entre grupos e redes sociais.
- Muitas avaliações são superficiais, inconsistentes ou pouco confiáveis.
- Falta anonimato seguro para quem deseja dar feedback sincero.
- Dificulta a escolha de matérias, professores e decisões acadêmicas.

Introdução

- Criamos um sistema em Python para organizar avaliações de professores de forma simples, padronizada e acessível.

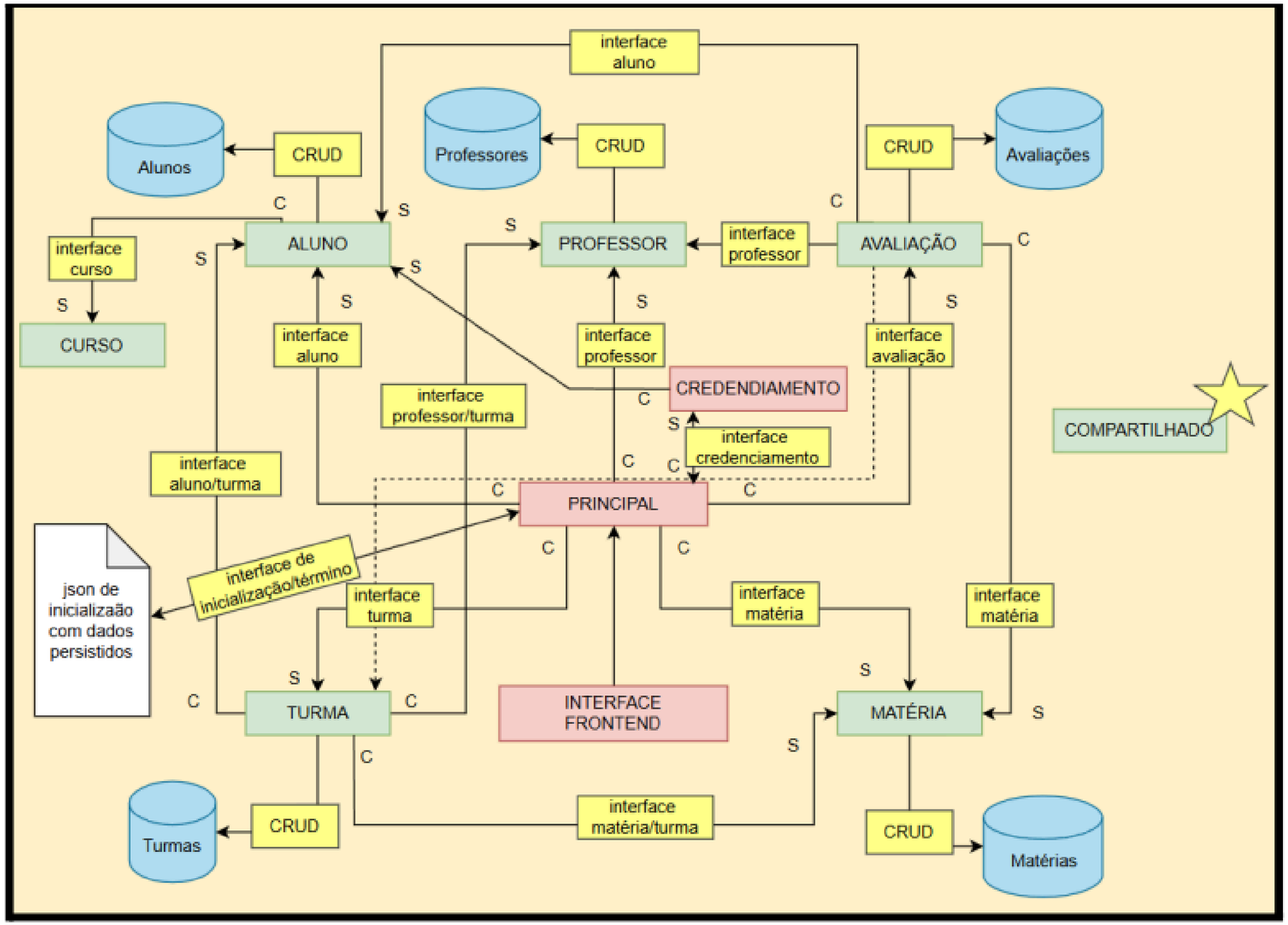
- A aplicação reúne professores, matérias e avaliações em um único ambiente, evitando informações soltas em redes sociais.

- A estrutura do projeto foi dividida em módulos independentes para facilitar manutenção, leitura e expansão futura.

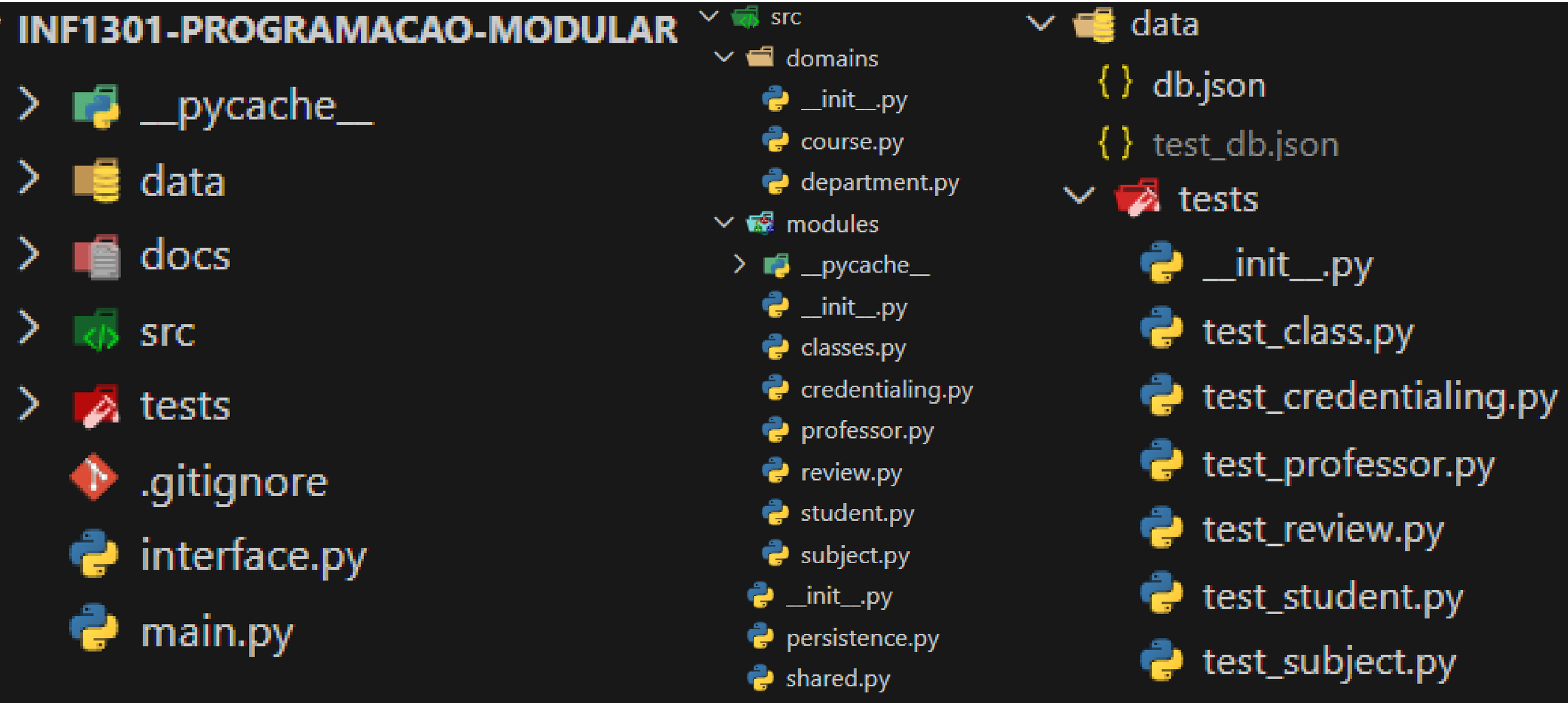
Solução / objetivo geral

- Plataforma que registra professores, matérias e avaliações detalhadas.
- Permite consultar listas organizadas de professores e feedbacks.
- Inclui testes automatizados para garantir confiabilidade das funções.
- Estruturação com princípios de programação modular

Arquitetura Geral



Estrutura do Projeto



Módulos de sistema

subject.py: Gerencia o catálogo de matérias (disciplinas), incluindo códigos, nomes e créditos
.
student.py: Mantém os dados pessoais dos alunos e gerencia seu histórico de matérias cursadas.

professor.py: Controla o cadastro de professores e suas respectivas alocações em departamentos.
classes.py: Gerencia a criação de turmas, vinculando uma matéria a professores e horários específicos.

review.py: Processa, valida e armazena as avaliações e comentários feitos pelos alunos.

***persistence.py*:** Responsável por salvar e carregar todos os dados do sistema em arquivo físico (JSON).

credentialing.py: Cuida da segurança, realizando o registro de novas contas e a autenticação (login) de usuários.

shared.py: Centraliza constantes globais e códigos de retorno padrão para evitar duplicidade.

interface.py: Exibe os menus interativos e gerencia a entrada e saída de dados para o usuário (Frontend).

main.py: O orquestrador que inicia o sistema, roda os testes de verificação e chama a interface.

Modulos de usuários

interface.py: Exibe os menus interativos e gerencia a entrada e saída de dados para o usuário (Frontend).

main.py: O orquestrador que inicia o sistema, roda os testes de verificação e chama a interface.

Especificações de funções

- objetivo
- descrição (requisitos especificados para esta função)
- acoplamento (parâmetros e retornos possíveis, cada um com uma breve explicação)
- condições de acoplamento
  - assertivas de entrada (regras válidas para os dados passados para a função antes de sua execução)
  - assertivas de saída (regras válidas após a execução da função)
- interface com o usuário (mensagens pro usuário e logs)

Especificações de funções

Função: *cria_aluno*

Objetivo: Criar uma nova conta de estudante, garantindo vínculo institucional e integridade dos dados.

Descrição: Gerencia a validação de campos obrigatórios (Matrícula, Nome, Usuário, Senha, E-mail, Curso) antes da persistência no banco de dados.

Asserções de Entrada: O Aluno deve existir. A Matéria deve existir O aluno não pode já ter cursado essa matéria (Checagem de duplicidade).

Asserções de Saída: Se SUCESSO, o código da matéria é adicionado à lista student['subjects'].

Especificações de funções

```
def retrieve_subject(code: int) -> Union[Dict, None]:
    """
    Objective: Retrieve a specific subject's details.
    Description: Public interface to search for a subject. Corresponds to busca_materia.
    Coupling:
        :param code (int): Subject's unique code.
        :return Union[Dict, None]: The subject object or None if not found/invalid.
    Coupling Conditions:
        Input Assertions: code must be a positive integer.
        Output Assertions: Returns the subject dictionary matching the code.
    User Interface: (Internal Log).
    """
```

Testes automatizados

- Testes unitários automatizados com unittest
- Metodologia TDD: cada requisito → caso de teste
- Cobertura completa das regras de negócio
- Inclui Caminho Feliz e Tratamento de Exceções
- Ambiente isolado: banco temporário + mocks
- Garante integridade da lógica sem afetar dados reais

Testes automatizados

```python
# --- Testes de Criação (create_student) ---

def test_01_create_student_t1_success(self):
    """T1: Retorna SUCESSO e insere o aluno no banco."""
    print("\nCaso de Teste 01 - Criação com Sucesso")
    ret_code = create_student(VALID_STUDENT_DATA)
    self.assertEqual(ret_code, RETURN_CODES['SUCCESS']) # Espera 1 (Sucesso)
    self.assertEqual(len(database['students']), 1)
    self.assertEqual(database['students'][0]['enrollment'], VALID_ENROLLMENT)
```

Funcionamento da Aplicação em Tempo Real...