Regras de Negócio & Critérios de Aceitação

Sistema de Gestão de Aulas

ISTEC

Minimum Viable Product (MVP)

Versão 1.0

Produzido pela equipa técnica: João Figueira, Rodrigo Rodrigues, Ricardo Pinto, Diogo Fernandes

Índice

1. Introdução e Contexto

Este documento define formalmente as Regras de Negócio (RN) e os Critérios de Aceitação do Sistema de Gestão de Aulas - ISTEC, desenvolvido em Python como projeto final da unidade curricular de Algoritmos e Estruturas de Dados.

As Regras de Negócio descrevem as **condições**, **restrições** e **comportamentos obrigatórios** que o sistema deve respeitar para funcionar corretamente.

Representam o contrato entre o sistema e os seus utilizadores.

Os Critérios de Aceitação definem, de forma objetiva e estável, em que condições o sistema é considerado correto para cada requisito.

Legenda de Prioridade: ALTA = requisito crítico, o sistema não funciona sem ele | MÉDIA = importante mas contornável | BAIXA = melhoria ou funcionalidade secundária

Legenda de Tipo de CA: DEVE PASSAR = teste positivo (fluxo normal) | DEVE FALHAR = teste negativo (validação de erro)

2. Regras de Negócio

2.1 Autenticação e Controlo de Acesso

O sistema distingue dois tipos de utilizadores com permissões totalmente separadas. O acesso a qualquer funcionalidade é condicionado pela autenticação prévia com credenciais válidas.

ID	Área	Descrição da Regra	Módulo Origem	Prior.
RN-01	Autenticação	O ID/Número do utilizador deve ser um valor numérico inteiro positivo. Qualquer entrada não numérica deve ser rejeitada com mensagem de erro.	main.py autenticar_usuario()	ALTA
RN-02	Autenticação	A autenticação do Aluno é feita pelo campo 'numero' (int) e 'password' (string). Ambos devem coincidir com os valores armazenados na base de dados.	main.py classAluno.py	ALTA
RN-03	Autenticação	A autenticação do Professor é feita pelo campo 'id' (int) e 'senha' (string). Ambos devem coincidir com os valores armazenados na base de dados.	main.py classProfessor.py	ALTA
RN-04	Autenticação	Credenciais inválidas não devem dar acesso ao sistema. Deve ser exibida a mensagem 'Credenciais Inválidas!' e o utilizador regressa ao menu principal.	main.py	ALTA
RN-05	Acesso	Um Aluno autenticado só tem acesso ao menu do Aluno (marcar presença). Não pode aceder a funcionalidades do Professor.	classAluno.py	ALTA
RN-06	Acesso	Um Professor autenticado só tem acesso ao menu do Professor. Não existe escalada de privilégios entre papéis.	classProfessor.py	ALTA

2.2 Gestão do Estado da Aula

O estado da aula é partilhado em memória através de um dicionário global. Todas as operações de abertura, fecho e registo de presenças dependem deste estado.

ID	Área	Descrição da Regra	Módulo Origem	Prior.
RN-07	Estado da Aula	Só pode existir uma aula aberta de cada vez. Tentar abrir uma aula já aberta deve exibir aviso e não reiniciar o cronómetro nem limpar as presenças.	classProfessor.py	ALTA
RN-08	Estado da Aula	Ao abrir uma aula, o cronómetro inicia imediatamente com time.time() e a lista de presenças é reiniciada para vazia.	classProfessor.py	ALTA
RN-09	Estado da Aula	Não é possível fechar uma aula que não esteja aberta. Tentar fazê-lo deve exibir aviso sem alterar nenhum estado.	classProfessor.py	ALTA
RN-10	Estado da Aula	A duração da aula é calculada em segundos como a diferença entre time.time() no fecho e o valor guardado na abertura.	classProfessor.py	MÉDIA

2.3 Marcação de Presença

A marcação de presença é a funcionalidade principal do Aluno. Está sujeita a dois pré-requisitos obrigatórios e a uma restrição de duplicação.

ID	Área	Descrição da Regra	Módulo Origem	Prior.
RN-11	Presença	Um Aluno só pode marcar presença se existir uma aula aberta (estado_aula['aberta'] == True). Caso contrário, deve ser exibida uma mensagem de erro.	classAluno.py	ALTA
RN-12	Presença	Cada Aluno só pode marcar presença uma vez por sessão de aula. A tentativa de marcar presença duplicada deve ser bloqueada com aviso.	classAluno.py	ALTA
RN-13	Presença	A presença é registada como um tuplo (numero, nome) e adicionada à lista estado_aula['presencas'].	classAluno.py	ALTA
RN-14	Presença	A verificação de duplicado é feita comparando o campo 'numero' do Aluno com o primeiro elemento de cada tuplo na lista de presenças.	classAluno.py	ALTA

2.4 Ordenação e Exportação de Relatórios

Ao fechar a aula, a lista de presenças deve ser ordenada antes de ser exibida ou exportada. O sistema oferece dois algoritmos de ordenação e dois formatos de exportação.

ID	Área	Descrição da Regra	Módulo Origem	Prior.
RN-15	Ordenação	A lista de presenças deve ser ordenada por ID (numero do aluno) em ordem crescente antes de ser exportada. É usado o Bubble Sort na funcionalidade de fecho de aula.	classProfessor.py algoritmos.py	ALTA
RN-16	Ordenação	O Bubble Sort deve operar sobre uma cópia da lista original, não modificando a lista em memória diretamente.	algoritmos.py	MÉDIA
RN-17	Ordenação	O Bubble Sort deve incluir otimização de early exit: se não ocorrer nenhuma troca numa passagem, o algoritmo termina antes de completar todas as iterações.	algoritmos.py	MÉDIA
RN-18	Exportação	A exportação de relatórios (CSV e PDF) só deve ocorrer após confirmação explícita do Professor com a resposta 's'.	classProfessor.py	MÉDIA
RN-19	Exportação CSV	O ficheiro CSV exportado deve conter cabeçalho com as colunas: ID Aluno, Nome, Hora da Aula, Duração (s).	classProfessor.py	MÉDIA
RN-20	Exportação PDF	A exportação PDF depende da biblioteca fpdf2. Se a biblioteca não estiver instalada, a funcionalidade é desativada com aviso, sem provocar crash do programa.	classProfessor.py	MÉDIA

2.5 Testes de Stress e Algoritmos

ID	Área	Descrição da Regra	Módulo Origem	Prior.
RN-21	Stress Test	A quantidade de registos para o teste de stress deve ser um inteiro positivo. Entrada não numérica ou valor ≤ 0 devem ser rejeitados com mensagem de erro.	classProfessor.py	MÉDIA
RN-22	Stress Test	Os IDs gerados para o teste devem ser únicos. Usa-se random.sample() para garantir que não existem duplicados, evitando que a BST ignore entradas silenciosamente.	testeStress.py	ALTA
RN-23	Stress Test	O número máximo de registos é limitado pelo universo de IDs disponíveis (1000 a 99999 = 98.999 valores). Se o valor pedido exceder este limite, é ajustado automaticamente.	testeStress.py	MÉDIA
RN-24	BST	A Árvore Binária de Busca (BST) não deve armazenar IDs duplicados. Uma tentativa de inserção com ID já existente é silenciosamente ignorada.	algoritmos.py	ALTA
RN-25	BST	O percurso in-order da BST deve produzir sempre a lista de presenças em ordem crescente de ID.	algoritmos.py	ALTA

2.6 Persistência de Dados

ID	Área	Descrição da Regra	Módulo Origem	Prior.
RN-26	Persistência	Os dados são carregados do ficheiro 'database.json' uma única vez no arranque do programa. Se o ficheiro não existir ou estiver corrompido, o programa exibe erro e termina.	leitorDados.py	ALTA
RN-27	Persistência	Ao sair do programa (opção 0 no menu principal), todos os dados atuais em memória são serializados e escritos no ficheiro 'database.json'.	main.py leitorDados.py	ALTA
RN-28	Persistência	Na leitura do JSON, registos com campos em falta ou de tipo inválido devem ser ignorados individualmente, com aviso, sem impedir o carregamento dos restantes.	leitorDados.py	MÉDIA
RN-29	Validação	Na criação de objetos Aluno, os campos nome (string não vazia), numero (int positivo) e idade (int positivo) são validados. Dados inválidos lançam ValueError.	classAluno.py	ALTA

3. Critérios de Aceitação

Cada critério de aceitação é objetivo, estável e rastreável a uma ou mais Regras de Negócio. O sistema é considerado correto quando todos os critérios marcados como **DEVE PASSAR**, produzem o resultado esperado, e todos os marcados como **DEVE FALHAR** são corretamente rejeitados.

3.1 Autenticação

ID	RN Ref.	Critério de Aceitação	Resultado Esperado	Tipo
CA-01	RN-01	Introduzir um ID não numérico (ex: 'abc') no login	O sistema exibe 'Erro: O ID/ Número deve ser numérico.' e regressa ao menu principal sem crash	**DEVE FALHAR**
CA-02	RN-02	Login de Aluno com numero=1001 e password='123' (existentes no JSON)	O sistema autentica o aluno e apresenta o menu do Aluno com o nome 'Rodrigo Rodrigues'	**DEVE PASSAR**
CA-03	RN-02	Login de Aluno com numero=1001 e password='errada'	O sistema exibe 'Credenciais Inválidas!' e regressa ao menu principal	**DEVE FALHAR**
CA-04	RN-03	Login de Professor com id=10 e senha='admin' (existentes no JSON)	O sistema autentica o professor e apresenta o painel do Professor com o nome 'Prof. Noronha'	**DEVE PASSAR**
CA-05	RN-04	Login com id e senha completamente errados para qualquer utilizador	O sistema exibe 'Credenciais Inválidas!' e regressa ao menu principal sem aceder a nenhum menu	**DEVE FALHAR**

3.2 Gestão da Aula

ID	RN Ref.	Critério de Aceitação	Resultado Esperado	Tipo
CA-06	RN-07	Professor tenta abrir uma aula que já está aberta	O sistema exibe aviso 'A aula já está aberta!' sem reiniciar o cronómetro nem limpar a lista de presenças	**DEVE FALHAR**
CA-07	RN-08	Professor abre a aula com sucesso	O estado estado_aula['aberta'] torna-se True, o cronómetro é iniciado e estado_aula['presencas'] é reiniciado para lista vazia	**DEVE PASSAR**

CA-08	RN-09	Professor tenta fechar uma aula que não está aberta	O sistema exibe aviso 'Não há nenhuma aula a decorrer para fechar.' sem alterar nenhum estado	**DEVE FALHAR**
CA-09	RN-10	Professor abre e fecha a aula após alguns segundos	A duração apresentada é um valor em segundos positivo e coerente com o tempo decorrido	**DEVE PASSAR**

3.3 Presença do Aluno

ID	RN Ref.	Critério de Aceitação	Resultado Esperado	Tipo
CA-10	RN-11	Aluno tenta marcar presença com a aula fechada	O sistema exibe 'Não existe nenhuma aula a decorrer no momento.' e não altera a lista de presenças	**DEVE FALHAR**
CA-11	RN-11,RN-13	Aluno marca presença com a aula aberta (primeira vez)	O sistema confirma 'Presença registada com sucesso!' e a lista estado_aula['presencas'] contém o tuplo (numero, nome) do aluno	**DEVE PASSAR**
CA-12	RN-12,RN-14	Aluno tenta marcar presença pela segunda vez na mesma aula	O sistema exibe 'Já tens a presença marcada nesta aula.' e a lista de presenças mantém apenas uma entrada do aluno	**DEVE FALHAR**
CA-13	RN-13	Dois alunos diferentes marcam presença na mesma aula aberta	A lista de presenças contém exatamente dois tuplos, um para cada aluno	**DEVE PASSAR**

3.4 Ordenação e Relatórios

ID	RN Ref.	Critério de Aceitação	Resultado Esperado	Tipo
CA-14	RN-15	Professor fecha aula com 3 presenças em ordem aleatória de IDs: 1003, 1001, 1002	A lista apresentada e exportada está ordenada como: 1001, 1002, 1003	**DEVE PASSAR**
CA-15	RN-16	Bubble Sort é chamado com uma lista de presenças	A função retorna uma nova lista ordenada e a lista original passada como argumento mantém-se inalterada	**DEVE PASSAR**
CA-16	RN-17	Bubble Sort é chamado com lista já ordenada	O algoritmo termina após a primeira passagem completa sem trocas (flag trocou=False ativa o early exit)	**DEVE PASSAR**
CA-17	RN-18	Professor recusa exportação introduzindo 'n' quando questionado	Nenhum ficheiro CSV nem PDF é gerado ou sobrescrito	**DEVE PASSAR**

CA-18	RN-19	Professor aceita exportação CSV com 2 presenças	Ficheiro 'relatorio_aula.csv' é criado com linha de cabeçalho e 2 linhas de dados com os campos corretos	**DEVE PASSAR**
CA-19	RN-20	Exportação PDF é solicitada sem a biblioteca fpdf2 instalada	O sistema exibe aviso de dependência em falta e não gera exceção não tratada nem crash	**DEVE FALHAR**

3.5 Testes de Stress

ID	RN Ref.	Critério de Aceitação	Resultado Esperado	Tipo
CA-20	RN-21	Professor introduz '0' ou '-5' como quantidade para o teste de stress	O sistema exibe 'O número de registos tem de ser positivo.' e não executa o teste	**DEVE FALHAR**
CA-21	RN-21	Professor introduz 'abc' como quantidade para o teste de stress	O sistema exibe 'Introduz um número inteiro válido.' e não executa o teste nem dá crash	**DEVE FALHAR**
CA-22	RN-22,RN-25	Executar Tree Sort (BST) com 1000 registos de IDs únicos	A lista devolvida por obter_lista_ordenada() contém exatamente 1000 elementos em ordem crescente de ID	**DEVE PASSAR**
CA-23	RN-24	Inserir dois registos com o mesmo ID na BST	A BST armazena apenas um registo com esse ID; o segundo é silenciosamente ignorado	**DEVE FALHAR**
CA-24	RN-23	Professor pede 200000 registos (acima do máximo de 98999)	O sistema ajusta automaticamente para 98999, avisa o utilizador e executa o teste com esse valor	**DEVE PASSAR**
CA-25	RN-22	Para N suficientemente grande (ex: N=5000), o Tree Sort deve superar o Bubble Sort em tempo	A saída mostra 'Tree Sort foi X.Xx mais rápido que Bubble Sort' com X > 1	**DEVE PASSAR**

3.6 Persistência de Dados

ID	RN Ref.	Critério de Aceitação	Resultado Esperado	Tipo
CA-26	RN-26	Arrancar o programa com 'database.json' inexistente	O sistema exibe mensagem de erro clara e termina de forma controlada, sem exceção não tratada	**DEVE FALHAR**
CA-27	RN-26	Arrancar o programa com 'database.json' com JSON mal formado	O sistema exibe 'Erro: O ficheiro está corrompido.' e termina de forma controlada	**DEVE FALHAR**
CA-28	RN-28	Ficheiro JSON contém um aluno com campo 'numero' a null	O registo inválido é ignorado com aviso, os restantes alunos válidos são carregados normalmente	**DEVE PASSAR**
CA-29	RN-27	Utilizador sai do programa com a opção 0	O ficheiro 'database.json' é atualizado com os dados atuais em memória	**DEVE PASSAR**
CA-30	RN-29	Tentar criar objeto Aluno com nome vazio (string '')	O construtor lança ValueError com mensagem descritiva e o objeto não é criado	**DEVE FALHAR**