🧠 Exercício de Fixação — Módulo 1 (Versão Intermediária)

🎯 Objetivos

- Reforçar os conceitos de análise de negócios, engenharia de requisitos e Scrum
- Aplicar lógica de programação em C# com foco em validação, repetição e decisão
- Estimular a argumentação com questões dissertativas
- Exigir maior atenção lógica e interpretação

> 💡 **Nota mínima para aprovação: 8,5 pontos (85%)**

📘 PARTE 1 — Exercícios Teóricos

1.1 — Marque a alternativa correta (0,5 ponto cada)

1. Qual das opções representa um requisito não funcional?

- [ ] A. O sistema deve gerar relatórios mensais
- [ ] B. O usuário poderá se cadastrar com e-mail ou telefone
- [x] C. O sistema deve carregar em até 2 segundos
- [ ] D. O administrador pode editar registros de usuários

2. O que é um backlog do produto no Scrum?

- [ ] A. Uma lista de erros pendentes
- [ ] B. Um documento com tarefas técnicas do desenvolvedor
- [ ] C. Um conjunto de ideias não aprovadas
- [x] D. Uma lista ordenada de tudo que é necessário no produto

3. Quando uma user story pode ser considerada “pronta para desenvolvimento”?

- [ ] A. Quando foi escrita pelo cliente
- [ ] B. Quando passou pela revisão técnica
- [x] C. Quando está bem definida, estimada e com critérios claros de aceitação
- [ ] D. Quando o Scrum Master autoriza

1.2 — Verdadeiro ou Falso (0,5 ponto cada)

4. Requisitos funcionais descrevem o comportamento esperado do sistema.

- [x] Verdadeiro
- [ ] Falso

5. O Product Owner tem autoridade para alterar o escopo do backlog a qualquer momento.

- [x] Verdadeiro
- [ ] Falso

6. O Analista de Negócios não participa da fase de levantamento de requisitos.

- [ ] Verdadeiro
- [x] Falso

1.3 — Questões Dissertativas (1 ponto cada)

7. Explique com suas palavras a importância do documento de visão em um projeto de software.

8. O Scrum prevê quatro reuniões principais: Planning, Daily Scrum, Review e Retrospective. Explique com suas palavras o propósito de cada uma dessas reuniões e como elas contribuem para a organização e o sucesso do projeto ágil.

🛠 PARTE 2 — Desafios Práticos em C#

🔹 Exercício 1 — Controle de Participantes com Nomes Únicos (1,0 ponto)

Descrição:
Permitir o cadastro de nomes de participantes para um evento.
O sistema deve impedir nomes duplicados e exibir a lista final ordenada alfabeticamente.

Requisitos:

- Ler a quantidade máxima de participantes
- Usar uma `List<string>` para armazenar os nomes
- Verificar se o nome já foi cadastrado
- Exibir a lista ordenada ao final

🔹 Exercício 2 — Controle de Lotes de Ingressos por Categoria (1,0 ponto)

📃 Descrição:
O sistema deve controlar a venda de dois tipos de ingressos: Padrão (R$ 30,00) e VIP (R$ 60,00). Cada tipo tem uma quantidade limitada. O sistema deve garantir que as vendas não ultrapassem o estoque e, ao final, exibir um resumo da venda.

🛠 Requisitos:

Defina um limite inicial de 100 ingressos padrão e 50 VIP;

Solicite ao usuário quantos ingressos de cada tipo deseja comprar;

Verifique se há quantidade disponível antes de confirmar a compra;

Calcule e exiba:

Quantidade total de ingressos vendidos

Valor total arrecadado

Quantidade restante de cada tipo

Exemplo esperado:

```
Ingressos padrão disponíveis: 100
Ingressos VIP disponíveis: 50

Quantos ingressos padrão deseja comprar? 70
Quantos ingressos VIP deseja comprar? 45

Venda confirmada!

Resumo:
Total vendidos: 115
Total arrecadado: R$ 4050,00
Ingressos padrão restantes: 30
Ingressos VIP restantes: 5
```

🔹 Exercício 3 — Verificação de Idade para Evento com Lista (1,0 ponto)

Descrição:
O sistema só aceita pessoas entre 18 e 60 anos.
Registrar nome e idade, até o limite de participantes informado.

Requisitos:

- Solicitar quantidade de vagas
- Para cada participante, solicitar nome e idade
- Validar:
  - Nome ≥ 5 caracteres
  - Idade entre 18 e 60
- Exibir mensagem de erro se inválido
- Ao final, mostrar todos os nomes aprovados

🔹 Exercício 4 — Menu de Evento com Datas (1,0 ponto)

Descrição:
Crie um sistema com menu que permite:

1. **Cadastrar Evento**
2. **Ver Detalhes**
3. **Sair**

Ao cadastrar um evento, solicite:

- Nome do evento
- Data (formato: dd/MM/yyyy)
- Valide se a data é futura
- Classifique como:
  - Evento próximo (até 30 dias)
  - Evento planejado (31 a 180 dias)
  - Evento distante (acima de 180 dias)

Requisitos Técnicos:

- Usar `DateTime.Parse()`
- Comparar com `DateTime.Now`
- Usar `switch` ou `if-else` para classificar o evento

🔹 Exercício 5 — Relatório de Presença em Evento (1,0 ponto)

Descrição:
Você deve criar um sistema para registrar a presença dos participantes em um evento. O organizador informará os nomes dos participantes previamente cadastrados e, em seguida, marcará quem esteve presente.

🛠 Requisitos:

Solicitar a quantidade total de participantes;

Receber os nomes dos participantes e armazenar em uma lista;

Perguntar individualmente se o participante esteve presente (S ou N);

Ao final, exibir:

Lista de presentes

Lista de ausentes

Percentual de presença

Exemplo esperado:

```
Quantos participantes? 3
Nome: Ana
Nome: Bruno
Nome: Carla

Ana esteve presente? (S/N): S
Bruno esteve presente? (S/N): N
Carla esteve presente? (S/N): S

Presentes:
- Ana
- Carla

Ausentes:
- Bruno

Percentual de presença: 66,67%
```

✅ Avaliação

Item	Pontuação
Teoria Objetiva (6 questões)	3,0 pts
Dissertativas (2 questões)	2,0 pts
Prática C# (5 exercícios)	5,0 pts
Nota Máxima	**10,0 pts**
Nota Mínima p/ Aprovação (85%)	**8,5 pts**
