Lousa — Fase 6: Repository CSV — persistência em arquivo

1. Visão geral da fase

- **Nome da fase:** Fase 6 — Repository CSV — persistência em arquivo
- **Meta central:** Evoluir o `Repository` da Fase 5 para **persistir em arquivo CSV**, mantendo o mesmo contrato (`IRepository<Book, int>`) e ampliando a disciplina de I/O.
- **Ideia-chave:** O domínio continua simples (ex.: `Book`), mas agora os dados sobrevivem em um **arquivo** `.csv` **em disco**, com cabeçalho, encoding e escape corretos.

---

2. Título e descrição (para ClassHero)

- **Título (ClassHero):** Fase 6 — Repository CSV (persistência em arquivo)
- **Descrição (ClassHero):**

> Evoluam o Repository para persistir em CSV (por exemplo, no domínio `Book`). Implementem `Add`, `GetById`, `ListAll`, `Update` e `Remove` gravando em arquivo CSV com cabeçalho. Definam claramente a política de Id (vem de fora ou é gerado) e o formato CSV (separador `,`, escape de aspas `"` → `""`, UTF-8). Incluam testes de integração usando arquivo temporário e cenários com vírgulas e aspas nos campos.

---

3. Entregáveis (no repositório único da equipe)

- **Pasta da fase:** `src/fase-05-repository-csv/`
- **Arquivo** `.md` **da fase:**
- Diagrama leve mostrando **Cliente → Repository (CSV) → Arquivo**.
- Snippets C# do contrato (reaproveitado), da implementação CSV e do uso via serviço cliente.
- **Testes de integração:**
- Cobrir arquivo ausente, arquivo vazio, inserção, atualização, remoção, id inexistente e campos com vírgulas/aspas.
- `README.md` **(raiz) atualizado:**
- Composição da equipe (nomes completos e RAs).
- Sumário com link/âncora para a Fase 6.
- Como executar os testes da Fase 6.

---

4. Contrato do Repository (relembre)

```
public interface IRepository<T, TId>
{
    T Add(T entity);
```

```
    T? GetById(TId id);
    IReadOnlyList<T> ListAll();
    bool Update(T entity);
    bool Remove(TId id);
}

public sealed record Book(int Id, string Title, string Author);
```

---

5. Implementação `CsvBookRepository` (persistência em CSV)

```
using System;
using System.Collections.Generic;
using System.IO;
using System.Linq;
using System.Text;

public sealed class CsvBookRepository : IRepository<Book, int>
{
    private readonly string _path;

    public CsvBookRepository(string path)
    {
        if (string.IsNullOrWhiteSpace(path))
            throw new ArgumentException("Path inválido", nameof(path));

        _path = path;
    }

    public Book Add(Book entity)
    {
        var list = Load();

        // política simples: se já existe Id, substitui; caso contrário,
adiciona
        list.RemoveAll(b => b.Id == entity.Id);
        list.Add(entity);

        Save(list);
        return entity;
    }

    public Book? GetById(int id)
    {
        return Load().FirstOrDefault(b => b.Id == id);
    }

    public IReadOnlyList<Book> ListAll()
    {
```

```
        return Load();
    }

    public bool Update(Book entity)
    {
        var list = Load();
        var index = list.FindIndex(b => b.Id == entity.Id);

        if (index < 0)
            return false;

        list[index] = entity;
        Save(list);
        return true;
    }

    public bool Remove(int id)
    {
        var list = Load();
        var removed = list.RemoveAll(b => b.Id == id) > 0;

        if (removed)
        {
            Save(list);
        }

        return removed;
    }

    // ------------ helpers privados ------------

    private List<Book> Load()
    {
        if (!File.Exists(_path))
            return new List<Book>();

        var lines = File.ReadAllLines(_path, Encoding.UTF8);
        if (lines.Length == 0)
            return new List<Book>();

        var list = new List<Book>();

        var startIndex = 0;
        if (lines[0].StartsWith("Id,"))
            startIndex = 1; // pula cabeçalho

        for (int i = startIndex; i < lines.Length; i++)
        {
            var line = lines[i];
            if (string.IsNullOrWhiteSpace(line))
                continue;
```

```
            var cols = SplitCsvLine(line);
            if (cols.Count < 3)
                continue; // ignora linha quebrada

            if (!int.TryParse(cols[0], out var id))
                continue;

            var title = cols[1];
            var author = cols[2];

            list.Add(new Book(id, title, author));
        }

        return list;
    }

    private void Save(List<Book> books)
    {
        var sb = new StringBuilder();

        // cabeçalho
        sb.AppendLine("Id,Title,Author");

        foreach (var book in books.OrderBy(b => b.Id))
        {
            var id = book.Id.ToString();
            var title = Escape(book.Title);
            var author = Escape(book.Author);


sb.Append(id).Append(',').Append(title).Append(',').Append(author).AppendLine();
        }

        File.WriteAllText(_path, sb.ToString(), Encoding.UTF8);
    }

    private static string Escape(string value)
    {
        if (value == null)
            return string.Empty;

        var needsQuotes = value.Contains(',') ||
                          value.Contains('"') ||
                          value.Contains('\n') ||
                          value.Contains('\r');

        var escaped = value.Replace("\"", "\"\"");
        return needsQuotes ? $"\"{escaped}\"" : escaped;
    }
```

```
private static List<string> SplitCsvLine(string line)
{
    var result = new List<string>();
    if (string.IsNullOrEmpty(line))
    {
        result.Add(string.Empty);
        return result;
    }

    var current = new StringBuilder();
    var inQuotes = false;

    for (int i = 0; i < line.Length; i++)
    {
        var c = line[i];

        if (inQuotes)
        {
            if (c == '"')
            {
                // possível aspas escapada
                if (i + 1 < line.Length && line[i + 1] == '"')
                {
                    current.Append('"');
                    i++; // consome a segunda aspas
                }
                else
                {
                    inQuotes = false;
                }
            }
            else
            {
                current.Append(c);
            }
        }
        else
        {
            if (c == ',')
            {
                result.Add(current.ToString());
                current.Clear();
            }
            else if (c == '"')
            {
                inQuotes = true;
            }
            else
            {
                current.Append(c);
            }
```

```
            }
        }

        result.Add(current.ToString());
        return result;
    }
}
```

> **Nota didática:** esta implementação é propositalmente simples e **não lida com concorrência** nem com esquemas evolutivos de versão de arquivo. Em projetos reais, considerar bibliotecas ou camadas adicionais para isso.

---

6. Exemplo de uso pelo cliente

```
public static class BookService
{
    public static Book Register(IRepository<Book, int> repo, Book book)
        => repo.Add(book);

    public static IReadOnlyList<Book> ListAll(IRepository<Book, int> repo)
        => repo.ListAll();
}

public static class Program
{
    public static void Main(string[] args)
    {
        var path = Path.Combine(AppContext.BaseDirectory, "books.csv");
        IRepository<Book, int> repo = new CsvBookRepository(path);

        BookService.Register(repo, new Book(1, "Código Limpo", "Robert C.
Martin"));
        BookService.Register(repo, new Book(2, "Domain-Driven Design", "Eric
Evans"));

        var all = BookService.ListAll(repo);

        Console.WriteLine("Livros cadastrados (CSV):");
        foreach (var book in all)
        {
            Console.WriteLine($"#{book.Id} - {book.Title} ({book.Author})");
        }
    }
}
```

---

7. Testes de integração (ideia rápida, xUnit)

```
using System;
using System.IO;
using System.Linq;
using Xunit;

public class CsvBookRepositoryTests
{
    private static string CreateTempPath()
    {
        var file = Path.GetTempFileName();
        // opcional: apagar o conteúdo inicial
        File.WriteAllText(file, string.Empty);
        return file;
    }

    private static CsvBookRepository CreateRepo(string path)
        => new CsvBookRepository(path);

    [Fact]
    public void ListAll_WhenFileDoesNotExist_ShouldReturnEmpty()
    {
        var path = Path.Combine(Path.GetTempPath(), Guid.NewGuid() + ".csv");
        var repo = CreateRepo(path);

        var all = repo.ListAll();

        Assert.Empty(all);
    }

    [Fact]
    public void Add_Then_ListAll_ShouldPersistInFile()
    {
        var path = CreateTempPath();
        var repo = CreateRepo(path);

        repo.Add(new Book(1, "Livro, com vírgula", "Autor ""com aspas"""));

        var all = repo.ListAll();

        Assert.Single(all);
        Assert.Equal(1, all[0].Id);
        Assert.Contains(",", all[0].Title);
        Assert.Contains("\"", all[0].Author);
    }

    [Fact]
    public void GetById_Existing_ShouldReturnBook()
    {
```

```
        var path = CreateTempPath();
        var repo = CreateRepo(path);

        repo.Add(new Book(1, "Livro A", "Autor"));

        var found = repo.GetById(1);

        Assert.NotNull(found);
        Assert.Equal("Livro A", found!.Title);
    }

    [Fact]
    public void GetById_Missing_ShouldReturnNull()
    {
        var path = CreateTempPath();
        var repo = CreateRepo(path);

        var found = repo.GetById(99);

        Assert.Null(found);
    }

    [Fact]
    public void Update_Existing_ShouldPersistChanges()
    {
        var path = CreateTempPath();
        var repo = CreateRepo(path);

        repo.Add(new Book(1, "Livro A", "Autor"));

        var updated = repo.Update(new Book(1, "Livro A (Revisto)", "Autor"));

        Assert.True(updated);
        Assert.Equal("Livro A (Revisto)", repo.GetById(1)!.Title);
    }

    [Fact]
    public void Remove_Existing_ShouldDeleteFromFile()
    {
        var path = CreateTempPath();
        var repo = CreateRepo(path);

        repo.Add(new Book(1, "Livro A", "Autor"));

        var removed = repo.Remove(1);

        Assert.True(removed);
        Assert.Empty(repo.ListAll());
    }
}
```

8. Critérios, tempo e peso sugeridos

• **Critérios (0–10):**
• Persistência CSV correta (inclui cabeçalho e escape de campos) — **0–3**
• Operações `Add`, `GetById`, `ListAll`, `Update`, `Remove` funcionando — **0–3**
• Testes de integração com arquivo temporário — **0–2**

• README e documentação clara do formato/política de Id — **0–2**

• **Tempo sugerido em sala: 25–30 minutos**

• 5–8 min: revisar contrato/Book e discutir formato CSV.
• 10–15 min: implementar `CsvBookRepository` (Load/Save/escape).

• 10–12 min: escrever testes com temp file e atualizar README.

• **Peso sugerido: 12/100** — consolida persistência simples em arquivo e disciplina de I/O, preparando a Fase 7 (JSON).