Desafio 2 MongoBD

Deal

Desafio 2 MongoBD

Proposta:

- ✓ **Iremos utilizar a LivrariaAPI como base de estudo.**
- ✓ **Nela iremos mudar sua leitura e gravação de uma base de dados relacional para uma não relacional.**
- ✓ **Iremos utilizar o MongoDB.**
- ✓ **Iremos usar um novo tipo de dados: Guid.**

Desafio 2 MongoBD

Passo a passo:

- ✓ **Iremos trocar o tipo de dados da propriedade Id em todos os arquivos que ele esteja presente. Exemplo: Atualmente usamos long Id, então iremos mudar para Guid Id.**
- ✓ **Devemos atualizar o DataContext da nossa API.**
- ✓ **Em seguida iremos substituir nosso repositório.**

Desafio 2 MongoBD

Atualizar nossa Entidade:

```
8 referências
public class Livro
{
    6 referências
    public Guid Id { get; private set; }
```

```
1 referência
public Livro(string nome, string autor, int edicao, string isbn, string imagem)
{
    Id = Guid.NewGuid();
    Nome = nome;
    Autor = autor;
    Edicao = edicao;
    Isbn = isbn;
    Imagem = imagem;
}
```

```
1 referência
public Livro(Guid id, string nome, string autor, int edicao, string isbn, string imagem)
{
    Id = id;
    Nome = nome;
    Autor = autor;
    Edicao = edicao;
    Isbn = isbn;
    Imagem = imagem;
}
```

Desafio 2 MongoBD

Atualizar nossos commands de entrada:

```
3 referências
public class AtualizarLivroCommand : Notifiable, ICommandPadrao
{
    [JsonIgnore]
    4 referências
    public Guid Id { get; set; }
```

```
2 referências
public class ApagarLivroCommand : Notifiable, ICommandPadrao
{
    [JsonIgnore]
    5 referências
    public Guid Id { get; set; }
```

```
6 referências
public bool ValidarCommand()
{
    try
    {
        if (Id == Guid.Empty)
            AddNotification("Id", "Id é um campo obrigatório");
```

Desafio 2 MongoBD

Atualizar nosso QueyResult:

```
19 referências
public class LivroQueryResult
{
    5 referências
    public Guid Id { get; set; }
    1 referência
    public string Nome { get; set; }
    1 referência
    public string Autor { get; set; }
    1 referência
    public int Edicao { get; set; }
    1 referência
    public string Isbn { get; set; }
    1 referência
    public string Imagem { get; set; }
}
```

Desafio 2 MongoBD

Atualizar nossa interface do repositório:

```
6 referências
public interface ILivroRepository
{
    2 referências
    void Inserir(Livro livro);
    2 referências
    void Alterar(Livro livro);
    2 referências
    void Deletar(Guid id);
    2 referências
    List<LivroQueryResult> Listar();
    2 referências
    LivroQueryResult ObterPorId(Guid id);
    3 referências
    bool CheckId(Guid id);
}
```

Desafio 2 MongoBD

Atualizar nosso Handler (Adicionar):

```
6 referências
public ICommandResult Handle(AdicionarLivroCommand command)
{
    try
    {
        if (!command.ValidarCommand())
            return new AdicionarLivroCommandResult(false, "Por favor, corrija as inconsistências abaixo", command.Notifications);

        string nome = command.Nome;
        string autor = command.Autor;
        int edicao = command.Edicao;
        string isbn = command.Isbn;
        string imagem = command.Imagem;

        Livro livro = new Livro(nome, autor, edicao, isbn, imagem);

        _repository.Inserir(livro);

        var retorno = new AdicionarLivroCommandResult(true, "Livro gravado com sucesso!", new
        {
            Id = livro.Id,
            Nome = livro.Nome,
            Autor = livro.Autor,
            Edicao = livro.Edicao,
            Isbn = livro.Isbn,
            Imagem = livro.Imagem
        });

        return retorno;
    }
    catch (Exception ex)
    {
        throw ex;
    }
}
```

Desafio 2 MongoBD

Atualizar nosso Handler (Atualizar):

```
6 referências
public ICommandResult Handle(AtualizarLivroCommand command)
{
    try
    {
        if (!command.ValidarCommand())
            return new AtualizarLivroCommandResult(false, "Por favor, corrija as inconsistências abaixo", command.Notifications);

        if (!_repository.CheckId(command.Id))
        {
            AddNotification("Id", "Id inválido. Este id não está cadastrado!");
            return new AtualizarLivroCommandResult(false, "Por favor, corrija as inconsistências abaixo", Notifications);
        }

        Guid id = command.Id;
        string nome = command.Nome;
        string autor = command.Autor;
        int edicao = command.Edicao;
        string isbn = command.Isbn;
        string imagem = command.Imagem;

        Livro livro = new Livro(id, nome, autor, edicao, isbn, imagem);

        _repository.Alterar(livro);

        var retorno = new AtualizarLivroCommandResult(true, "Livro alterado com sucesso!", new
        {
            Id = livro.Id,
            Nome = livro.Nome,
            Autor = livro.Autor,
            Edicao = livro.Edicao,
            Isbn = livro.Isbn,
            Imagem = livro.Imagem
        });

        return retorno;
    }
    catch (Exception ex)
    {
        throw ex;
```

Desafio 2 MongoBD

Atualizar nossas Controllers:

```
[HttpGet]
[Route("v1/livros/{id}")]
0 referências
public LivroQueryResult Livro(Guid id)...
```

```
[HttpDelete]
[Route("v1/livros/{id}")]
0 referências
public ICommandResult LivroExcluir(Guid id)...
```

```
[HttpPut]
[Route("v1/livros/{id}")]
0 referências
public ICommandResult LivroAlterar(Guid id, [FromBody] AtualizarLivroCommand command)...
```

Desafio 2 MongoBD

Atualizar nosso AppSettings:

```
"SettingsInfra": {
  "ConnectionString": "mongodb://localhost:27017/?readPreference=primary&appname=MongoDB%20Compass%20Community&ssl=false",
  "NomeBaseDados": "Livraria"
}
```

Desafio 2 MongoBD

Atualizar a classe SettingsInfra:

```
2 referências
public class SettingsInfra
{
    1 referência
    public string ConnectionString { get; set; }
    1 referência
    public string NomeBaseDados { get; set; }
}
```

Desafio 2 MongoBD

Atualizar nosso DataContext:

```
9 referências
public IMongoDatabase MongoDBConexao { get; set; }

0 referências
public DataContext(IOptions<SettingsInfra> options)
{
    try
    {
        IMongoClient client = new MongoClient(options.Value.ConnectionString);
        MongoDBConexao = client.GetDatabase(options.Value.NomeBaseDados);
    }
    catch (Exception ex)
    {
        throw ex;
    }
}
```

```
0 referências
public void Dispose()
{
    try
    {
        if (MongoDBConexao != null)
            MongoDBConexao = null;
    }
    catch (Exception ex)
    {
        throw ex;
    }
}
```

Desafio 2 MongoBD

Atualizar nosso Repositório:

```
private BsonDocument document = new BsonDocument();
private readonly DataContext _dataContext;

0 referências
public LivroRepository(DataContext dataContext)
{
    _dataContext = dataContext;
}
```

Desafio 2 MongoBD

Atualizar nosso Repositório:

```
2 referências
public void Inserir(Livro livro)
{
    try
    {
        IMongoCollection<BsonDocument> insert = _dataContext.MongoDBConexao.GetCollection<BsonDocument>("Livro");

        document.Add("_id", livro.Id);
        document.Add("Nome", livro.Nome);
        document.Add("Autor", livro.Autor);
        document.Add("Edicao", livro.Edicao);
        document.Add("Isbn", livro.Isbn);
        document.Add("Imagem", livro.Imagem);

        insert.InsertOne(document);
    }
    catch (Exception ex)
    {
        throw ex;
    }
}
```

Desafio 2 MongoBD

Atualizar nosso Repositório:

```
2 referências
public void Alterar(Livro livro)
{
    try
    {
        IMongoCollection<LivroQueryResult> update = _dataContext.MongoDBConexao.GetCollection<LivroQueryResult>("Livro");
        Expression<Func<LivroQueryResult, bool>> filter = x => x.Id.Equals(livro.Id);
        LivroQueryResult livroQR = update.Find(filter).FirstOrDefault();

        if (livroQR != null)
        {
            livroQR.Nome = livro.Nome;
            livroQR.Autor = livro.Autor;
            livroQR.Edicao = livro.Edicao;
            livroQR.Isbn = livro.Isbn;
            livroQR.Imagem = livro.Imagem;

            ReplaceOneResult result = update.ReplaceOne(filter, livroQR);
        }
    }
    catch (Exception ex)
    {
        throw ex;
    }
}
```

Desafio 2 MongoBD

Atualizar nosso Repositório:

```
2 referências
public void Deletar(Guid id)
{
    try
    {
        IMongoCollection<LivroQueryResult> delete = _dataContext.MongoDBConexao.GetCollection<LivroQueryResult>("Livro");
        Expression<Func<LivroQueryResult, bool>> filtro = x => x.Id.Equals(id);
        DeleteResult result = delete.DeleteOne(filtro);
    }
    catch (Exception ex)
    {
        throw ex;
    }
}
```

Desafio 2 MongoBD

Atualizar nosso Repositório:

```
2 referências
public List<LivroQueryResult> Listar()
{
    try
    {
        var collection = _dataContext.MongoDBConexao.GetCollection<LivroQueryResult>("Livro");

        var result = (from x in collection.AsQueryable()
                      orderby x.Id ascending
                      select x).ToList();

        return result;
    }
    catch (Exception ex)
    {
        throw ex;
    }
}
```

ace Livraria.Infra.Data.Repositori

Desafio 2 MongoBD

Atualizar nosso Repositório:

```
2 referências
public LivroQueryResult ObterPorId(Guid id)
{
    try
    {
        var collection = _dataContext.MongoDBConexao.GetCollection<LivroQueryResult>("Livro");

        var result = (from x in collection.AsQueryable()
                      where x.Id == id
                      select x).FirstOrDefault();

        return result;
    }
    catch (Exception ex)
    {
        throw ex;
    }
}
```

Desafio 2 MongoBD

Atualizar nosso Repositório:

```
3 referências
public bool CheckId(Guid id)
{
    try
    {
        IMongoCollection<LivroQueryResult> query = _dataContext.MongoDBConexao.GetCollection<LivroQueryResult>("Livro");
        Expression<Func<LivroQueryResult, bool>> filtro = x => x.Id.Equals(id);
        IList<LivroQueryResult> usuario = query.Find(filtro).ToList();

        return usuario.Count != 0;
    }
    catch (Exception ex)
    {
        throw ex;
    }
}
```


Dúvidas?