LINGUAGEM DE PROGRAMAÇÃO 4

NOME:_____ **TURMA:**_____

EXERCÍCIO 01 – SIGNINFORM.CS

O método abaixo é disparado sempre que o evento **Click** do botão **btnCadastrar** é lançado. Este método é usado para inserir um novo registro de usuário no banco de dados. Para isso, é necessário obter um email (que é usado como chave primária pelo banco), um username e uma senha.

```
private void btnCadastrar_Click(object sender, EventArgs e)
{
    // Série de validações...

    UsuarioRepository repo = new UsuarioRepository(DbUtil.ConnectionString);
    Usuario novoUsuario = new Usuario
    {
        Email = email,
        Username = username,
        Senha = senha
    };

    try
    {
        int linhas = repo.InserirUsuario(novoUsuario);
        if (linhas > 0)
        {
            lblMensagem.ForeColor = Color.Green;
            lblMensagem.Text = "Usuário cadastrado com sucesso!";
            LimparCampos();
        }
        else
        {
            lblMensagem.Text = "Erro ao cadastrar.";
        }
    }
    // Tratamento de exceções...
```

No entanto, sabe-se que essa abordagem irá inserir no banco a senha em claro do usuário gerando potencialmente uma falha de segurança grave. Altere o método apontado de modo a salvar no banco o hash da senha. A função abaixo pode ser usada para retornar o hash de qualquer string usando o algoritmo de criptografia SHA-256:

```
string GerarHashSha256(string texto)
```

EXERCÍCIO 02 – LOGINFORM.CS

a) Como a senha agora está sendo armazenada como hash no banco, isso significa que no momento de logar alguma providência também seja tomada.

A função abaixo é usada para logar um usuário no sistema. Que alteração deve ser realizada de modo a autenticar o usuário corretamente?

```
private void btnEntrar_Click(object sender, EventArgs e)
{
    string email = txtEmail.Text.Trim();
    string senha = txtSenha.Text;

    UsuarioRepository repo = new UsuarioRepository(DbUtil.ConnectionString);
    Usuario usuario = repo.ObterPorEmailESenha(email, senha);

    if (usuario != null)
    {
        MainForm main = new MainForm(usuario);
        this.Hide();
        main.ShowDialog();
        this.Show();
    }
    else
    {
        MessageBox.Show("Credenciais inválidas.");
    }
}
```

b) Se o usuário errar o email/senha três vezes seguidas, uma **MessageBox** deve sinalizar que ele foi bloqueado e então o **LoginForm** deve encerrar a aplicação. Para encerrar a aplicação, observe o código no método:

```
private void btnSair_Click(object sender, EventArgs e)
```

EXERCÍCIO 03 – MAINFORM.CS

a) Sempre que o **MainForm** é carregado, o evento **Load** é lançado. Para reagir a este evento, o método abaixo foi implementado:

```
private void MainForm_Load(object sender, EventArgs e)
{
    FilmeRepository repo = new FilmeRepository(DbUtil.ConnectionString);
    filmes = repo.ObterTodosFilmes();
    ExibirFilme();
}
```

Este método busca todos os filmes do banco (através da classe **FilmeRepository**) e preenche uma lista de filmes que será usada por este formulário assim que ele é carregado. Note que esta lista foi declarada no escopo de todo o formulário, de modo que pode ser acessível por qualquer método.

```
internal partial class MainForm : Form
{
    private Usuario usuarioLogado;
    private List<Filme> filmes; // Lista acessível em qualquer método deste formulário
    private int indiceAtual = 0;
    //...
}
```

Com base nessa informação, é pedido que você adapte esta solução de modo a preencher a lista de filmes com os filmes na ordem de seu lançamento (do mais antigo para o mais recente).

Dica: isso pode ser feito diretamente na consulta SQL usada na classe **FilmeRepository**.

b) Insira um botão novo neste formulário e o chame de **btnSortear**. Sempre que clicado, este botão deve sortear um novo filme da lista de filmes (que já está preenchida) e mostrá-lo para o usuário.

Dica: para sortear um número inteiro aleatório entre 0 e a quantidade de registros na lista filmes, use a classe Random e chame o método Next.

```
Random dado = new Random();
int indiceSorteado = dado.Next(0, filmes.Count);
```

c) Insira uma caixa de texto nova neste formulário e a chame de **txtFiltro**. Sempre que o valor desta caixa de texto mudar, a lista de filmes deve ser atualizada para carregar apenas os filmes cujo nome seja parecido com o texto informado.

Dica: para isso, utilize o operador LIKE na consulta SQL com coringas (%).