ETEC PROF. HORÁCIO AUGUSTO DA SILVEIRA

DESENVOLVIMENTO DE SISTEMAS – 2024/1

Luiz Fernando Souza / Quitéria Danno

Resolução da Atividade da Aula 11

(1) Método para alteração da senha de um usuário existente. (controller de usuários)

```
//Método para alteração de Senha.
[HttpPut("AlterarSenha")]
0 references
public async Task<IActionResult> AlterarSenhaUsuario(Usuario credenciais)
{
    try
    {
        Usuario? usuario = await _context.TB_USUARIOS //Busca o usuário no banco através do login
            .FirstOrDefaultAsync(x => x.Username.ToLower().Equals(credenciais.Username.ToLower()));

        if (usuario == null) //Se não achar nenhum usuário pelo login, retorna mensagem.
            throw new System.Exception("Usuário não encontrado.");

        Criptografia.CriarPasswordHash(credenciais.PasswordString, out byte[] hash, out byte[] salt);
        usuario.PasswordHash = hash; //Se o usuário existir, executa a criptografia
        usuario.PasswordSalt = salt; //guardando o hash e o salt nas propriedades do usuário

        _context.TB_USUARIOS.Update(usuario);
        int linhasAfetadas = await _context.SaveChangesAsync(); //Confirma a alteração no banco
        return Ok(linhasAfetadas); //Retorna as linhas afetadas (Geralmente sempre 1 linha msm)
    }
    catch (System.Exception ex)
    {
        return BadRequest(ex.Message);
    }
}
```

(2) Método para listar todos os usuários. (controller de usuários)

```
[HttpGet("GetAll")]
0 references
public async Task<IActionResult> GetUsuarios()
{
    try
    {
        List<Usuario> lista = await _context.TB_USUARIOS.ToListAsync();
        return Ok(lista);
    }
    catch (System.Exception ex)
    {
        return BadRequest(ex.Message);
    }
}
```

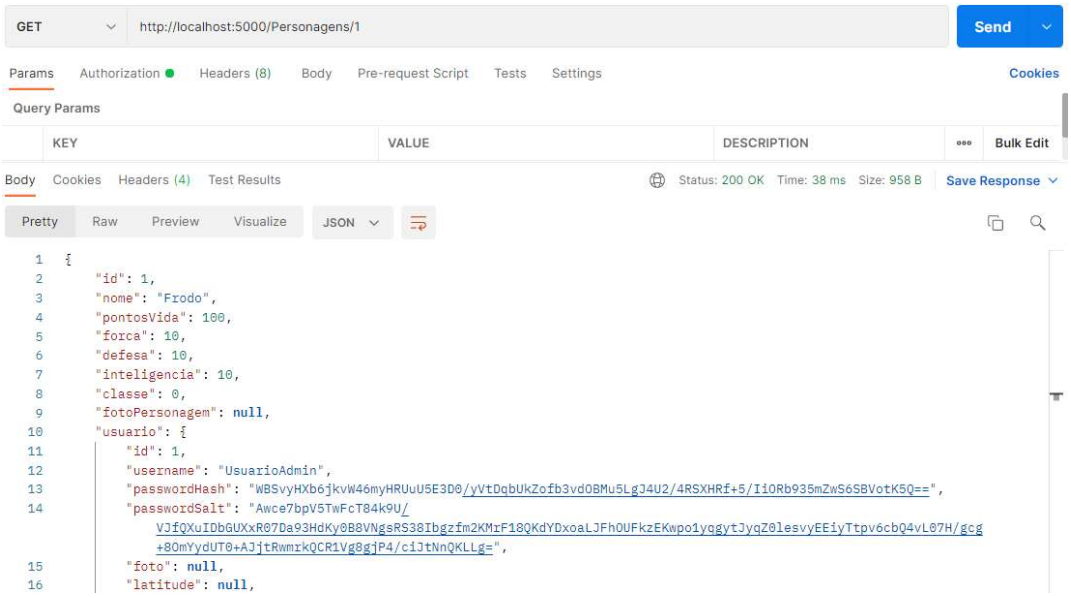
(3) Atualização da data de Acesso no método “Autenticar”. Esse método já existe, bastará inserir a operação sinalizada no print. O teste a ser feito é compilar, executar a API e realizar a chamada para a autenticação. Se der Ok, faça um select na tabela de Usuários para verificar se a data de acesso foi gravada.

```
[HttpPost("Autenticar")]
0 references
public async Task<IActionResult> AutenticarUsuario(Usuario credenciais)
{
    try
    {
        Usuario? usuario = await _context.TB_USUARIOS
            .FirstOrDefaultAsync(x => x.Username.ToLower().Equals(credenciais.Username.ToLower()));

        if (usuario == null)
        {
            throw new System.Exception("Usuário não encontrado.");
        }
        else if (!Criptografia.VerificarPasswordHash(credenciais.PasswordString, usuario.PasswordHash, usuario.PasswordSalt))
        {
            throw new System.Exception("Senha incorreta.");
        }
        else
        {
            usuario.DataAcesso = System.DateTime.Now;
            _context.TB_USUARIOS.Update(usuario);
            await _context.SaveChangesAsync(); //Confirma a alteração no banco

            return Ok(usuario);
        }
    }
    catch (System.Exception ex)
    {
        return BadRequest(ex.Message);
    }
}
```

(4) Programação da exibição do usuário que o personagem pertence no método GetSingle de PersonagensController:

```
[HttpGet("{id}")] //Buscar pelo id
0 references
public async Task<IActionResult> GetSingle(int id)
{
    try
    {
        Personagem p = await _context.TB_PERSONAGENS
            .Include(ar => ar.Arma) //Inclui na propriedade Arma do objeto p
            .Include(us => us.Usuario)  //Inclui na propriedade Usuario do objeto p
            .Include(ph => ph.PersonagemHabilidades)
                .ThenInclude(h => h.Habilidade) ////Inclui na lista de PersonagemHabilidade de p
            .FirstOrDefaultAsync(pBusca => pBusca.Id == id);

        return Ok(p);
    }
    catch (System.Exception ex)
    {
        return BadRequest(ex.Message);
    }
}
```

Teste através do postman



(5) Busca da listagem de PersonagemHabilidades de um determinado personagem passando o id do personagem por parâmetro. (controller PersonagemHabilidades)

```
[HttpGet("{personagemId}")]
        public async Task<IActionResult> GetHabilidadesPersonagem(int personagemId)
        {
            try
            {
                List<PersonagemHabilidade> phLista = new List<PersonagemHabilidade>();
                phLista = await _context.TB_PERSONAGENS_HABILIDADES
                .Include(p => p.Personagem)
                .Include(p => p.Habilidade)
                .Where(p => p.Personagem.Id == personagemId).ToListAsync();
                return Ok(phLista);
            }
            catch (System.Exception ex)
            {
                return BadRequest(ex.Message);
            }
        }
```

(6) Busca de todas as habilidades cadastradas na tabela de Habilidades através da controller PersonagemHabilidades. Não faremos uma controller para interagir com as habilidades, à medida que quisermos, faremos a adição diretamente nas tabelas do banco de dados.

```
[HttpGet("GetHabilidades")]
        public async Task<IActionResult> GetHabilidades()
        {
            try
            {
                List<Habilidade> habilidades = new List<Habilidade>();
                habilidades = await _context.TB_HABILIDADES.ToListAsync();
                return Ok(habilidades);
            }
            catch (System.Exception ex)
            {
                return BadRequest(ex.Message);
            }
        }
```

(7) Remoção da habilidade do personagem em **PersonagensController**. A busca é feita através do Id do personagem e do Id da Habilidade presente no objeto ph.

```
[HttpPost("DeletePersonagemHabilidade")]
        public async Task<IActionResult> DeleteAsync(PersonagemHabilidade ph)
        {
            try
            {
               PersonagemHabilidade? phRemover = await _context.TB_PERSONAGENS_HABILIDADES

                    .FirstOrDefaultAsync(phBusca => phBusca.PersonagemId == ph.PersonagemId
                     && phBusca.HabilidadeId == ph.HabilidadeId);
                if(phRemover == null)
                    throw new System.Exception("Personagem ou Habilidade não encontrados");

                _context.TB_PERSONAGENS_HABILIDADES.Remove(phRemover);
                int linhasAfetadas = await _context.SaveChangesAsync();
                return Ok(linhasAfetadas);
            }
            catch (System.Exception ex)
            {
                return BadRequest(ex.Message);
            }
        }
```