

**UNIVERSIDADE PAULISTA – UNIP**

Gabrielle de Lima Gabaldi **N523BJ1**

Kaique Marcelino Gomes **F1112H3**

Paola Regina Garbato **N528DG3**

Rafael Carneiro da Silva **N5065D6**

**PROJETO INTEGRADO MULTIDISCIPLINAR IV:**

Sistema de Controle Operacional para a Empresa BlockTech

**SÃO PAULO**

**2020**

Gabrielle de Lima Gabaldi **N523BJ1**

Kaique Marcelino Gomes **F1112H3**

Paola Regina Garbato **N528DG3**

Rafael Carneiro da Silva **N5065D6**

**PROJETO INTEGRADO MULTIDISCIPLINAR IV:**

Sistema de Controle Operacional Para a Empresa BlockTech

Projeto Integrado Multidisciplinar para a avaliação semestral do curso de Gestão Tecnológica em Análise e Desenvolvimento de Sistemas apresentado à Universidade Paulista - UNIP

Orientador: Prof. Emerson Abraham

**SÃO PAULO**

**2020**

Gabrielle de Lima Gabaldi **N523BJ1**

Kaique Marcelino Gomes **F1112H3**

Paola Regina Garbato **N528DG3**

Rafael Carneiro da Silva **N5065D6**

**PROJETO INTEGRADO MULTIDISCIPLINAR IV:**

Sistema de Controle Operacional Para a Empresa BlockTech

Projeto Integrado Multidisciplinar para a avaliação semestral no curso de Gestão Tecnológica em Análise e Desenvolvimento de Sistemas apresentado à Universidade Paulista - UNIP

Aprovado em:

BANCA EXAMINADORA

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_/\_\_\_\_/\_\_\_\_**Prof. Emerson Abraham  
Universidade Paulista - UNIP

**DEDICATÓRIA**

Dedicamos esse trabalho aos alunos e professores, que se mantiveram determinados, apesar de todos os obstáculos e as dificuldades desse semestre, devido a situação atual.

*"Encontra ânimo na dor e no desafio. Nesta vida só nos são colocados à frente os obstáculos que somos capazes de ultrapassar."*

*Augusto Branco*

**RESUMO**

A empresa Innovate and Renew Technology oferece serviços baseados na tecnologia blockchain, oferecendo serviços para o investimento em criptomoedas. Para uma melhor experiência de seus usuários, contratou a BlockTech para que fosse desenvolvido um sistema de gerenciamento desses investimentos. Tal sistema deve contemplar um sistema Desktop, e um sistema Web, sendo que, o Desktop deve ser acessível apenas pela empresa, e o Web, tanto pela empresa, quanto para os usuários. O sistema dará uma melhor visão de seus investimentos, tendo em vista que terão uma maior visibilidade através de gráficos semanais, poderão acompanhar as criptomoedas em gráficos diários, semanais e mensais, além de acompanhar seu lucro ou perda, diariamente.

Palavras-chave: Blockchain. Criptomoedas.

**ABSTRACT**

Keywords:

**LISTAS DE ILUSTRAÇÕES**

[Figura 1 - Diagrama de Casos de Uso - Visualizar Cadastro e Transações 19](#_Toc56778166)

[Figura 2 - Diagrama de Casos de Uso - Criptomoedas 20](#_Toc56778167)

[Figura 3 - Diagrama de Sequência 21](#_Toc56778168)

[Figura 4 - Diagrama de Classe 22](#_Toc56778169)

[Figura 5 - Notação Peter Chen 23](#_Toc56778170)

[Figura 6 - DER 24](#_Toc56778171)

[Figura 7 – DER - SQL Server Management Studio 25](#_Toc56778172)

[Figura 8 - LINQ 29](#_Toc56778173)

[Figura 9 - Estrutura do Projeto 31](#_Toc56778174)

[Figura 10 - Model - ContaCliente 32](#_Toc56778175)

[Figura 11 - ApplicationDbContext 33](#_Toc56778176)

[Figura 12 - Controlador - CriptomoedasHoje 34](#_Toc56778177)

[Figura 13 - Views e Controllers 35](#_Toc56778178)

[Figura 14 - ViewStart 36](#_Toc56778179)

[Figura 15 - Layout.cshtml 36](#_Toc56778180)

[Figura 16 - appsettings.json 37](#_Toc56778181)

[Figura 17 - Startup.cs 37](#_Toc56778182)

**LISTA DE TABELAS**

[Tabela 1 - Requisitos Não Funcionais 14](#_Toc56777940)

[Tabela 2 - Requisitos Funcionais e Regras de Negócio - Aplicativo Web 14](#_Toc56777941)

[Tabela 3 - Requisitos Funcionais e Regras de Negócio - Aplicativo Web 17](#_Toc56777942)

[Tabela 4 - Dicionário de Dados - Usuário 26](#_Toc56777943)

[Tabela 5 - Dicionário de Dados – Conta 26](#_Toc56777944)

[Tabela 6 - Dicionário de Dados - ContaCliente 26](#_Toc56777945)

[Tabela 7 - Dicionário de Dados - Criptomoeda 26](#_Toc56777946)

[Tabela 8 - Dicionário de Dados – CriptomoedaHoje 27](#_Toc56777947)

[Tabela 9 - Dicionário de Dados - CriptoSaldo 27](#_Toc56777948)

[Tabela 10 - Dicionário de Dados - Saldo 27](#_Toc56777949)

[Tabela 11 - Dicionário de Dados - Transacao 27](#_Toc56777950)

**LISTA DE ABREVIATURAS E SIGLAS**

**SUMÁRIO**

[1 INTRODUÇÃO 13](#_Toc56775513)

[2 ENGENHARIA DE SOFTWARE 14](#_Toc56775514)

[2.1 Requisitos Não Funcionais 14](#_Toc56775515)

[2.2 Requisitos Funcionais 14](#_Toc56775516)

[3 ANÁLISE DE SISTEMAS ORIENTADA A OBJETOS 18](#_Toc56775517)

[3.1 Diagrama de Casos de uso 18](#_Toc56775518)

[3.2 Diagrama de Sequência 20](#_Toc56775519)

[3.3 Diagrama de Classes 21](#_Toc56775520)

[4 BANCO DE DADOS 23](#_Toc56775521)

[4.1 Dicionário de Dados 26](#_Toc56775522)

[5 PROGRAMAÇÃO ORIENTADA A OBJETOS 27](#_Toc56775523)

[5.1 Web - ASP.NET 28](#_Toc56775524)

[5.1.1 Padrão MVC 28](#_Toc56775525)

[5.1.2 Entity Framework 28](#_Toc56775526)

[5.1.3 Estrutura do Projeto 30](#_Toc56775527)

[6 CONCLUSÃO 39](#_Toc56775528)

**REFERÊNCIAS......................................................................................**

**GLOSSÁRIO ................................................................................................**

**APÊNDICE A –**

**APÊNDICE B –**

**APÊNDICE C –**

**APÊNDICE D -**

# INTRODUÇÃO

# ENGENHARIA DE SOFTWARE

Na primeira parte do projeto, todos os requisitos foram levantados através de perguntas direcionadas e reuniões com os clientes. Com o decorrer do projeto, e das reuniões, algumas alterações foram necessárias para melhor atendê-los.

## Requisitos Não Funcionais

Os requisitos não funcionais abordam como as funcionalidades do sistema (requisitos funcionais), serão oferecidas ao usuário. Esses requisitos podem estar relacionados à qualidade, implementação, segurança, como pode ser visto na tabela 1.

Tabela 1 - Requisitos Não Funcionais

| **RNF** | **CÓD.** | **PRIORIDADE** | **GRAU DE DIFICULDADE** |
| --- | --- | --- | --- |
| O aplicativo web deve ser responsivo | 001 | Alta | Baixa - 2 |
| O sistema deve criptografar a senha dos usuários | 002 | Alta | Alta - 14 |
| O sistema deve ser entregue em 3 meses. | 003 | Alta | Alta - 15 |
| O aplicativo web deve utilizar a plataforma ASP.NET Core | 004 | Alta | Alta - 14 |
| O aplicativo desktop deve utilizar a plataforma Windows Forms | 005 | Alta | Média - 8 |
| Deve ser utilizado o banco de dados SQL Server Management Studio | 006 | Alta | Baixa - 6 |

Fonte: Paola Regina Garbato (2020)

## Requisitos Funcionais

Os requisitos funcionais tratam das funcionalidades do sistema. Podem ser respondidos com uma simples pergunta: “O que o sistema deve fazer?”.

A tabela 2 mostra os requisitos funcionais, e as regras de negócio do aplicativo web, e a tabela 3 do aplicativo desktop.

Tabela 2 - Requisitos Funcionais e Regras de Negócio - Aplicativo Web

(continua)

| **RF** | **CÓD.** | **PRIORIDADE** | **GRAU DE DIFICULDADE** | **REGRA DE NEGÓCIO** |
| --- | --- | --- | --- | --- |
| O sistema deve ser acessível pelos usuários e o Administrador. | 001 | Alta | Baixa - 2 | O administrador possui uma única contra pré-definida |
| O sistema deve permitir cadastro de clientes. | 002 | Alta | Baixa - 3 |  |
| O cadastro do cliente deverá conter: nome ou razão social, e-mail, CEP, UF, cidade, rua, número, telefone, nome de usuário e senha. | 003 | Alta | Baixa - 5 | O cliente só poderá se cadastrar ao inserir todos seus dados |
| O sistema deve permitir a realização do login com a inserção do nome de usuário e senha definidos no cadastro. | 004 | Alta | Baixa - 3 |  |
| O sistema deve permitir o cadastro de novas criptomoedas. | 005 | Alta | Baixa - 3 | Acessível apenas pelo Administrador |
| O sistema deve permitir alteração e exclusão das criptomoedas do sistema. | 006 | Alta | Baixa - 4 | Acessível apenas pelo Administrador |
| O sistema deve permitir a busca das criptomoedas por seu nome. | 007 | Baixa | Média - 7 |  |
| O sistema deve permitir adicionar o valor das criptomoedas todos os dias, de acordo com as criptomoedas cadastradas. | 008 | Alta | Média - 8 | O valor das criptomoedas deve ser alterados uma vez no dia. Acessível apenas pelo Administrador. |
| O sistema deve permitir busca das criptomoedas por data. | 009 | Alta | Média – 8 | Deve ser exibido o nome, símbolo e data das criptomoedas. |
| O sistema deve permitir a alteração do valor da criptomoeda, a alteração e deleção da mesma. | 010 | Alta | Média - 7 | Acessível apenas pelo Administrador. |
| O sistema deve permitir visualizar todo histórico de transações realizadas. | 011 | Alta | Alta - 13 | O usuário pode visualizar suas transações. Já o Administrador, visualiza as transações de todos os usuários. |
| O sistema deve exibir o tipo de transação – Compra/Venda. Além de ordená-las por data. O administrador poderá filtrar a lista pelo nome do usuário, ou pelo nome da criptomoeda. O usuário poderá filtrar a busca pelo nome da criptomoeda. | 012 | Alta | Alta - 12 | Devem ser exibidas todas as informações da transação: data, nome do usuário, tipo de transação, valor, nome da criptomoeda e sua quantidade. |
| O sistema deve exibir informações sobre a conta do cliente, além de sua conta vinculada. | 013 | Alta | Baixa - 6 | O saldo total do cliente deve ser exibido. O valor deve ser baseado no valor atual das criptomoedas cujo usuário tenha investimento. |
| O sistema deve exibir uma lista de todas as aplicações do usuário. | 014 | Alta | Média - 9 | O Administrador deve visualizar uma lista das aplicações de todos os usuários. |
| O sistema deve conter um gráfico com o valor atual de todas as criptomoedas. | 015 | Alta | Alta – 14 |  |
| O sistema deve conter um gráfico para cada criptomoeda com o valor dos seus últimos 7 dias + o dia atual. | 016 | Alta | Alta – 14 |  |
| O sistema deve conter um gráfico para cada criptomoeda com o valor dos seus últimos 30 dias + o dia atual. | 017 | Alta | Alta - 14 |  |
| O sistema deve conter um gráfico com o valor de todas as criptomoedas nos últimos 7 dias + o dia atual. | 018 | Alta | Alta - 14 |  |
| O sistema deve conter um gráfico com o valor de todas as criptomoedas nos últimos 30 dias + o dia atual. | 019 | Alta | Alta - 14 |  |
| O sistema deve conter um gráfico dos últimos 7 dias + dia atual, indicando a porcentagem do valor da criptomoeda em relação ao dia anterior. | 020 | Alta | Alta - 16 |  |
| O sistema deve conter um gráfico dos últimos 30 dias + dia atual, indicando a porcentagem do valor da criptomoeda em relação ao dia anterior. | 021 | Alta | Alta - 16 |  |
| O sistema deve exibir todos os investimentos do usuário, indicando o valor investido e o valor atual. Além de gráficos semanais e mensais indicando seu investimento. | 022 | Alta | Alta – 16 | Para o administrador, deve ser exibido o valor total investido nas criptomoedas por todos seus clientes. |
| O sistema deve permitir a alteração do telefone, e-mail e senha do usuário. | 023 | Alta | Média – 8 | Para alterar a senha, é necessário inserir a senha atual. Somente o próprio usuário pode fazer alterações em sua conta. |

Fonte: Paola Regina Garbato (2020)

Tabela 3 - Requisitos Funcionais e Regras de Negócio - Aplicativo Web

| **RF** | **CÓD.** | **PRIORIDADE** | **GRAU DE DIFICULDADE** | **REGRA DE NEGÓCIO** |
| --- | --- | --- | --- | --- |
| O sistema deve ser acessível apenas pelo administrador. | 001 | Alta | Baixa - 2 |  |
| O sistema deve realizar o login com o CPF e senha. | 002 | Alta | Baixa - 3 |  |
| O sistema deve exibir o valor atual das criptomoedas. | 003 | Alta | Baixa – 5 |  |
| O sistema deve indicar se o valor da criptomoeda está superior ou inferior ao valor do dia anterior. | 004 | Média | Baixa - 6 |  |
| O sistema deve conter um gráfico para cada criptomoeda exibindo o valor dos últimos 7 dias. | 005 | Alta | Média – 9 |  |
| O sistema deve listar todos os usuários. | 006 | Alta | Média – 9 |  |
| O sistema deve listar todas as transações ordenadas por data. | 007 | Alta | Média - 9 |  |
| O sistema deve calcular a quantidade total de investimentos em cada criptomoeda. | 008 | Alta | Média - 9 |  |

Fonte: Paola Regina Garbato (2020)

# ANÁLISE DE SISTEMAS ORIENTADA A OBJETOS

## Diagrama de Casos de uso

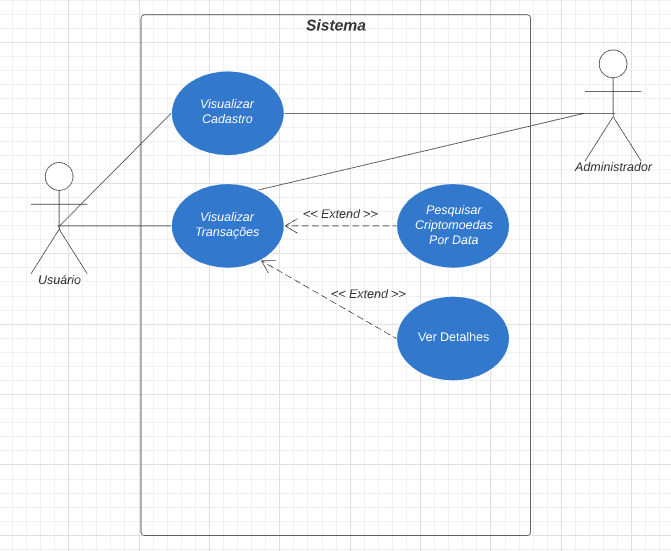
O Diagrama de Caso de Uso descreve as funcionalidades do sistema, como o sistema age conforme os requisitos solicitados pelo usuário. Para a montagem do Diagrama de caso de uso, definimos:

* O Cenário: Como o sistema deve interagir com o usuário
* O Ator: Usuário e Administrador
* Use Case: Conforme solicitado pelo Usuário
* A Comunicação: Tudo que liga o ator com o caso de uso

E em seguida, as funcionalidades do sistema:

* Visualizar cadastro
* Visualizar criptomoedas
* Visualizar transações
* Alterar criptomoeda
* Deletar criptomoeda
* Adicionar criptomoeda
* Pesquisar criptomoedas por data
* Ver detalhes

Figura 1 - Diagrama de Casos de Uso - Visualizar Cadastro e Transações



Fonte: Gabrielle de Lima Gabaldi (2020)

Como podemos ver na figura 1, o usuário, e o administrador podem visualizar os cadastros, e as transações. O <<extend>> indica que os itens “Pesquisar Criptomoedas por Data”, e “Ver Detalhes”, são opcionais.

Podemos observar na figura 2, que o Administrador deve realizar o login para ter acesso a visualização das Criptomoedas, indicado pelo <<include>>. E pode (opcionalmente) alterar, deletar, e adicionar criptomoedas. Para isso, o administrador deve selecionar a criptomoeda, para assim realizar as devidas alterações na mesma.

Figura 2 - Diagrama de Casos de Uso - Criptomoedas



Fonte: Gabrielle de Lima Gabaldi (2020)

## Diagrama de Sequência

O Diagrama de Sequência determina a Linha de vida, e as mensagens trocadas entre usuário e administrador, determinando em que ordem acontecerá os eventos, para demonstrar seu desempenho até o fim da linha de vida. Os atores são os mesmos do Diagrama de Caso de uso. Como mostra a figura 3, o diagrama de sequência indica:

1. O usuário realiza o login no sistema, e o sistema retorna uma mensagem de validação – liberando o acesso.

2. O usuário solicita a visualização de criptomoedas, e o sistema retorna a listagem das Criptomoedas.

3. O usuário seleciona a criptomoeda desejada, e o sistema retorna a criptomoeda.

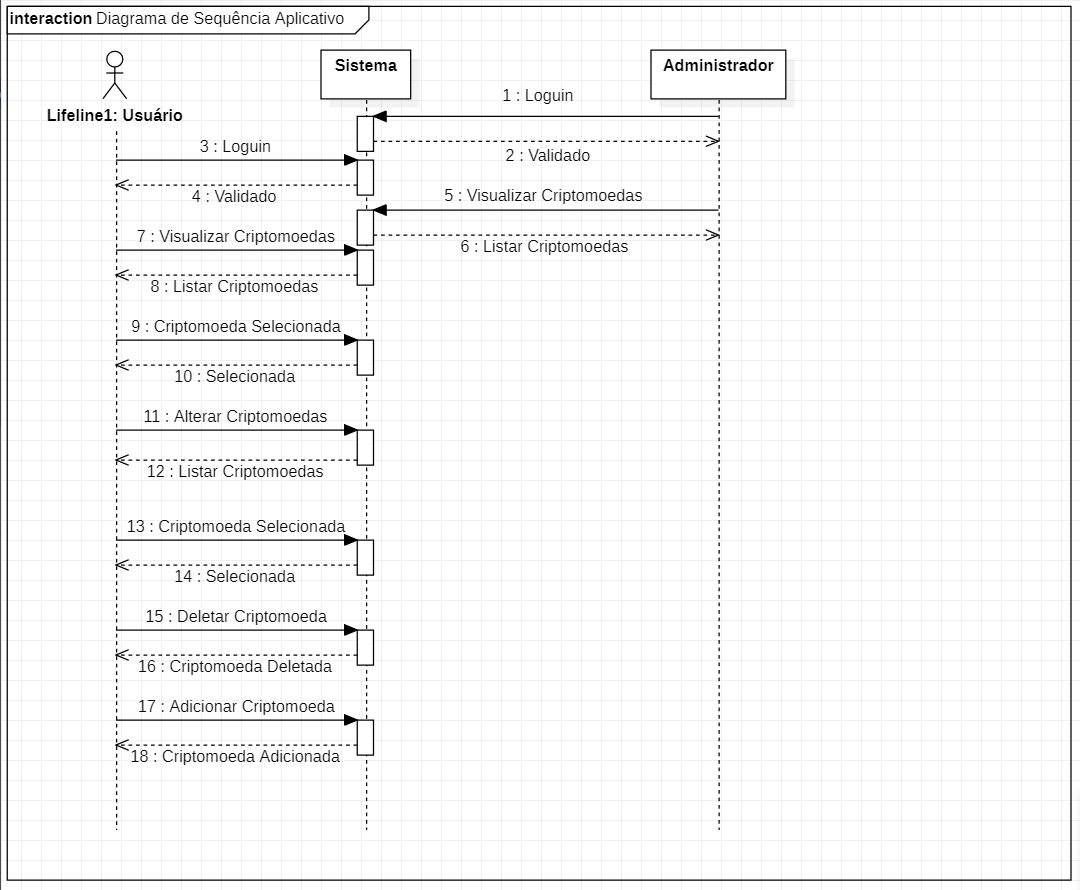
4. O usuário pede para deletar a criptomoeda, e o sistema retorna a criptomoeda solicitada pelo usuário.

5. O usuário pede para adicionar uma criptomoeda, e o sistema responde com a adição da criptomoeda.

6. O administrador realiza o login, e o sistema retorna validação - liberado o acesso ao sistema.

7. O administrador pede para visualizar as criptomoedas, o sistema lista as criptomoedas.

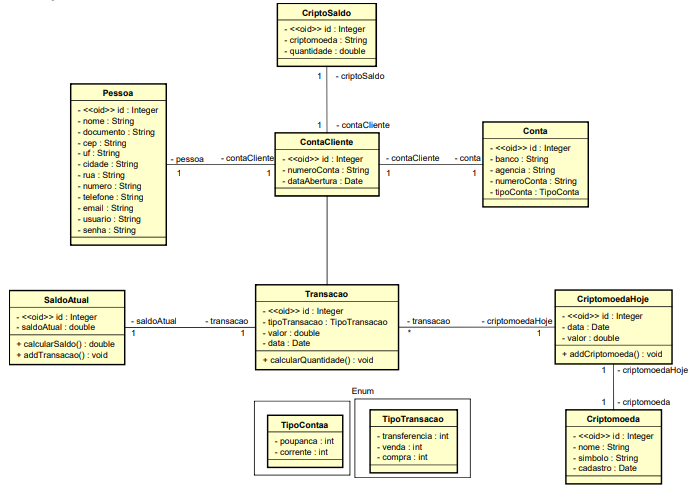
Figura 3 - Diagrama de Sequência

  
 Fonte: Gabrielle Gabaldi e Kaique Marcelino (2020)

## Diagrama de Classes

Em comparação a primeira parte do projeto, o diagrama de classes sofreu pequenas alterações, como pode ser visto na figura 1. Essas alterações se devem ao fato de, na primeira parte do projeto, o conhecimento em UML e Programação Orientada a Objetos serem extremamente rasos. Com a continuidade dos estudos, com um conhecimento maior em ambos, foi possível desenvolver um diagrama mais correto (assim como o programa).

Figura 4 - Diagrama de Classe



Fonte: Paola Regina Garbato (2020)

As alterações, foram: a classe Pessoa, antes era dividida em 4: Pessoa, e, Pessoa Física e Pessoa Jurídica herdando da primeira, e Cliente. Além disso, essa classe recebeu usuário e senha.

A classe ContaCliente, passou a armazenar somente o número da conta, e a data de abertura. Foi criada uma classe chamada Conta, pensando que, para que sejam feitos depósitos e transferências com a conta BlockTech, é necessário que haja uma conta vinculada. Para definir o tipo de conta, foi criado um Enum.

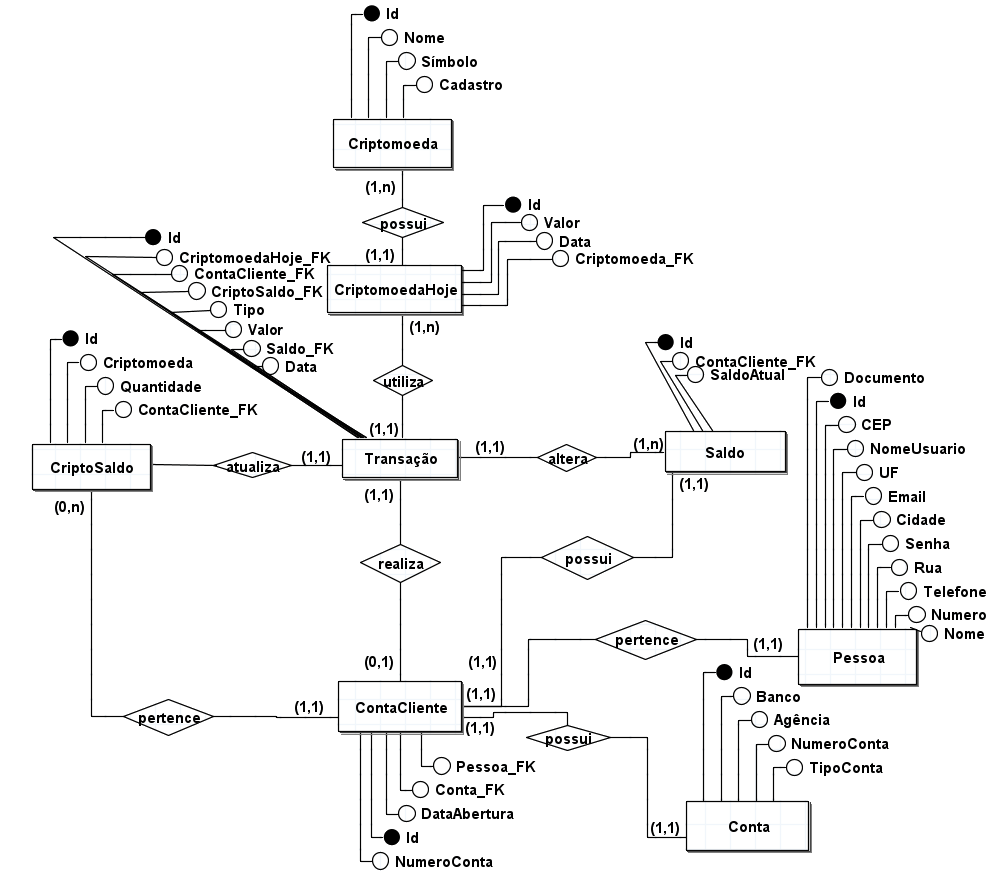
Agora, a classe Criptomoeda, armazena apenas o nome, o símbolo e sua data de cadastro, e, para adicionar seu valor, que é atualizado diariamente, foi criada uma nova classe chamada CriptomoedaHoje.

A classe Transação, substitui a antiga classe Operação. A classe Operação, antes, estava associada a três classes: Compra, Venda e Transferência. Essas classes, foram definidas na classe Transação pelo atributo “TipoTransacao”, e foi criado um Enum para adicionar essas três opções. Para armazenar o saldo em real e em criptomoedas dos clientes, foram criadas respectivamente as classes SaldoAtual, e CriptoSaldo.

# BANCO DE DADOS

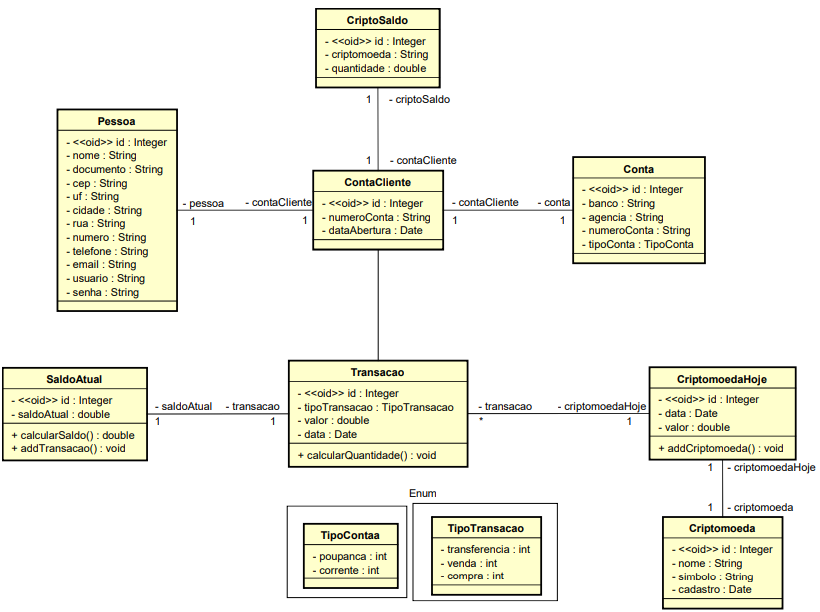
Apesar do banco de dados ter sido elaborado na primeira parte do projeto, ao longo do desenvolvimento do projeto, foram necessárias pequenas alterações no mesmo. Todas as informações necessárias dos clientes, foram armazenadas, - porém, diferente do entregue na primeira parte do projeto, - apenas em uma tabela. Outra mudança, foi a criação de um usuário e senha, e também a tabela Operação que existia, passou a ser um ENUM no programa, e indicado como “tipo” na tabela transação, como pode ser visto na figura 1 – notação Peter Chen, e na figura 2 – Diagrama Entidade Relacionamento, e também na figura 3 - Diagrama feito no SGBD, que continua sendo o SQL Server Management Studio.

Figura 5 - Notação Peter Chen

****

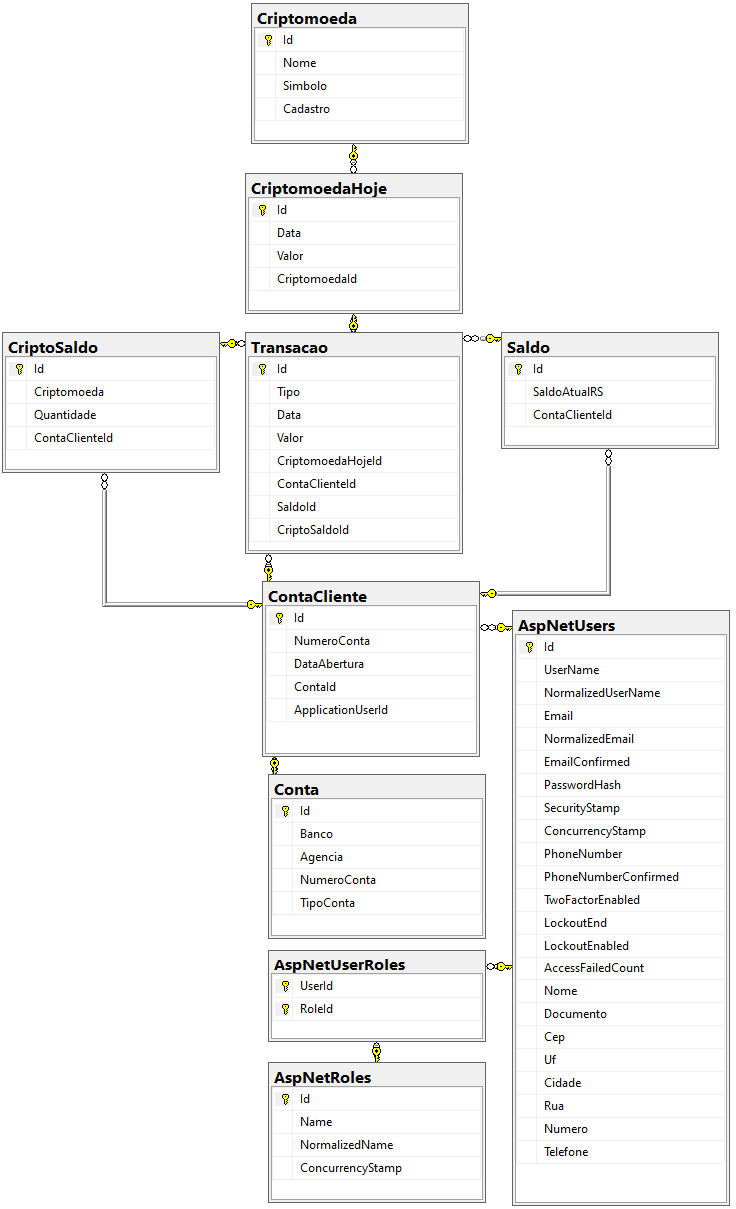
Fonte: Paola Regina Garbato (2020)

Figura 6 - DER



Fonte: Paola Regina Garbato (2020)

Figura 7 – DER - SQL Server Management Studio



Fonte: Paola Regina Garbato (2020)

## Dicionário de Dados

Devido a essas alterações, o dicionário de dados também foi atualizado.

Tabela 4 - Dicionário de Dados - Usuário

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Atributo** | | **Tipo de Dados** | **Tamanho** | | **Restrições** | | **Descrição** |
| Id | Varchar | | | 60 bytes | | PK, NOT NULL | Código criptografado de identificação do usuário |
| NomeUsuario | Varchar | | | 70 bytes | | NOT NULL | Nome de usuário |
| Nome | Varchar | | | 60 bytes | | NOT NULL | Nome ou Razão Social do usuário |
| Senha | Varchar | | | 20 bytes | | NOT NULL | Senha de acesso |
| Endereco | Varchar | | | 70 bytes | | NOT NULL | Endereço do cliente |
| CEP | Varchar | | | 8 bytes | | NOT NULL | CEP do cliente |
| UF | Char | | | 2 bytes | | NOT NULL | Estado do cliente |
| Cidade | Varchar | | | 60 bytes | | NOT NULL | Cidade do cliente |
| NumeroCasa | Char | | | 6 bytes | | NOT NULL | Número da casa do cliente |
| Telefone | Varchar | | | 12 bytes | | NOT NULL | Telefone do cliente |
| Email | Varchar | | | 70 bytes | | NOT NULL | E-mail do cliente |
| Documento | Varchar | | | 20 bytes | | NOT NULL | CPF/CNPJ do usuário |

Fonte: Paola Regina Garbato (2020)

Tabela 5 - Dicionário de Dados – Conta

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Atributo** | **Tipo de Dados** | **Tamanho** | **Restrições** | **Descrição** |
| Id | Inteiro | 4 bytes | PK, NOT NULL | Código de identificação da conta vinculada |
| Banco | Varchar | 40 bytes | NOT NULL | Nome do banco |
| Agência | Varchar | 8 bytes | NOT NULL | Número da agência |
| NumeroConta | Varchar | 20 bytes | NOT NULL | Número da conta |
| TipoConta | Varchar | 16 bytes | NOT NULL | Tipo de conta |

Fonte: Paola Regina Garbato (2020)

Tabela 6 - Dicionário de Dados - ContaCliente

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Atributo** | **Tipo de Dados** | **Tamanho** | **Restrições** | **Descrição** |
| Id | Inteiro | 4 bytes | PK, NOT NULL | Código de identificação da conta do usuário |
| NumeroConta | Varchar | 8 bytes | NOT NULL | Número da conta |
| DataAbertura | Date | 8 bytes | NOT NULL | Data de abertura da conta |
| IDConta | Inteiro | 4 bytes | FK, NOT NULL | Código de identificação da conta vinculada |
| IDUsuario | Varchar | 60 bytes | FK, NOT NULL | Código de identificação do usuário |

Fonte: Paola Regina Garbato (2020)

Tabela 7 - Dicionário de Dados - Criptomoeda

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Atributo** | **Tipo de Dados** | **Tamanho** | **Restrições** | **Descrição** |
| Id | Inteiro | 4 bytes | PK, NOT NULL | Código de identificação da criptomoeda |
| Nome | Varchar | 70 bytes | NOT NULL | Nome da criptomoeda |
| Simbolo | Varchar | 4 bytes | NOT NULL | Símbolo da criptomoeda |
| Cadastro | Date | 8 bytes | NOT NULL | Data de cadastro da criptomoeda |

Fonte: Paola Regina Garbato (2020)

Tabela 8 - Dicionário de Dados – CriptomoedaHoje

| **Atributo** | **Tipo de Dados** | **Tamanho** | **Restrições** | **Descrição** |
| --- | --- | --- | --- | --- |
| Id | Inteiro | 4 bytes | PK, NOT NULL | Código de identificação da criptomoeda hoje |
| Data | Date | 8 bytes | NOT NULL | Data de atualização da criptomoeda |
| Valor | Decimal | 8 bytes | NOT NULL | Valor da criptomoeda |
| IDCriptomoeda | Inteiro | 8 bytes | FK, NOT NULL | Código de identificação da criptomoeda associada |

Fonte: Paola Regina Garbato (2020)

Tabela 9 - Dicionário de Dados - CriptoSaldo

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Atributo** | **Tipo de Dados** | **Tamanho** | **Restrições** | **Descrição** |
| Id | Inteiro | 4 bytes | PK, NOT NULL | Código de identificação do saldo de criptomoedas |
| Criptomoeda | Varchar | 70 bytes | NOT NULL | Nome da criptomoeda |
| Quantidade | Decimal | 8 bytes | NOT NULL | Quantidade de criptomoeda |
| IDContaCliente | Inteiro | 4 bytes | FK, NOT NULL | Código de identificação da conta do cliente associado |

Fonte: Paola Regina Garbato (2020)

Tabela 10 - Dicionário de Dados - Saldo

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Atributo** | **Tipo de Dados** | **Tamanho** | **Restrições** | **Descrição** |
| Id | Inteiro | 4 bytes | PK, NOT NULL | Código de identificação do saldo |
| SaldoAtualRS | Decimal | 8 bytes | NOT NULL | Saldo atual em real |
| IDContaCliente | Inteiro | 4 bytes | FK, NOT NULL | Código de identificação da conta do cliente associado |

Fonte: Paola Regina Garbato (2020)

Tabela 11 - Dicionário de Dados - Transacao

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Atributo** | **Tipo de Dados** | **Tamanho** | **Restrições** | **Descrição** |
| Id | Inteiro | 4 bytes | PK, NOT NULL | Código criptografado de identificação do usuário |
| Tipo | Varchar | 20 bytes | NOT NULL | Tipo de transação |
| Data | Date | 8 bytes | NOT NULL | Data da transação |
| Valor | Decimal | 20 bytes | NOT NULL | Valor da transação |
| IDCriptomoedaHoje | Inteiro | 4 bytes | FK, NOT NULL | Código de identificação da criptomoeda hoje |
| IDContaCliente | Inteiro | 4 bytes | FK, NOT NULL | Código de identificação da conta cliente |
| IDSaldo | Inteiro | 4 bytes | FK, NOT NULL | Código de identificação do saldo |
| IDCriptoSaldo | Inteiro | 4 bytes | FK, NOT NULL | Código de identificação do cripto saldo |

Fonte: Paola Regina Garbato (2020)

# PROGRAMAÇÃO ORIENTADA A OBJETOS

## Web - ASP.NET

A versão web do software foi desenvolvida em C#, utilizando a IDE Visual Studio, que traz um grande suporte ao desenvolvedor. Foi utilizada a estrutura ASP.NET CORE MVC. O ASP.NET Core é uma extensão do .NET Framework.

### **Padrão MVC**

O MVC é um padrão de desenvolvimento de aplicações, que a divide em três partes: Model (modelo), Control (controlador) e View (visão).

**Modelo**:

* Mantém a estrutura dos dados, conhecido como "o sistema",
* Possui as entidades do negócio e os serviços,
* É a representação das regras do negócio,
* Quem possui a inteligência do sistema.

**Controlador**:

* Recebe e trata a interação do usuário com o sistema, como um botão que o usuário clica, - quem recebe a ação é o controlador,
* É o intermediário entre o sistema e as telas do sistema (View)

**Visão**:

* Apresentação para o usuário
* Define as estruturas e comportamento das telas

Essas camadas se relacionam da seguinte forma: por meio das telas (view), os usuários interagem com o sistema por meio de requisições. As requisições vão para o controlador (controller) que faz chamadas ao sistema (model) e retorna as respostas para as telas.

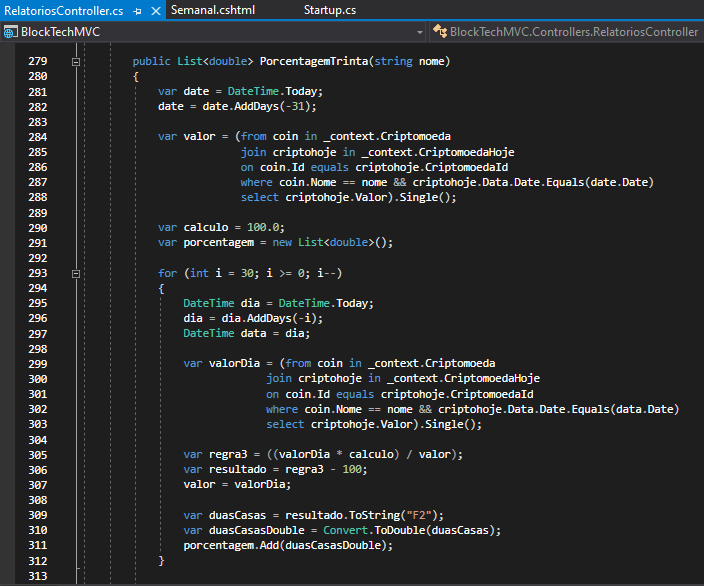
### **Entity Framework**

Pela dificuldade em criar um sistema orientado a objetos se comunicando com um banco de dados relacional, se utiliza muito ORM – Object-Relational Mappers, que significa Mapeamento objeto-relacional. A ORM é uma ferramenta que permite realizar a programação orientada a objetos, e automaticamente, ela é mapeada para o banco de dados relacional. A ORM da Microsoft se chama Entity Framework.

Para que essa ferramenta funcione, ela precisa de Providers. Os providers, são implementações da comunicação com o banco de dados utilizado, como por exemplo: Provider do SQLServer, Provider do MYSQL...

O Entity Framework possui duas classes principais que utilizamos, sendo elas, a “DBContext” e a “DbSet <TEntity>”.No DbSet, utilizamos operações do LINQ que, em tempo de execução, são convertidas para consultas SQL, como mostra o exemplo da figura 8.

Figura 8 - LINQ



Fonte: Paola Regina Garbato (2020)

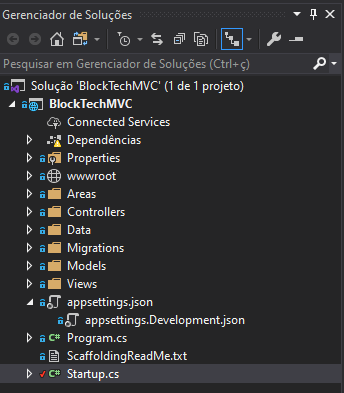
Esse exemplo mostra um método utilizado para obter a porcentagem do valor da criptomoeda, em relação ao dia anterior. Primeiramente, a data atual é armazenada na variável “date”, como mostra a linha 281. Foi utilizado DateTime.Today ao invés de DateTime.Now por não ser necessário obter o horário de cadastro das criptomoedas. Em seguida, foi utilizado o método AddDays para obter a data de 31 dias atrás. Esse método é necessário para que não haja erro de cálculos, se apenas subtraísse 31, como um número inteiro, o cálculo não daria certo – como ocorreu durante o desenvolvimento do software. Em seguida, foi realizada uma consulta com LINQ, guardando na variável “valorDia”, o valor da criptomoeda, 31 dias antes do dia atual, comparando o nome da criptomoeda com o nome passado por parâmetro no método, e a data com a data da variável “date”, declarada no início do método. Em seguida, foi criada uma lista chamada “porcentagem”, onde foram guardados todos os valores – utilizados nos gráficos. Então, foi feito um laço for, para percorrer um mês, ou seja, 30 dias. O laço for se inicia, com a variável i recebendo o valor 30, decrementa 1, e ocorre enquanto a variável for igual a 0. Dentro desse laço, foi declarada uma variável “dia”, que recebe o valor do dia atual, e subtrai o valor de dias de acordo com o valor da variável do laço for, então, primeiro recebe o valor de 30 dias atrás, depois 29, e assim, até retornar ao dia atual, onde subtrai 0 dias. Para cada um desses dias, foi feita uma consulta com LINQ, comparando novamente o nome da criptomoeda passada por parâmetro no método, e comparando a data com a armazenada na variável “data”, que recebe o valor da variável dia. Após obter o valor da criptomoeda no dia desejado, é feita uma regra de 3, sendo que, na primeira vez, é utilizado o valor obtido na linha 284, que seriam de 31 dias atrás. Após esse cálculo, para que seja obtido o valor de 30 dias, ao invés de 31, a variável “valor”, recebe o valor da criptomoeda – obtido na linha 299. Para que o resultado desse cálculo seja exibido apenas com duas casas decimais, ele foi convertido para string, com duas casas decimais. Em seguida, foi novamente convertido para o tipo double, e guardado dentro da Lista. E por fim, é retornada essa lista – como o método mostra, o retorno do mesmo é uma lista do tipo double.

Esse método foi chamado por todas as criptomoedas, para confecção de um gráfico para cada uma delas.

### **Estrutura do Projeto**

A figura 9 mostra toda a solução do projeto.

Figura 9 - Estrutura do Projeto



Fonte: Paola Regina Garbato (2020)

Algumas pastas, que serão descritas a seguir, sofreram alterações ao longo do projeto, outras, como “Properties”, não.

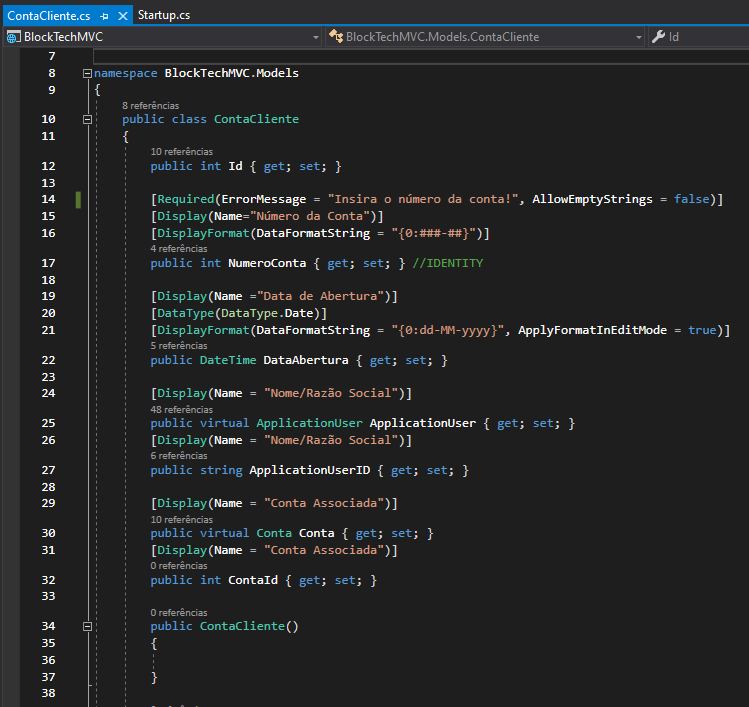
Em “Dependências” estão todos os pacotes do NuGet que foram utilizados no projeto, como o Microsoft.EntityFrameworkCore.SqlServer, - como foi citado, essa ORM facilitou o desenvolvimento de um projeto Orientado a Objetos com um banco de dados relacional. Com ele, o projeto foi feito seguindo o “Code-First”.

Em “wwwroot” estão todos os ícones, imagens, arquivos css, bootstrap, javascript utilizados no projeto – cada um em sua respectiva sub-pasta.

Em “Areas” estão os arquivos do Identity, como o Razor da página de login e cadastro. O Razor possibilita criarmos páginas HTML juntamente com o C#. Ele possui a extensão “cshtml”, sendo CS de CSharp.

Em “Controllers” estão todos os controladores citados anteriormente. Para que seja mais bem exemplificado, primeiro serão citados os Models. Na pasta “Model”, se encontram todas as classes – primeira coisa a ser criada no projeto. Como foi utilizado o Code-First, nessas classes já foram indicadas as relações que existiriam entre elas no banco de dados, como mostra o exemplo da figura 10.

Figura 10 - Model - ContaCliente

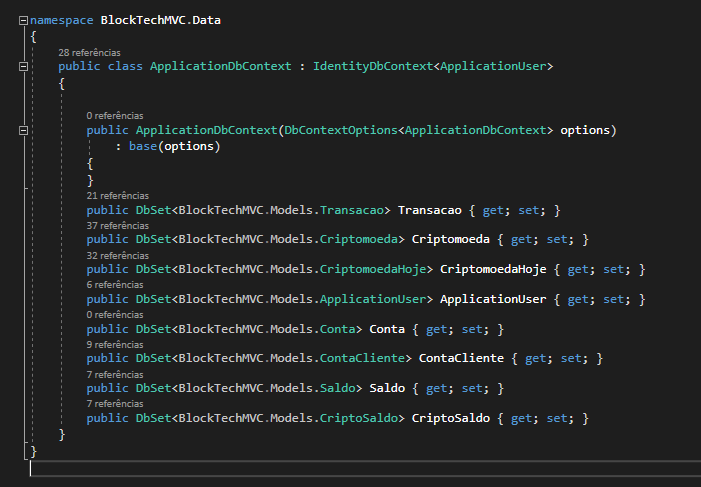


Fonte: Paola Regina Garbato (2020)

Como mostra a figura 10, na própria classe já foi inserido um Id. Para indicar uma chave estrangeira do banco de dados, foi inserido o nome da entidade (classe) seguido do “ID”, exemplo na linha 32. O tipo da variável - que para Id geralmente é número inteiro, deve ser o mesmo tipo do ID da outra classe. Porém, como pode ser observado na linha 27, foi utilizado o tipo string, isso ocorreu, pois, na tabela de usuários, o ID é criptografado, assim como a senha, então, por haver números e letras, não é possível utilizar o tipo inteiro. Também foram utilizadas DataAnnotations como pode ser visto, para adicionar nomes diferentes aos das variáveis para serem exibidos ao usuário, definir tipos e máscaras, ou até mesmo mensagens de erro caso campos obrigatórios (Required) não sejam preenchidos.

Na pasta “Data” existe o arquivo ApplicationDbContext - que herda do DbContext. Essa é a classe responsável para a comunicação com o banco de dados. Toda classe criada no Model deve ser inserida no ApplicationDbContext para que assim sejam criadas as tabelas correspondentes no banco de dados, como mostra a figura 11.

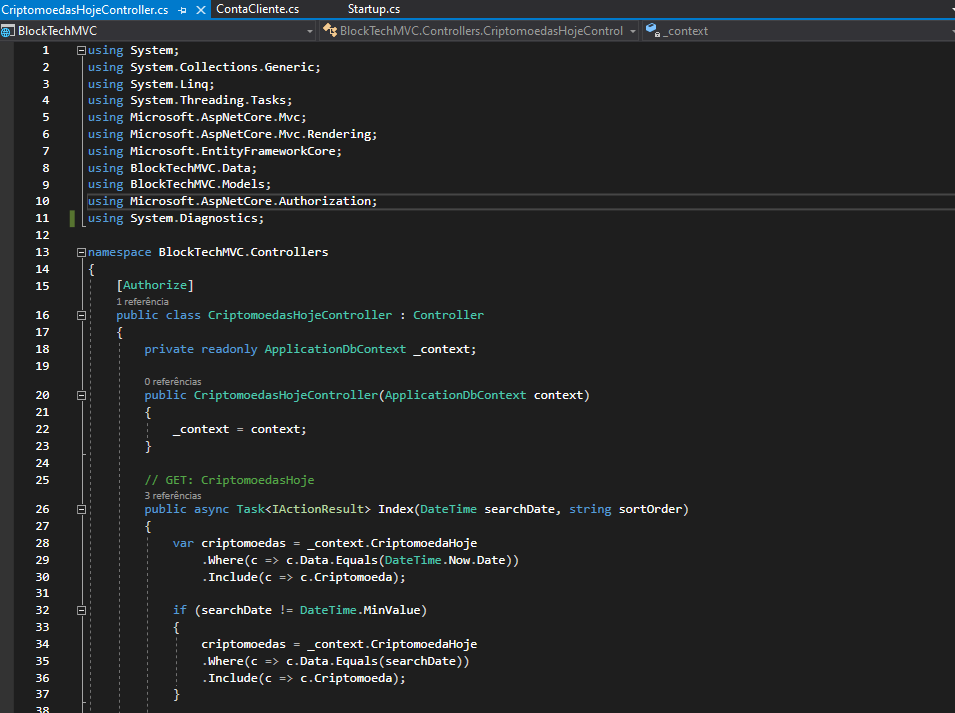
Figura 11 - ApplicationDbContext



Fonte: Paola Regina Garbato (2020)

Após a criação de todas as classes, foi criado o controlador das classes, que contém informações que serão exibidas para os clientes. Todas os controladores criados herdam da classe Controller, como pode ser visto na figura 12.

Figura 12 - Controlador - CriptomoedasHoje

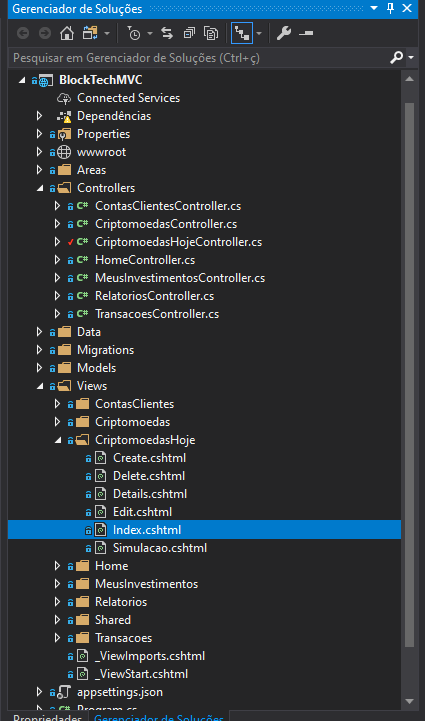


Fonte: Paola Regina Garbato (2020)

No início de todos os controladores, possui uma instância do ApplicationDbContext para que possa consumir os dados do banco, e em seu construtor, é feita a injeção de dependência.

Para cada método do controlador, que retorna um “IActionResult”, é criado um Razor em uma View. No retorno do método do controlador, estão as informações que serão utilizadas na View. A View possui os arquivos Razor que serão exibidos para o usuário. Para essa exibição, é necessário, primeiro criar uma pasta com o mesmo nome do controlador. Exemplo: Na imagem 8 possui um controlador chamado “CriptomoedasHojeController”, então o nome da pasta na View deve ser “CriptomoedasHoje”. Então, nessa pasta, é criado um arquivo Razor com o mesmo nome do método do controller correspondente com o retorno IActionResult. A figura 9 mostra um exemplo do que foi citado. Todos os controllers possuem pastas correspondentes na View. Como pode ser visto como exemplo, existe o arquivo Razor chamado Index, assim como o método da figura 13.

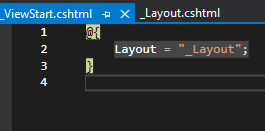
Figura 13 - Views e Controllers



Fonte: Paola Regina Garbato (2020)

Dentro da pasta View, também há a pasta Shared. Nela existe um arquivo Razor chamado \_Layout.cshtml. Nesse arquivo estão: o cabeçalho, o rodapé, os arquivos CSS e JavaScript utilizados, todas as outras páginas herdam dessa. O arquivo \_ViewStart.cshtml, que também pode ser visto na figura 13, faz essa indicação, como pode ser visto na figura 14.

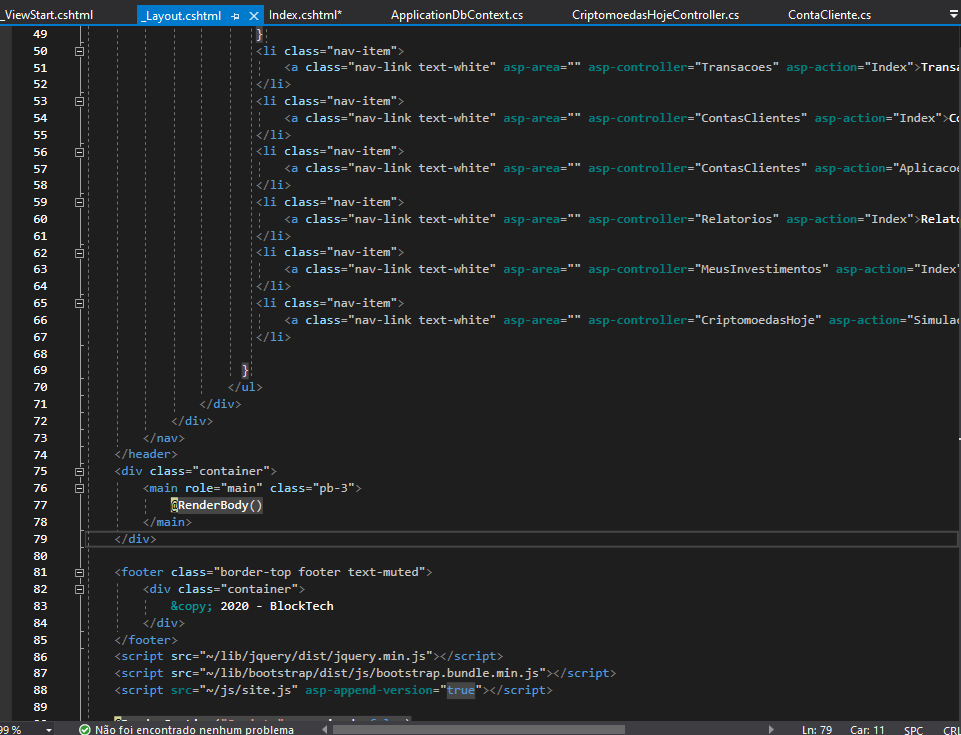
Figura 14 - ViewStart



Fonte: Paola Regina Garbato (2020)

Entre o cabeçalho e o rodapé, está o “@RenderBody()”. Como pode ser visto na linha 77, da figura 15. Ele indica que, todo o conteúdo das páginas que herdam dele, no caso, todas, serão renderizadas nesse local.

Figura 15 - Layout.cshtml



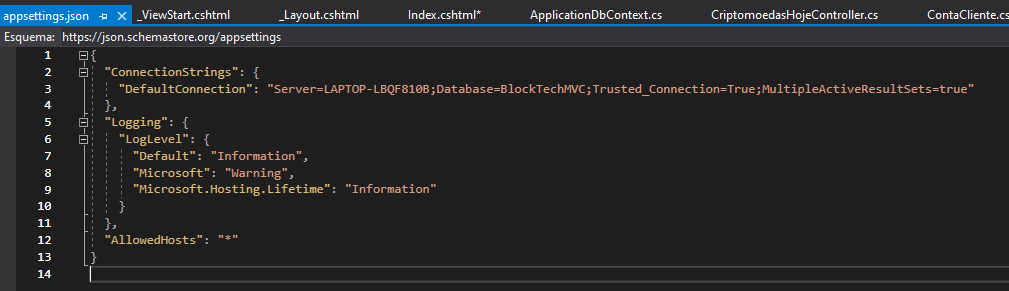
Fonte: Paola Regina Garbato (2020)

Esses arquivos citados começam com um underline, para que o usuário não possa acessá-los.

Na pasta “Migrations” estão os arquivos que indicam as criações de tabelas no banco de dados, e também as alterações. Toda vez que uma classe é alterada, é necessário adicionar uma nova “Migration” para que o banco de dados seja atualizado conforme a classe. Para adicionar essa migration, basta inserir no Console o Gerenciador de Pacotes, o comando: “Add-Migration NomeDaMigration”, e em seguida o comando para atualizar o banco: “Update-Database”.

No arquivo “appsettings.json” está a rota do banco de dados utilizado, como pode ser visto na figura 16.

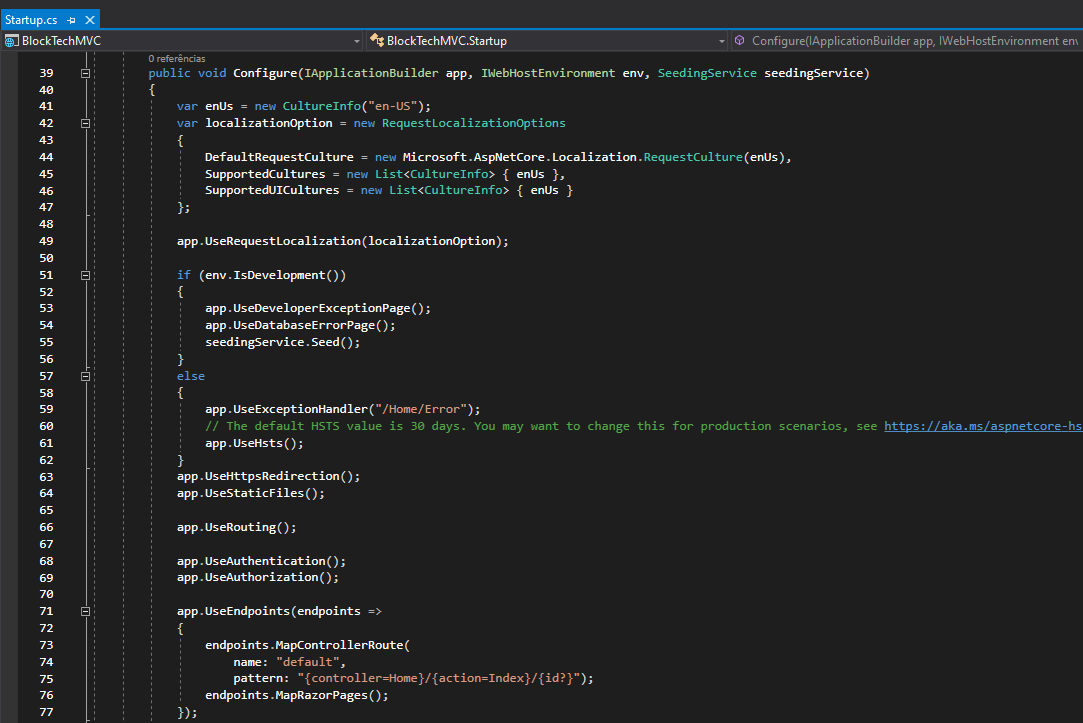
Figura 16 - appsettings.json



Fonte: Paola Regina Garbato (2020)

No arquivo “Startup.cs” estão algumas configurações importantes, como a rota em que o site inicia, que no caso é a Index do Controller Home, como pode ser visto na linha 75 da figura 17.

Figura 17 - Startup.cs



Fonte: Paola Regina Garbato (2020)

Todo o programa, e suas telas podem ser visualizadas no Manual do Usuário – Apêndice B – página X

# CONCLUSÃO

**REFERÊNCIAS**

**GLOSSÁRIO**

**APÊNDICE A – MANUAL DESKTOP**

**APÊNDICE B – MANUAL DO USUÁRIO WEB**

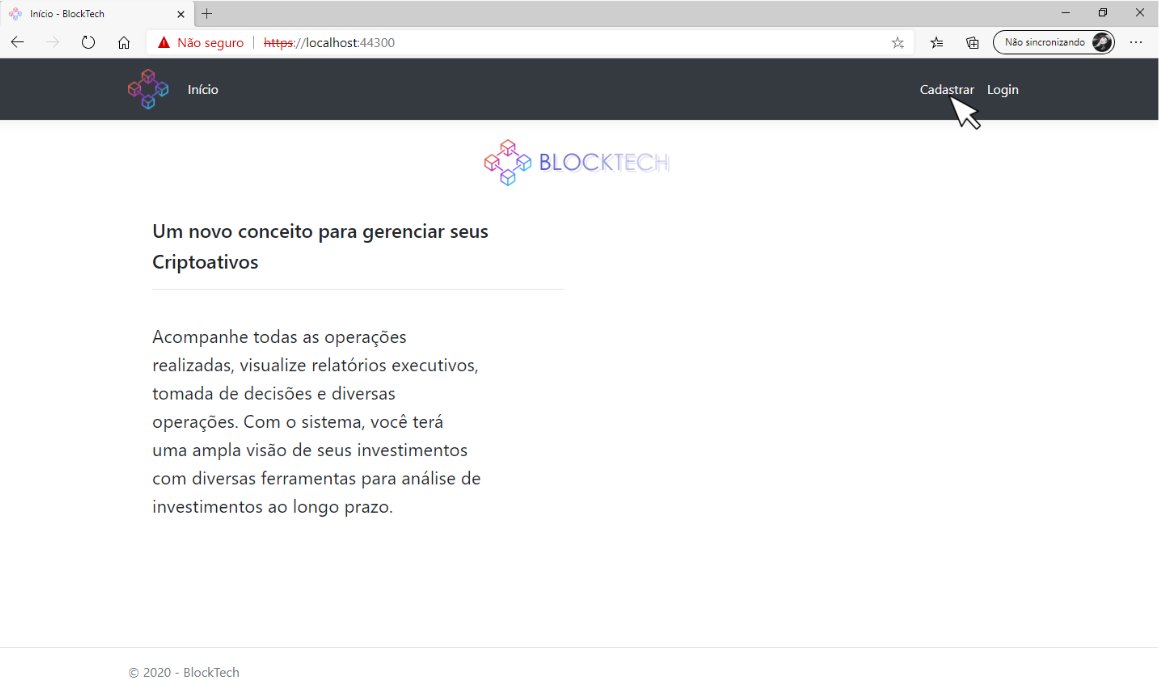
**Introdução**

Agradecemos por fazer parte do novo sistema da BlockTech! Agora, você terá uma grande facilidade para acompanhar seus investimentos e as criptomoedas!

**Primeiro Acesso**

1. Após acessar o site da BlockTech, clique em “Cadastrar”, conforme a Figura 1.

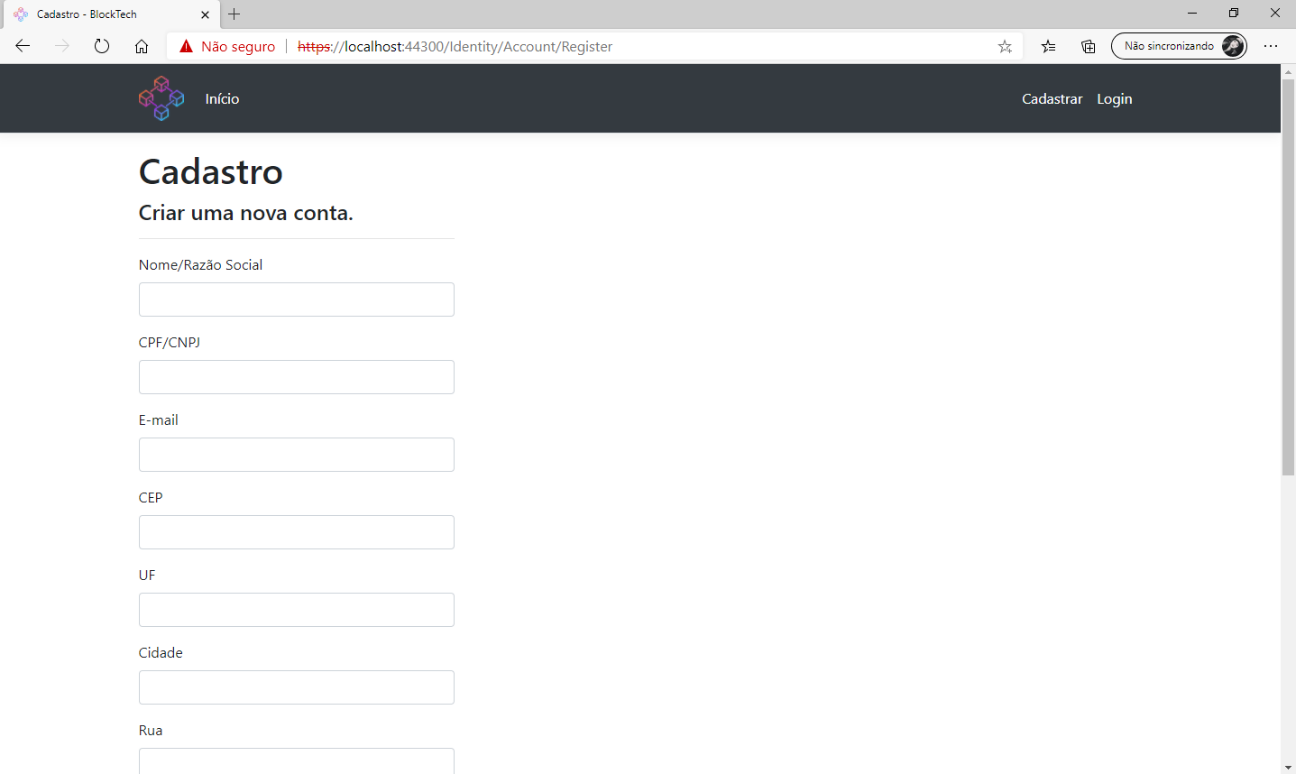
Figura 1 - Aplicativo Web - Cadastro

****

Fonte: Paola Regina Garbato (2020)

2. A tela de cadastro será aberta, conforme a Figura 2, preencha todos os campos.

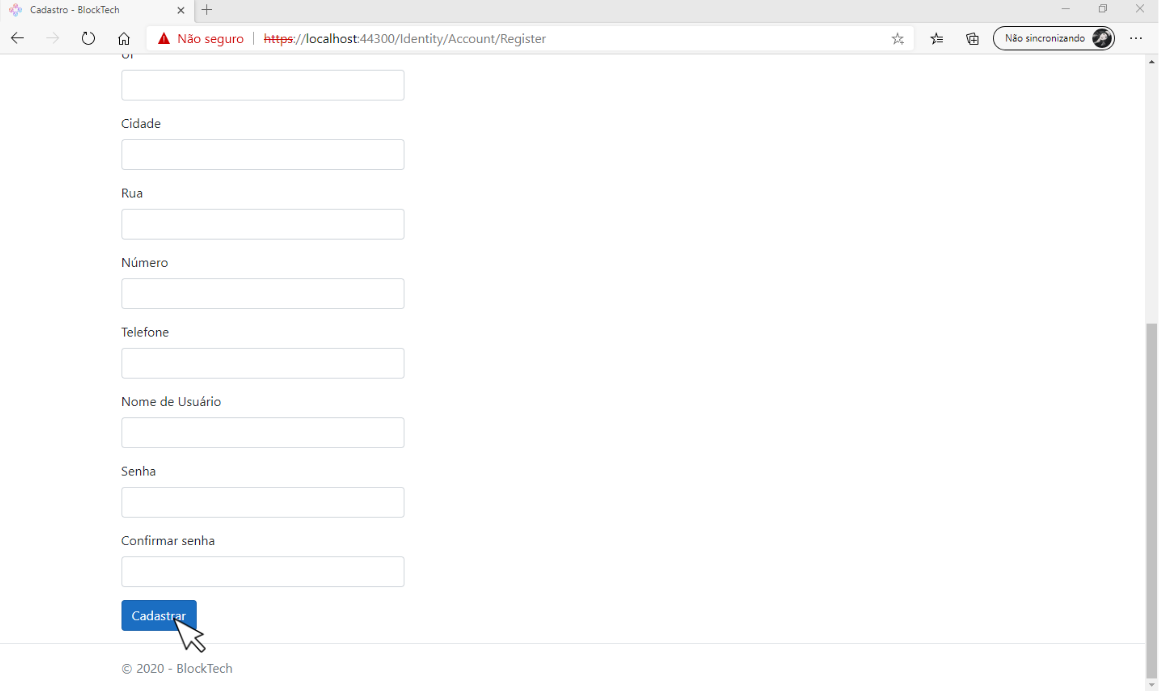
Figura 2 - Aplicativo Web - Realizando Cadastro



Fonte: Paola Regina Garbato (2020)

3. Após preencher todos os campos, clique em “Cadastrar”, conforme a Figura 3.

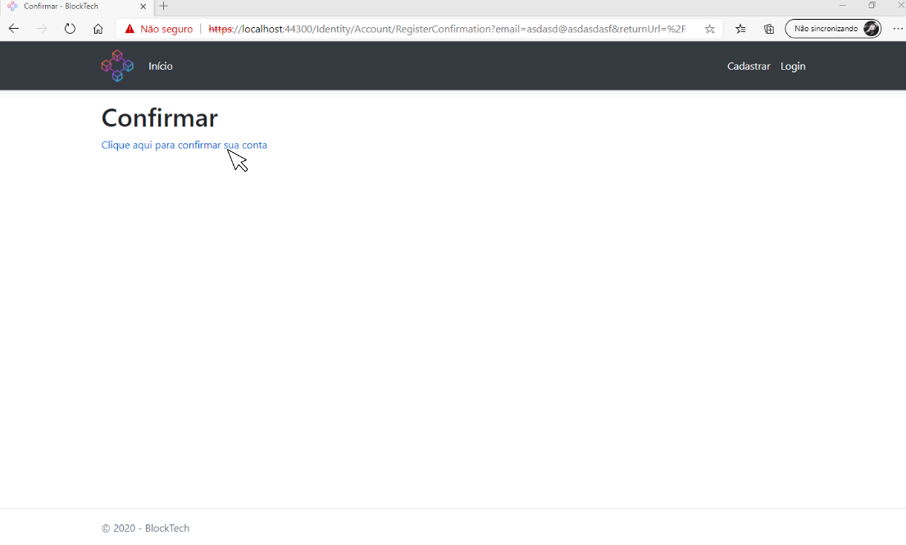
Figura 3 - Aplicativo Web - Finalizando Cadastro

****

Fonte: Paola Regina Garbato (2020)

4. Após realizar o cadastro, confirme a sua conta, clicando em “Clique aqui para confirmar sua conta”, conforme a Figura 4.

Figura 4 - Aplicativo Web - Confirmar Cadastro

****

Fonte: Paola Regina Garbato (2020)

**Usuários Cadastrados**

1. Após acessar o site da BlockTech, clique em “Login”, conforme a Figura 5.

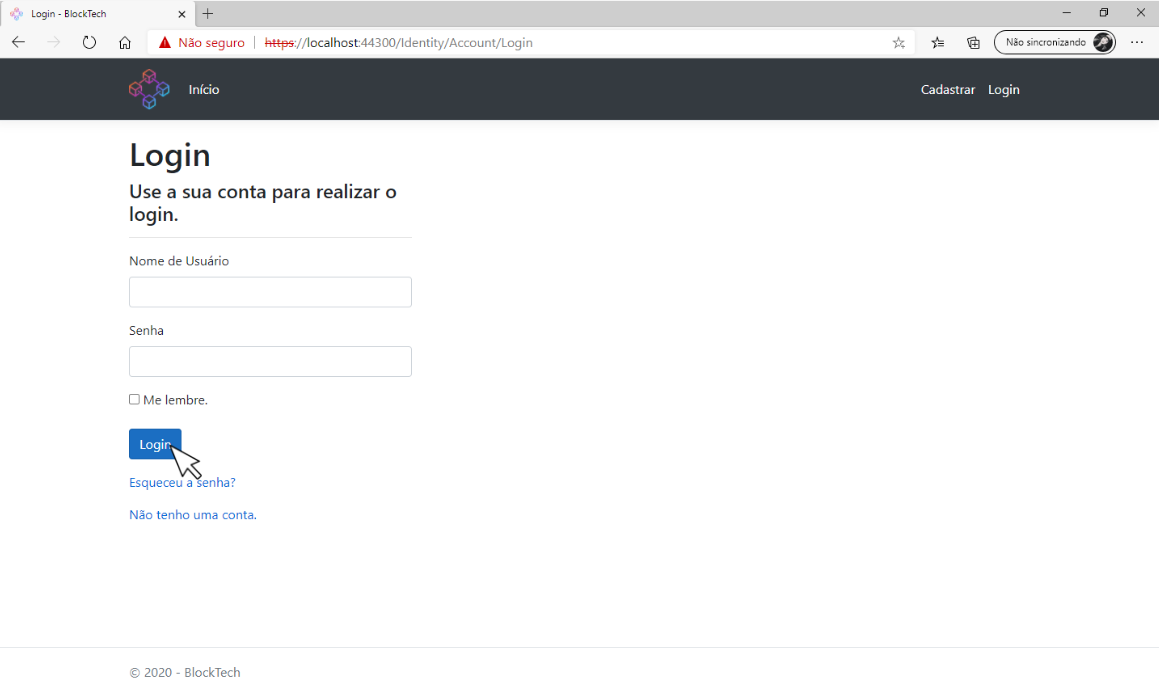
Figura 5 - Aplicativo Web - Login



Fonte: Paola Regina Garbato (2020)

2. Para acessar o sistema, insira seu nome de usuário, sua senha, e selecione o botão “Login”, conforme a Figura 6.

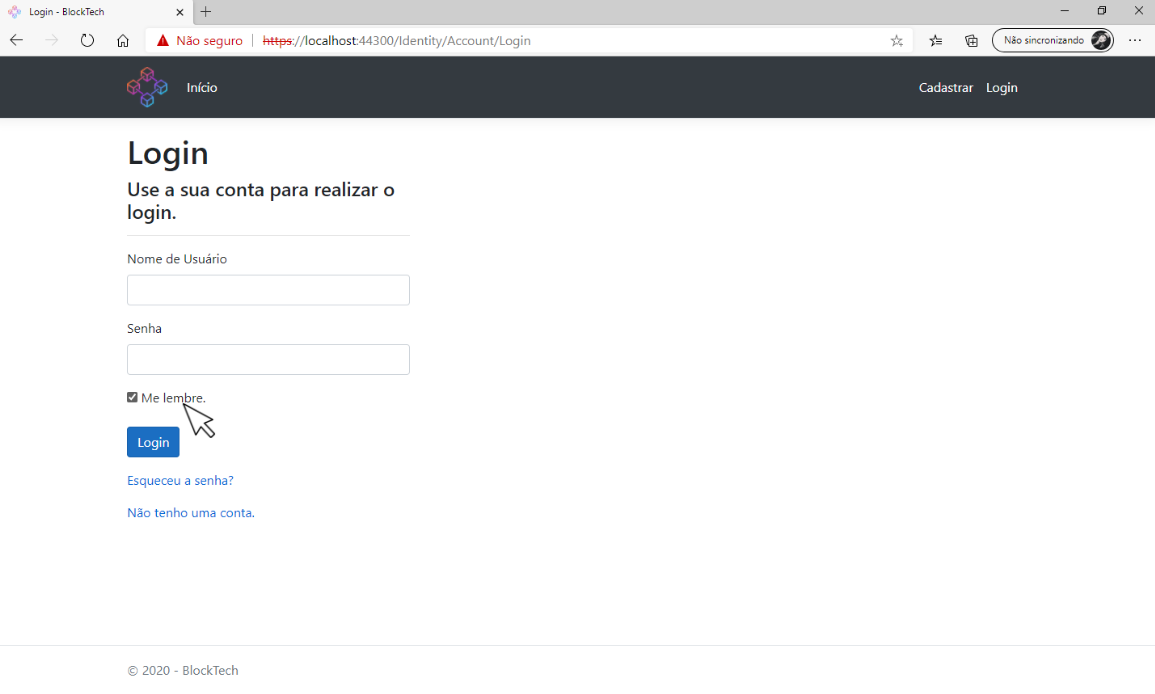
Figura 6 - Aplicativo Web - Realizando o Login



Fonte: Paola Regina Garbato (2020)

3. Caso queira manter-se logado, selecione “Me lembre”, conforme a Figura 7.

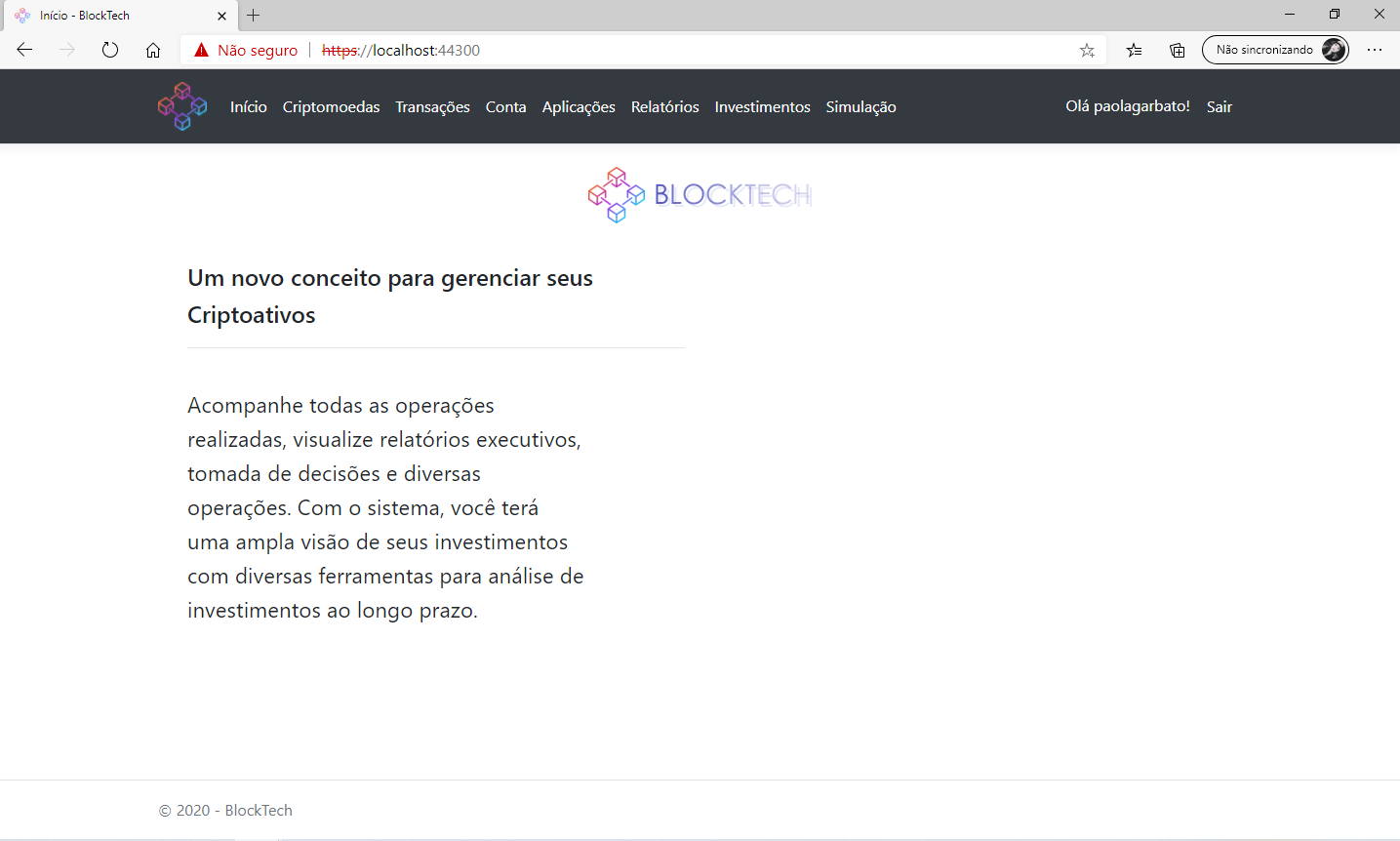
Figura 7 - Aplicativo Web - Lembrar o Login



Fonte: Paola Regina Garbato (2020)

4. Após realizar o login, você terá acesso à todas as funcionalidades do sistema, conforme pode visualizar na Figura 8.

Figura 8 - Aplicativo Web - Funcionalidades do Sistema



Fonte: Paola Regina Garbato (2020)

**Criptomoedas**

1. Para visualizar as criptomoedas e seus valores atuais, clique em “Criptomoedas” no menu, conforme a Figura 9.

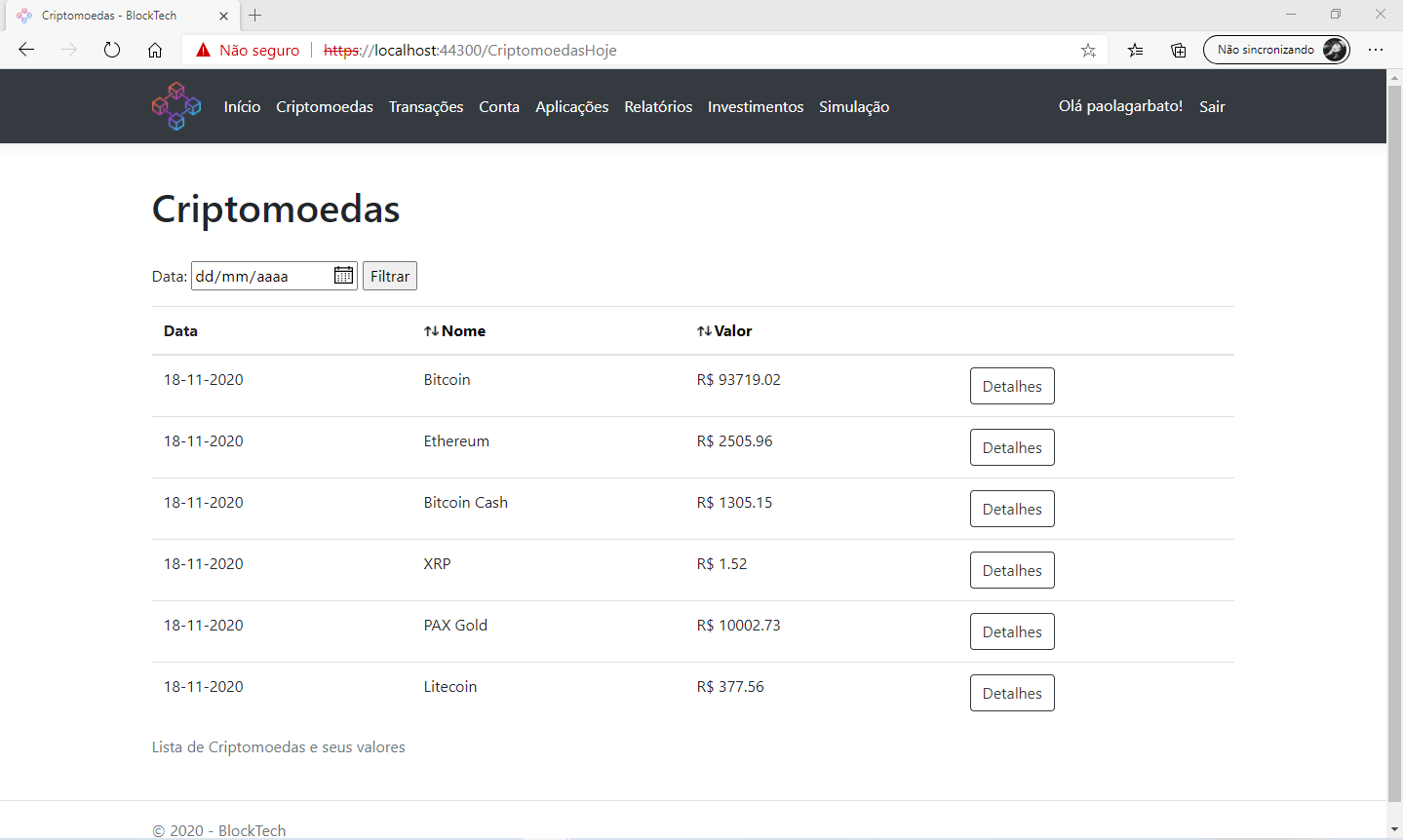
Figura 9 - Aplicativo Web - Criptomoedas



Fonte: Paola Regina Garbato (2020)

2. Todas as criptomoedas serão listadas, exibindo seus valores atuais, como mostra a Figura 10.

Figura 10 - Aplicativo Web - Valores Criptomoedas -



Fonte: Paola Regina Garbato (2020)

3. Para visualizar o valor das criptomoedas de algum dia específico, insira a data no campo “Data”, como mostra o exemplo da Figura 11.

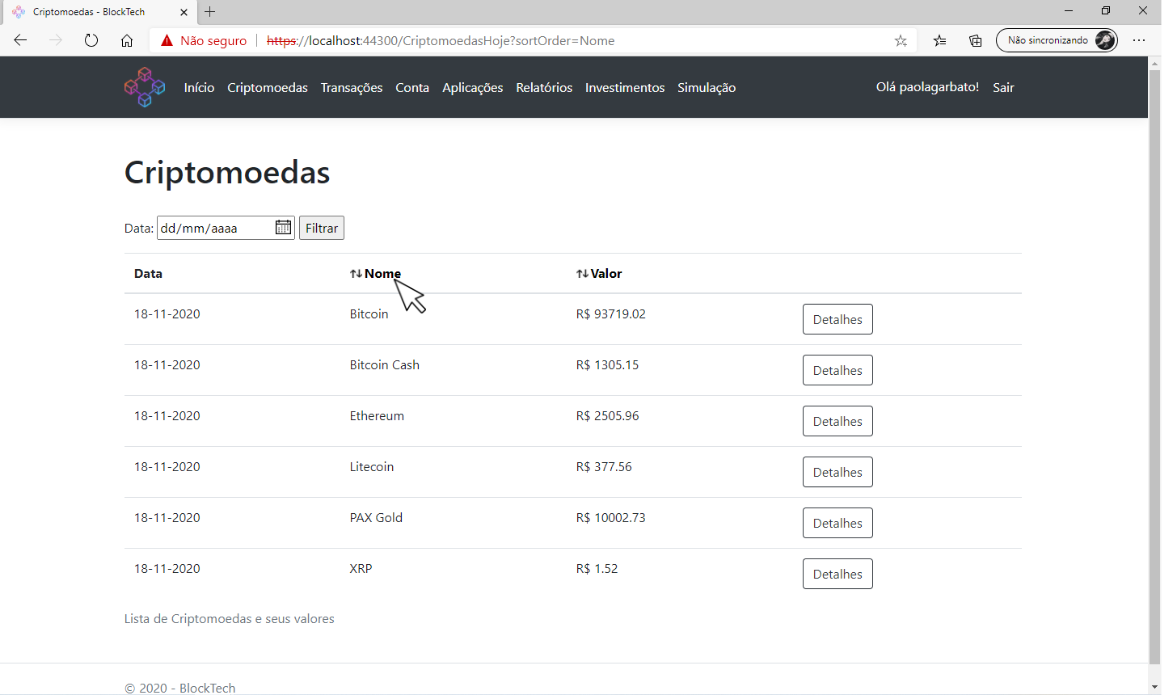
Figura 11 - Aplicativo Web - Pesquisa por Data - Criptomoedas



Fonte: Paola Regina Garbato (2020)

4. É possível ordenar as criptomoedas por nome. Para isso, basta clicar em “Nome”, como mostra a Figura 12.

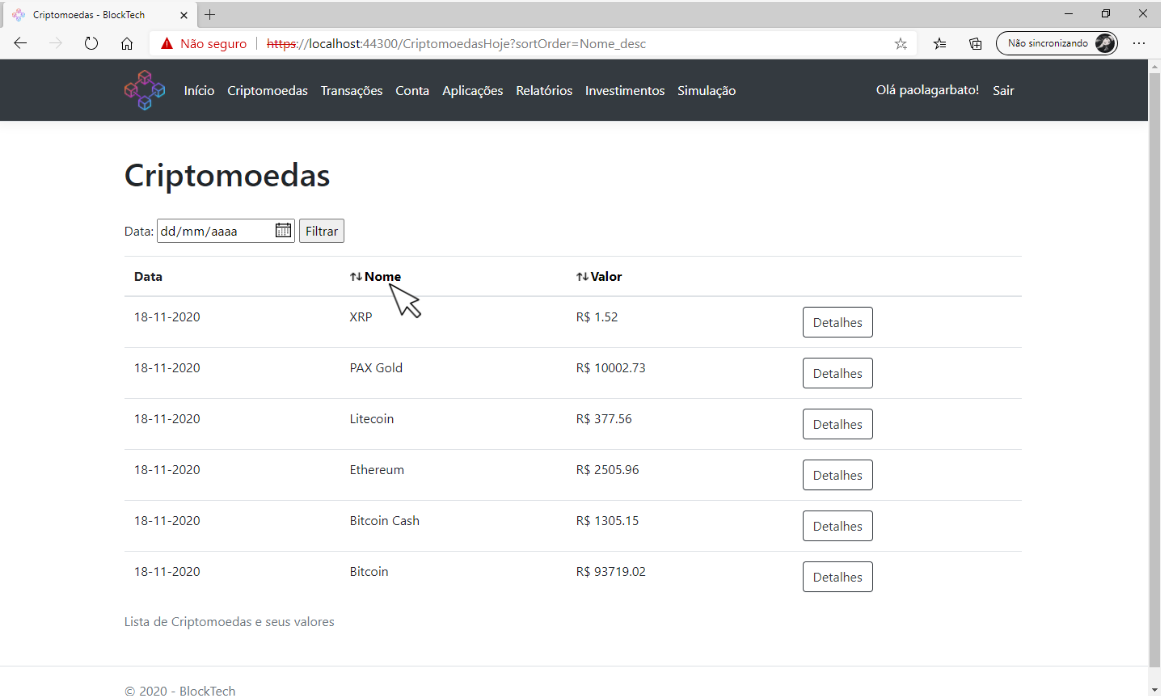
Figura 12 - Aplicativo Web - Ordenando por Nome - Criptomoedas



Fonte: Paola Regina Garbato (2020)

5. Para que as criptomoedas sejam listadas em ordem decrescente, basta clicar novamente sob seu nome, como mostra a Figura 13.

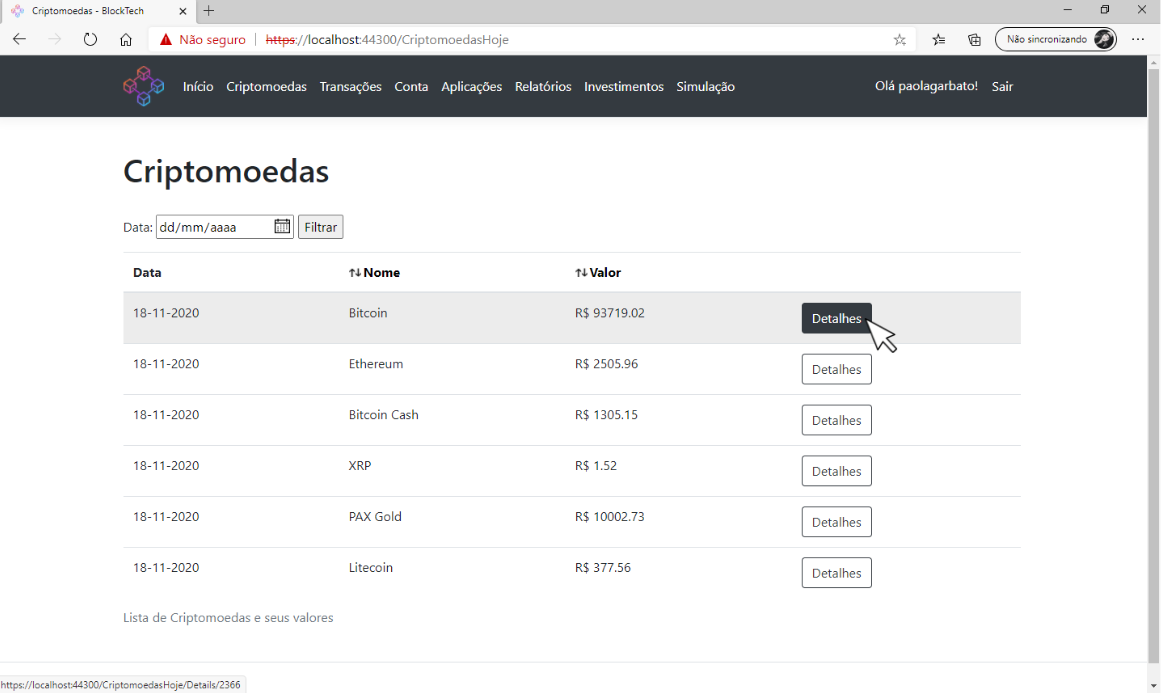
Figura 13 - Aplicativo Web - Criptomoedas - Ordem Decrescente



Fonte: Paola Regina Garbato (2020)

6. Para visualizar as criptomoedas de forma detalhada, clique em “Detalhes”, conforme a Figura 14.

Figura 14 - Aplicativo Web - Criptomoedas - Detalhes



Fonte: Paola Regina Garbato (2020)

7. Após clicar em “Detalhes”, poderá visualizar também o símbolo da Criptomoeda selecionada, conforme a Figura 15.

Figura 15 - Aplicativo Web - Criptomoeda – Visualizando Detalhes



Fonte: Paola Regina Garbato (2020)

**Transações**

1. Para visualizar suas transações, clique em “Transações”, conforme a Figura 16.

Figura 16 - Aplicativo Web - Transações



Fonte: Paola Regina Garbato (2020)

2. Em transações, são exibidas as seguintes informações: a data em que foi realizada, o tipo de transação, o valor, o nome da criptomoeda e quantidade da mesma, como mostra a Figura 17.

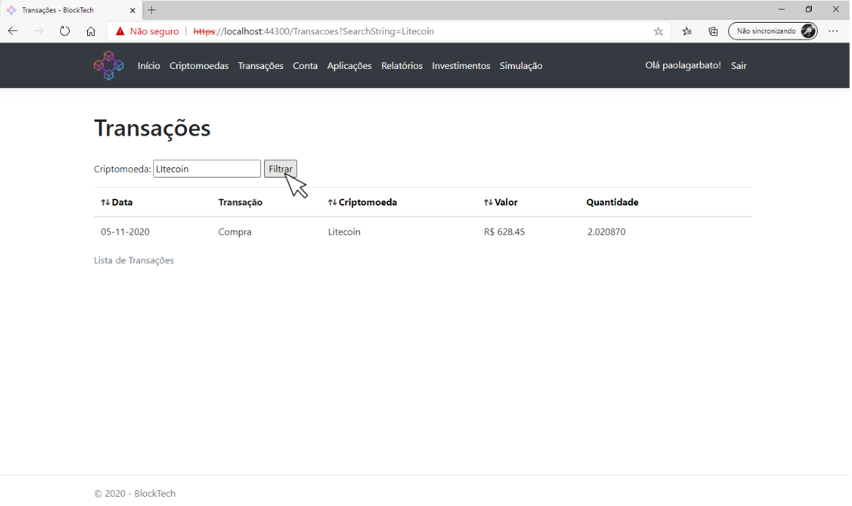
Figura 17 - Aplicativo Web - Lista de Transações



Fonte: Paola Regina Garbato (2020)

3. Para visualizar as transações de uma criptomoeda específica, insira seu nome no campo indicado na Figura 18, e clique em “Filtrar”. Também é possível ordenar as transações por data, valor e nome de criptomoeda, basta clicar sob seus nomes, como o exemplo em Criptomoedas.

Figura 18 - Aplicativo Web - Transações - Busca por nome



Fonte: Paola Regina Garbato (2020)

**Conta**

1. Para visualizar as informações da sua conta, selecione “Conta”, conforme a Figura 19.

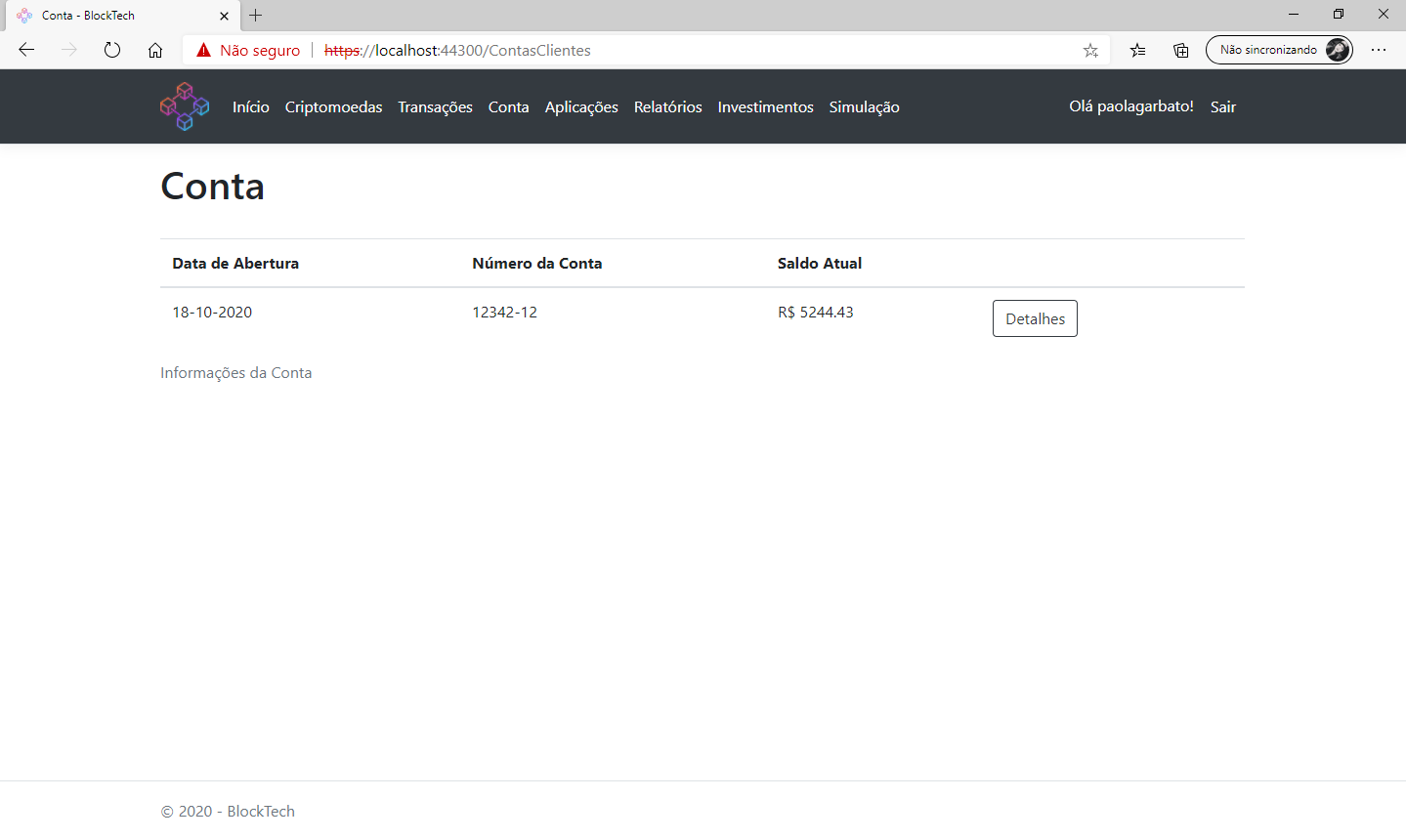
Figura 19 - Aplicativo Web - Conta



Fonte: Paola Regina Garbato (2020)

2. Em Conta, é possível visualizar o número da sua conta, a data de abertura, e todo o saldo que possui – de acordo com o valor atual das criptomoedas que possui investimento, como é possível visualizar na Figura 20.

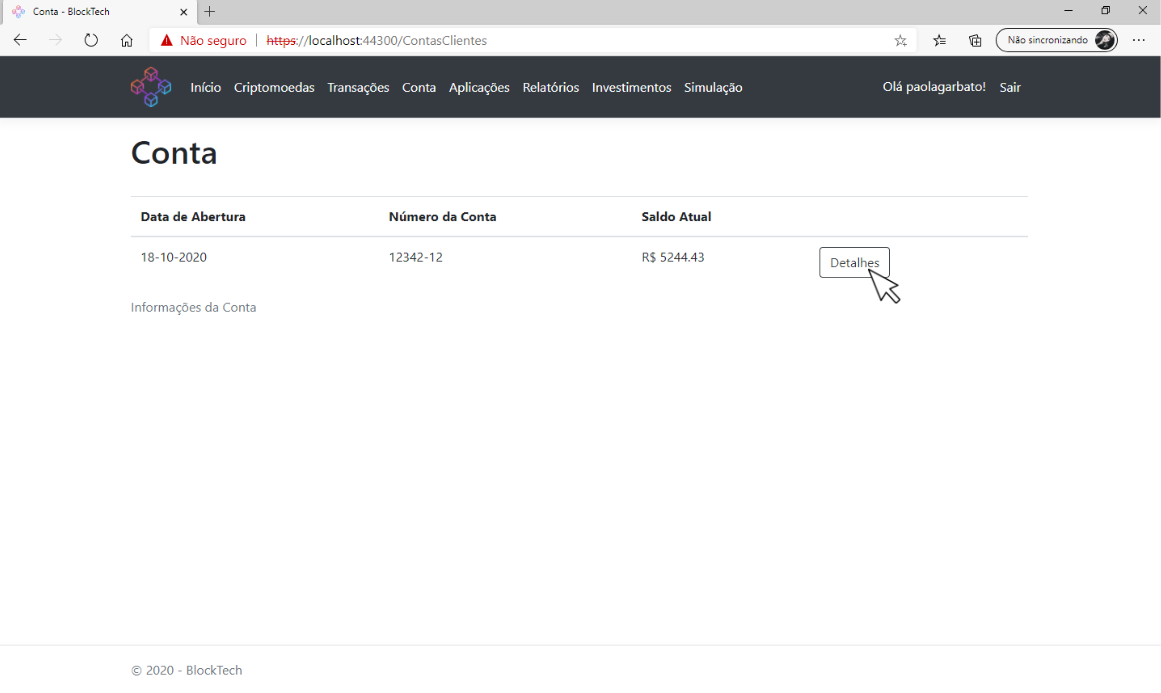
Figura 20 - Aplicativo Web - Conta



Fonte: Paola Regina Garbato (2020)

3. Para visualizar mais detalhes, como a sua conta vinculada, clique em “Detalhes”, como mostra a Figura 21.

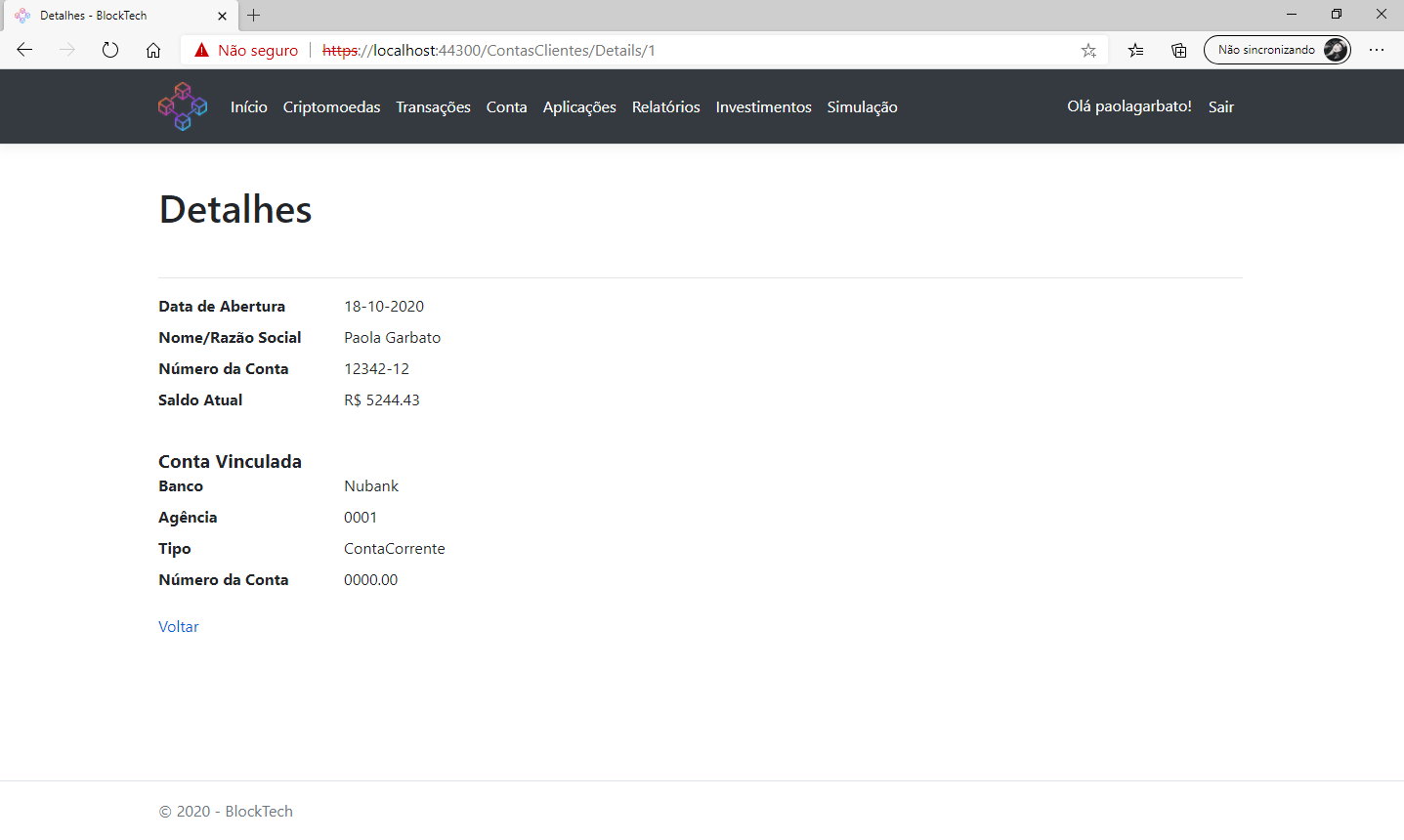
Figura 21 - Aplicativo Web – Conta - Detalhes



Fonte: Paola Regina Garbato (2020)

4. Após clicar em “Detalhes”, você terá acesso as informações da sua conta, e também a de sua conta vinculada, conforme o exemplo da Figura 22.

Figura 22 - Aplicativo Web - Detalhes da Conta



Fonte: Paola Regina Garbato (2020)

**Aplicações**

1. Para visualizar suas aplicações, clique em “Aplicações”, conforme a Figura 23.

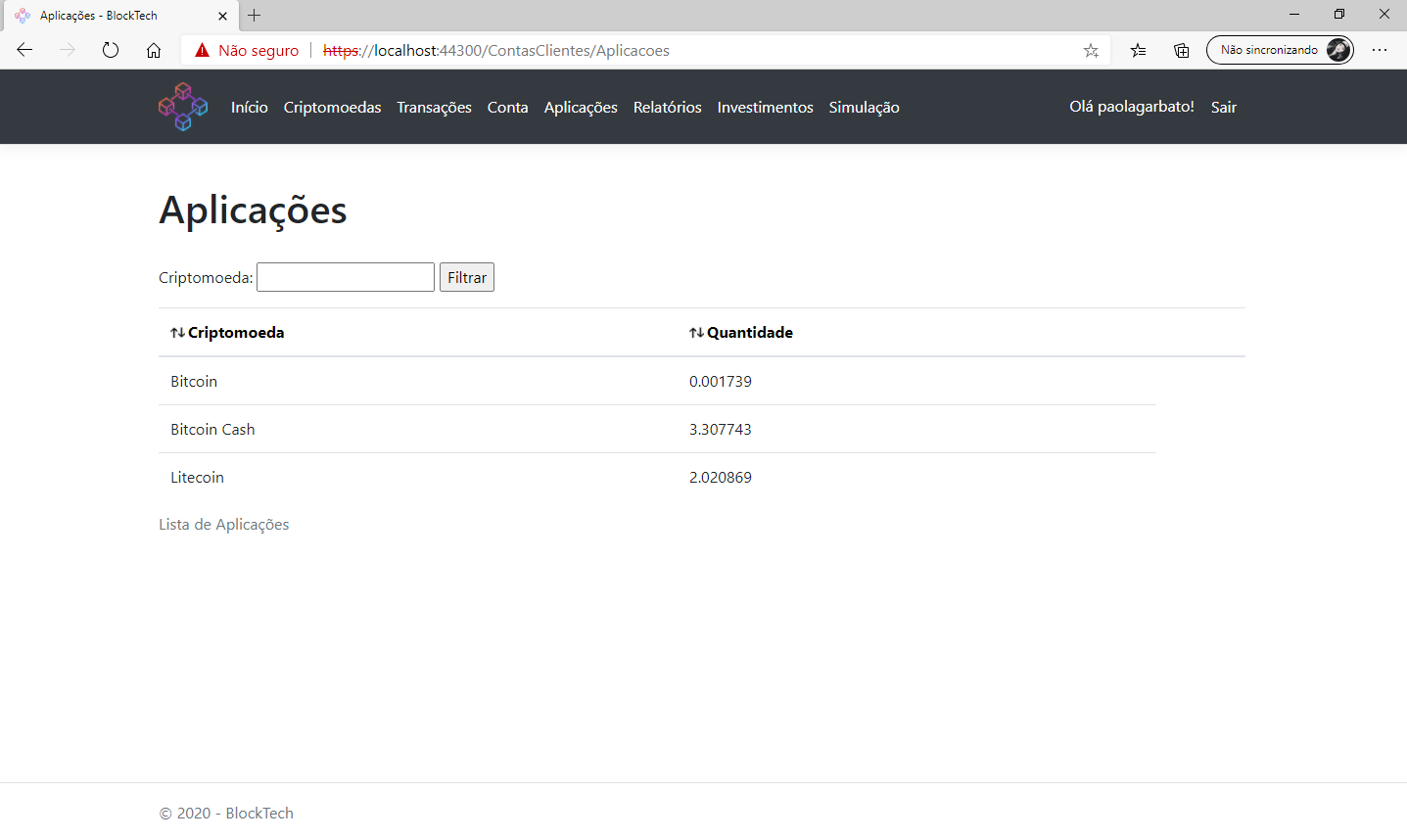
Figura 23 - Aplicativo Web - Aplicações



Fonte: Paola Regina Garbato (2020)

2. Em Aplicações, poderá visualizar o nome das Criptomoedas que investiu, e suas respectivas quantidades, conforme a Figura 24.

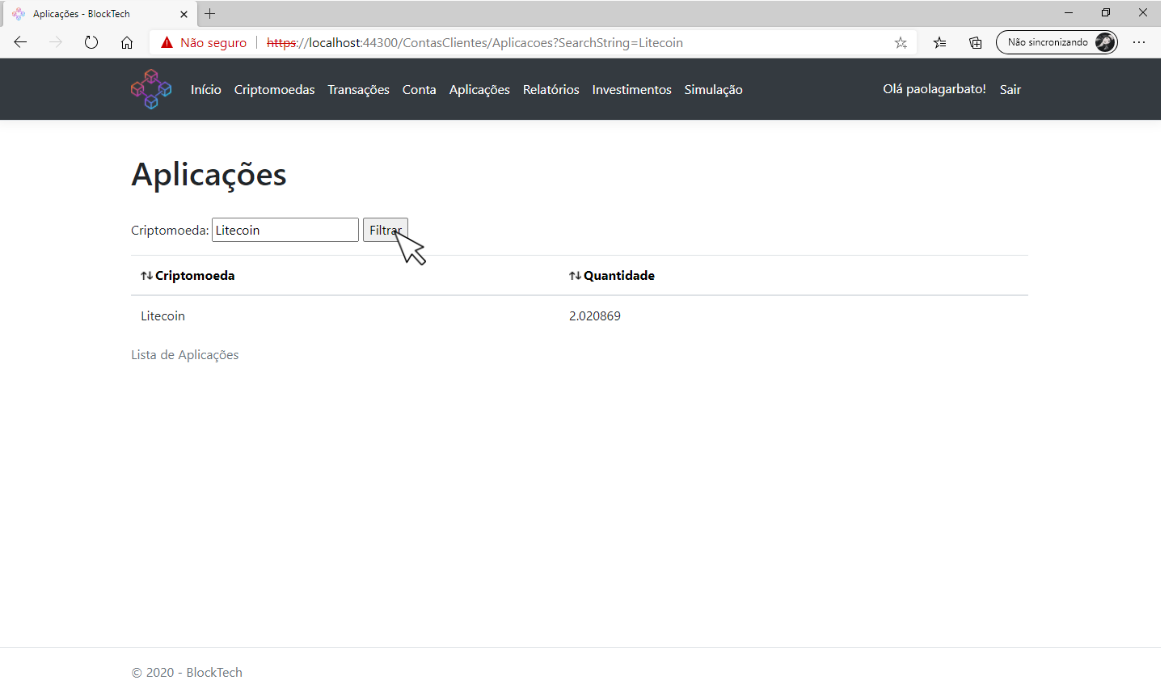
Figura 24 - Aplicativo Web - Listagem de Aplicações



Fonte: Paola Regina Garbato (2020)

2. Para visualizar uma criptomoeda específica, insira seu nome no campo “Criptomoeda”, e clique em “Filtrar”, conforme a Figura 25.

Figura 25 - Aplicativo Web - Aplicações – Filtrando Criptomoedas por Nome



Fonte: Paola Regina Garbato (2020)

**Relatórios**

1. Para visualizar relatórios das criptomoedas, clique em “Relatórios”, conforme a Figura 26.

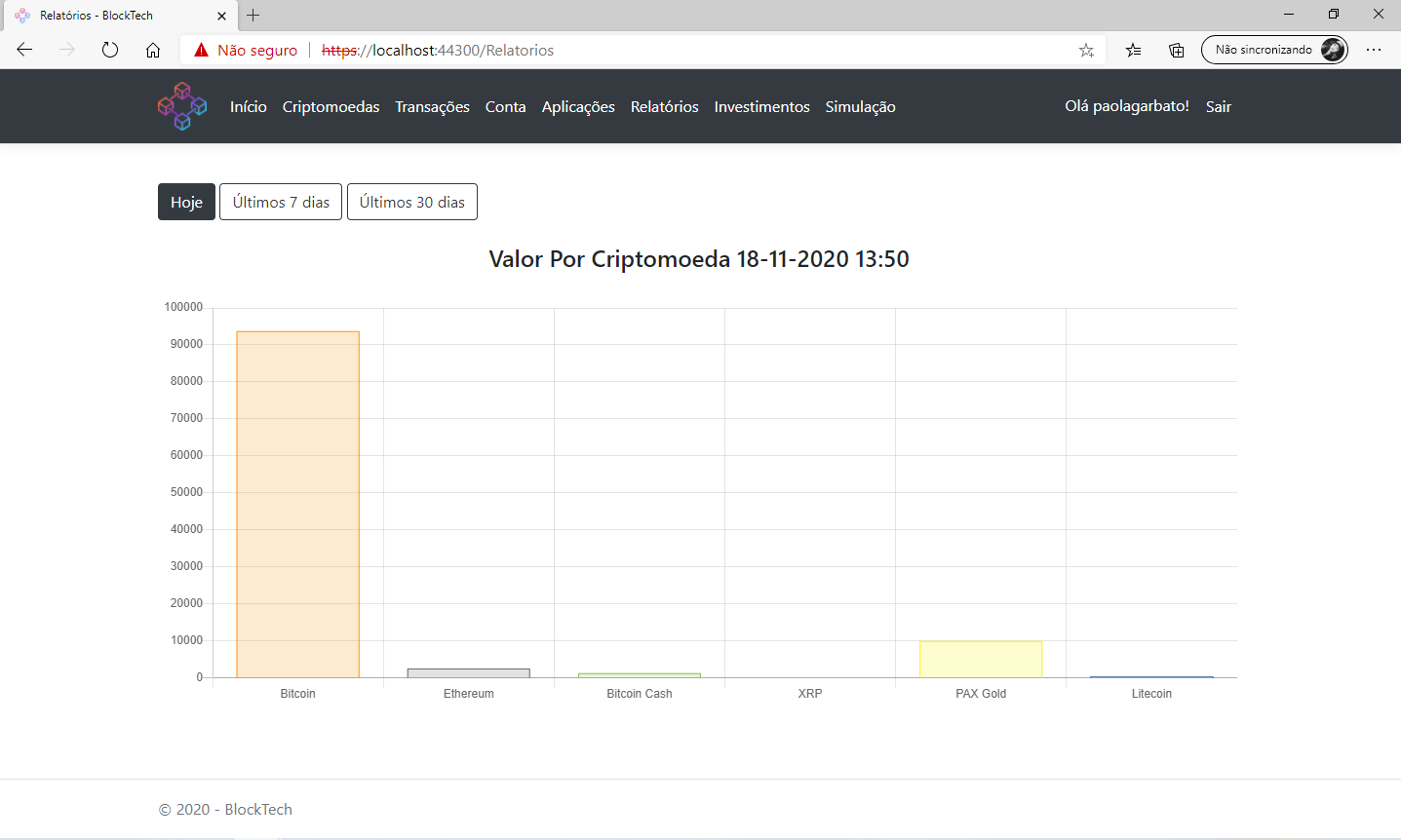
Figura 26 - Aplicativo Web - Relatórios



Fonte: Paola Regina Garbato (2020)

2. Após acessar os relatórios, irá visualizar um gráfico com o valor atual de todas as criptomoedas em que pode investir na BlockTech, conforme a Figura 27.

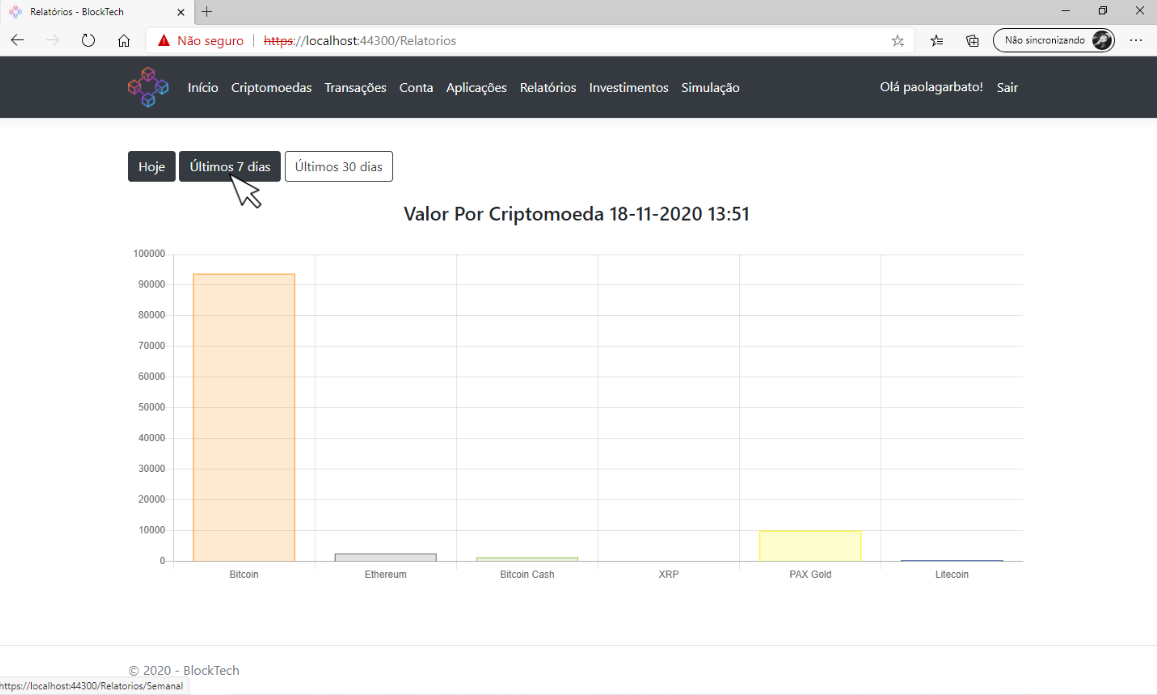
Figura 27 - Aplicativo Web – Relatório Diário



Fonte: Paola Regina Garbato (2020)

3. Para visualizar as criptomoedas nos últimos 7 dias, clique em “Últimos 7 dias”, conforme a Figura 28.

Figura 28 - Aplicativo Web - Acessando últimos 7 dias



Fonte: Paola Regina Garbato (2020)

4. Após acessar “Últimos 7 dias”, poderá visualizar um gráfico com a porcentagem de crescimento/queda de todas as criptomoedas nesse período, conforme mostra a Figura 29.

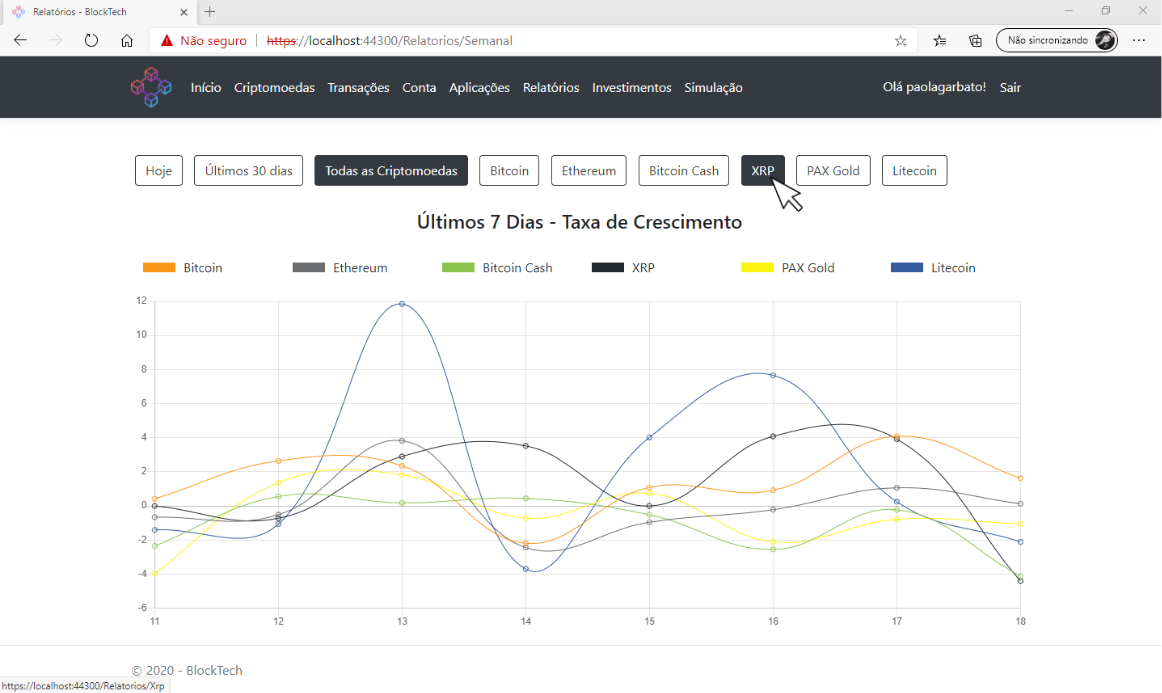
Figura 29 - Aplicativo Web - Relatórios - Últimos 7 dias



Fonte: Paola Regina Garbato (2020)

5. Para ver os valores de uma criptomoeda específica nos últimos 7 dias, clique em seu nome, conforme o exemplo da Figura 30.

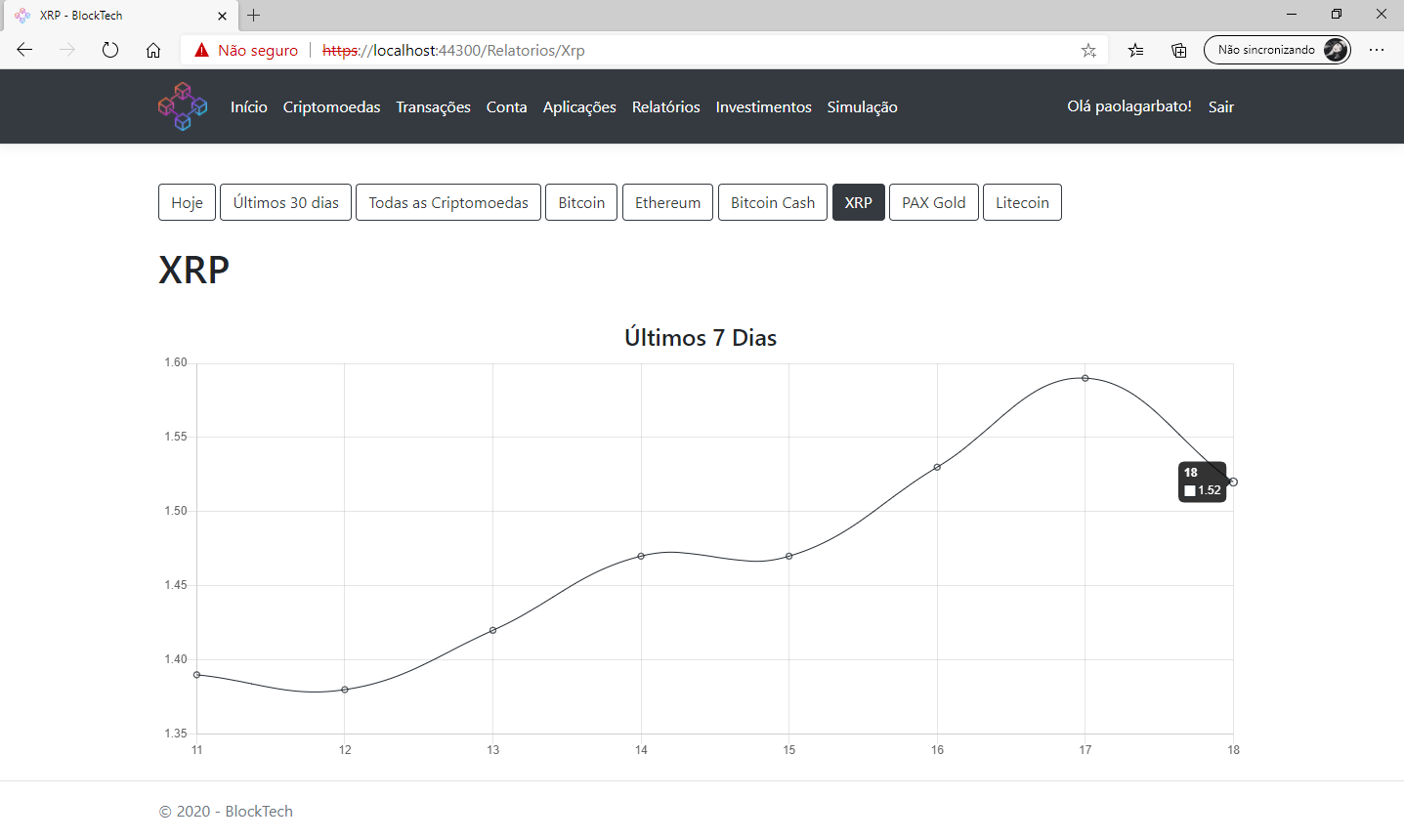
Figura 30 - Aplicativo Web - Relatórios – Visualizando Criptomoeda Específica



Fonte: Paola Regina Garbato (2020)

6. Após selecionar a criptomoeda desejada, poderá visualizar um gráfico com seu valor nos últimos 7 dias, conforme a Figura 31.

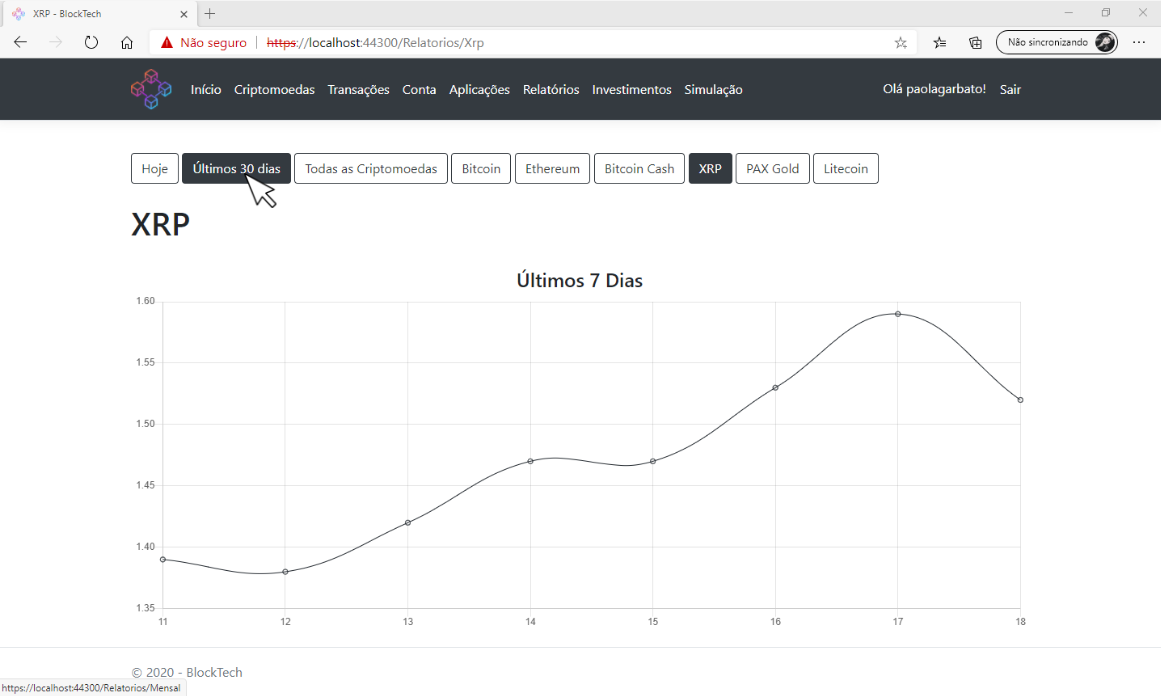
Figura 31 - Aplicativo Web - XRP - Últimos 7 Dias



Fonte: Paola Regina Garbato (2020)

7. Para visualizar a porcentagem de crescimento/queda de todas as criptomoedas nos últimos 30 dias, clique em “Últimos 30 dias”, conforme a Figura 32.

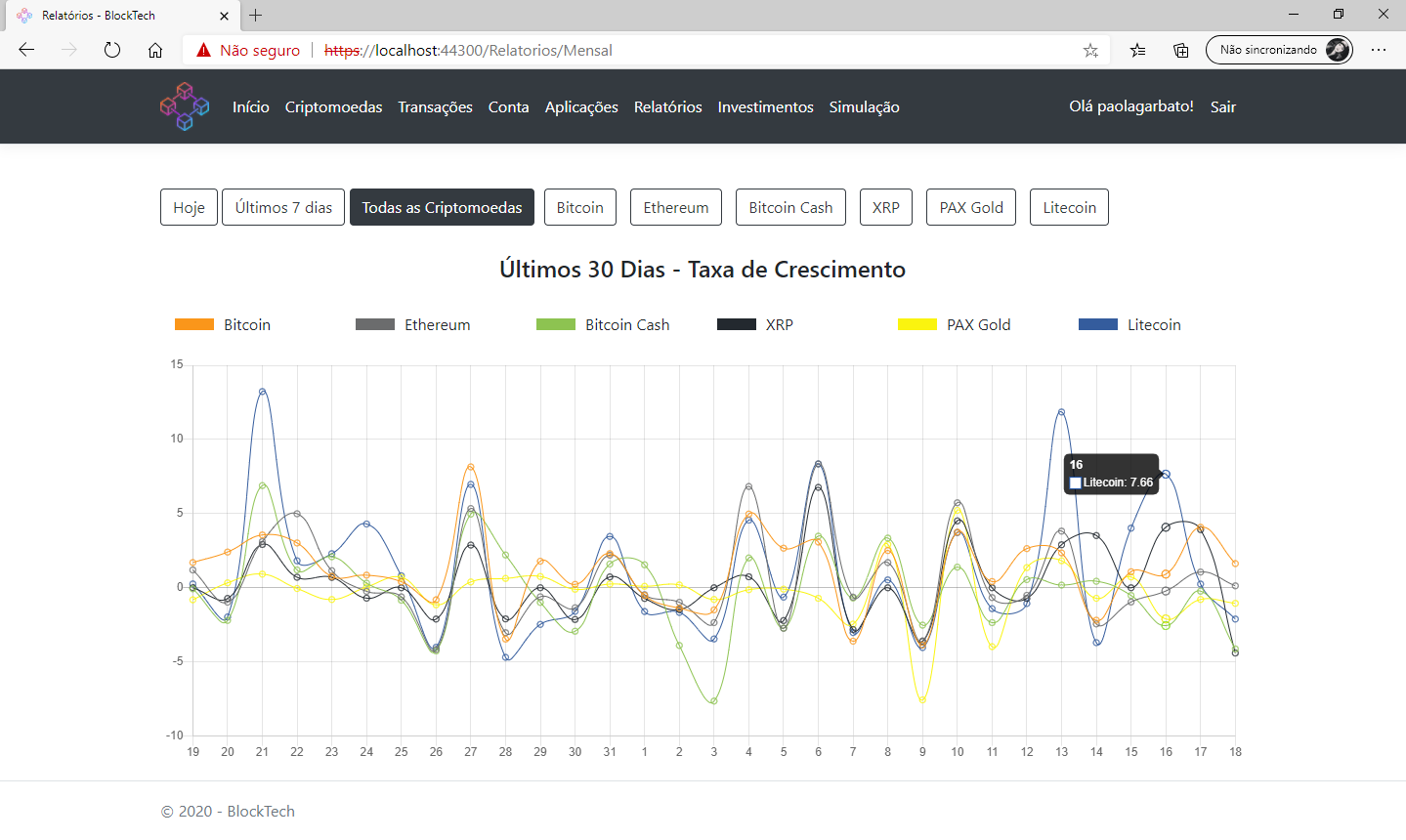
Figura 32 - Aplicativo Web - Relatórios - Últimos 30 Dias



Fonte: Paola Regina Garbato (2020)

8. Após selecionar “Últimos 30 dias”, poderá visualizar a porcentagem de crescimento/queda de todas as criptomoedas nesse período, conforme a Figura 33.

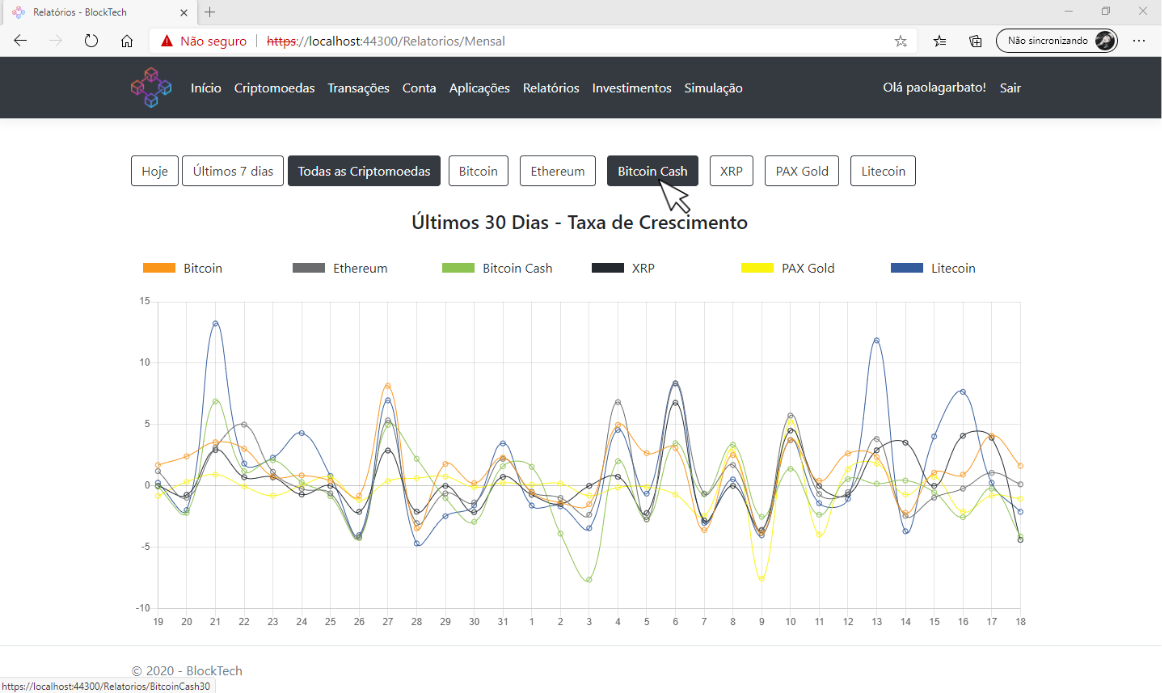
Figura 33 - Aplicativo Web – Relatórios - Últimos 30 Dias



Fonte: Paola Regina Garbato (2020)

9. Para ver os valores dos últimos 30 dias de uma criptomoeda específica, clique sob seu nome como mostra a Figura 34.

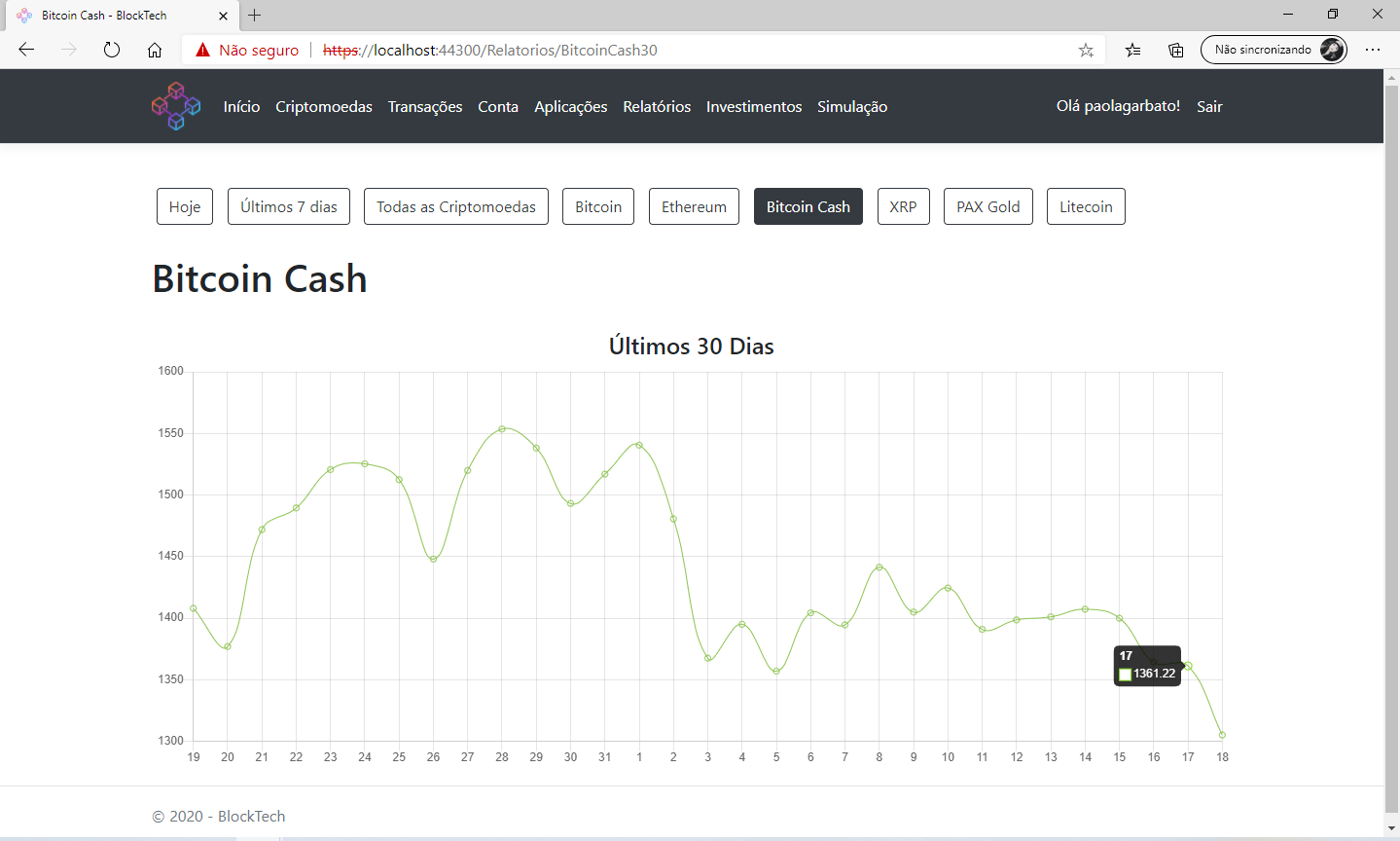
Figura 34 - Aplicativo Web - Relatórios - Acessando Bitcoin Cash



Fonte: Paola Regina Garbato (2020)

10. Após selecionar a criptomoeda desejada, poderá visualizar o seu valor nos últimos 30 dias, conforme a Figura 35.

Figura 35 - Aplicativo Web - Relatórios 30 Dias - Bitcoin Cash

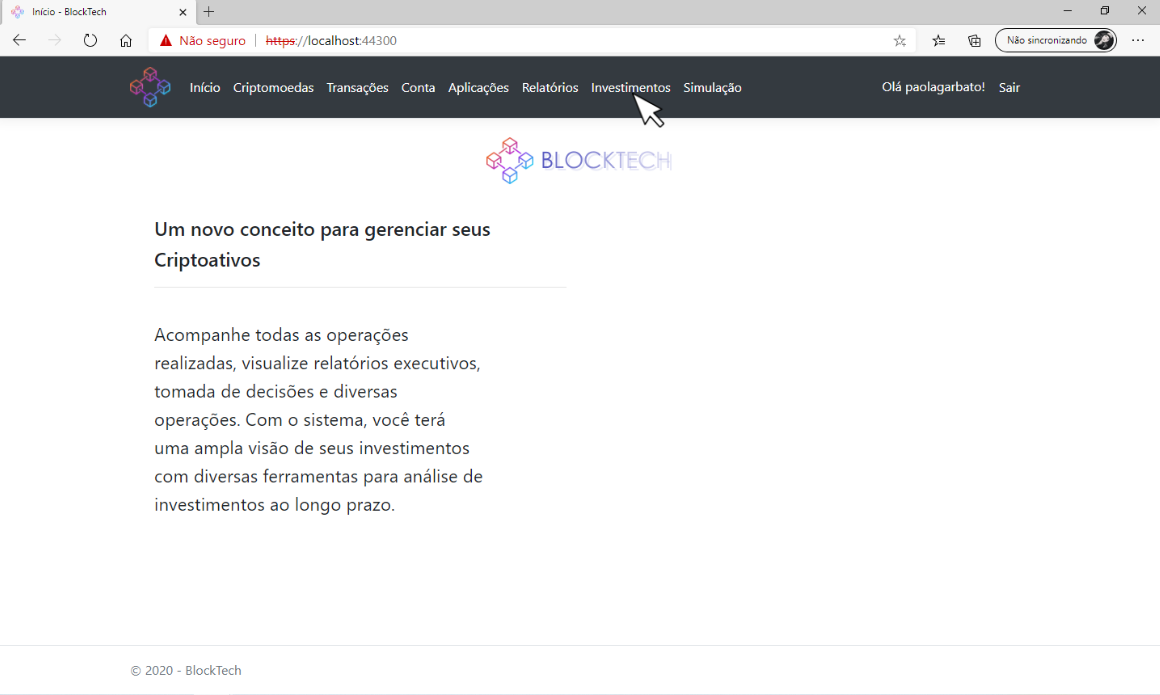


Fonte: Paola Regina Garbato (2020)

**Investimentos**

1. Para visualizar detalhes de seus investimentos, clique em “Investimentos”, conforme a Figura 36.

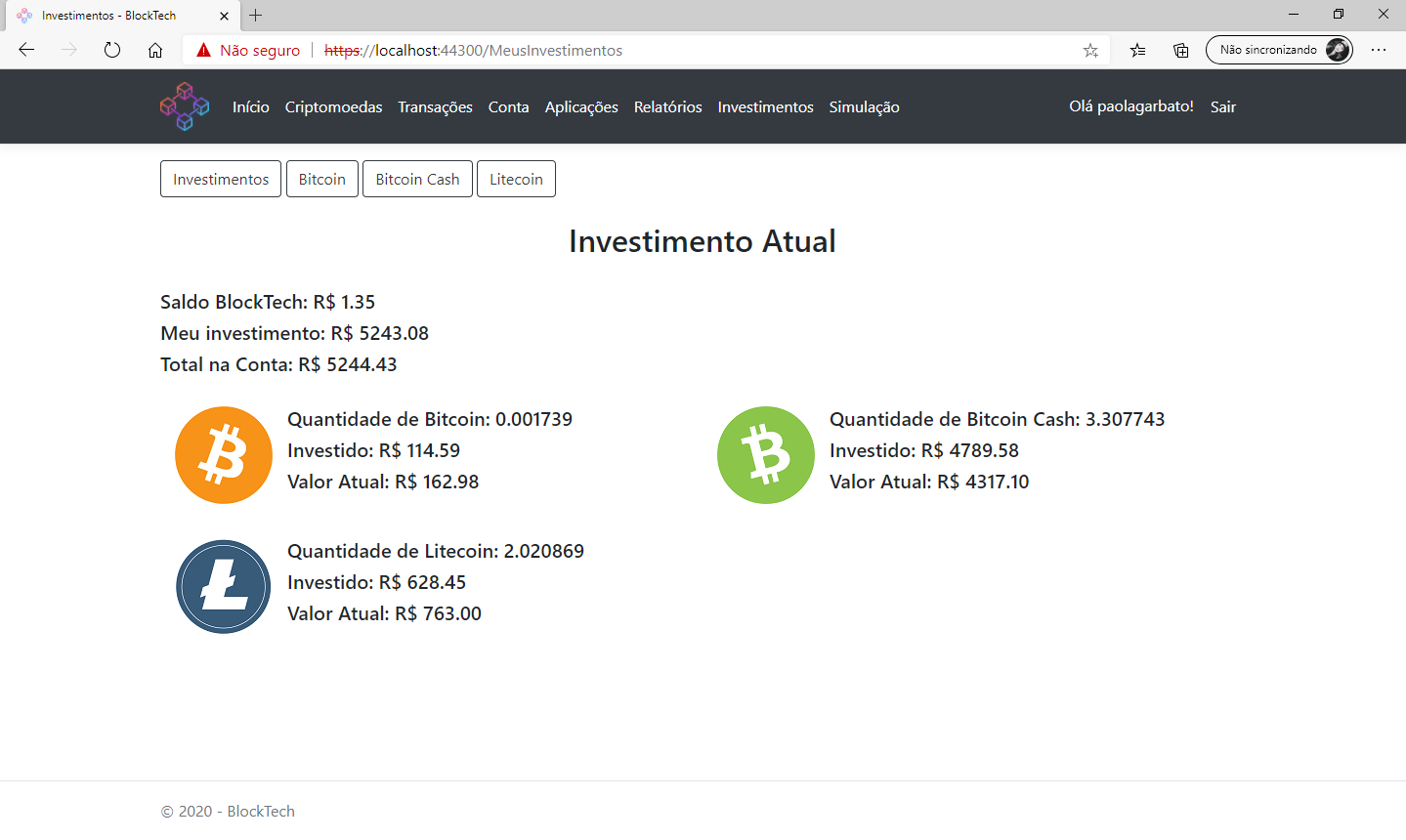
Figura 36 - Aplicativo Web - Investimentos



Fonte: Paola Regina Garbato (2020)

2. Após acessar Investimentos, serão exibidos o seu saldo na BlockTech, o total que investiu, e o saldo atual em sua conta. Além disso, verá as criptomoedas nas quais possui investimento – o quanto possui em quantidade e saldo atual nelas, e também o quanto foi investido em cada uma, conforme a Figura 37.

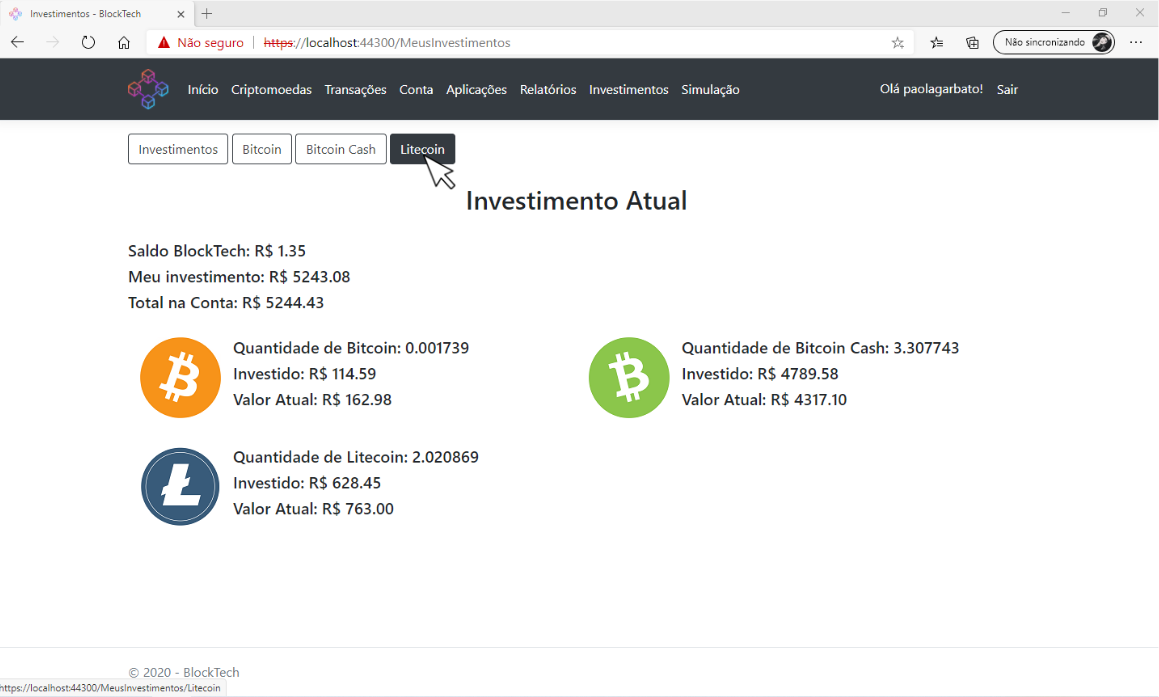
Figura 37 - Aplicativo Web - Visualizando Investimentos



Fonte: Paola Regina Garbato (2020)

3. Para visualizar seu saldo dos últimos 7 dias de uma criptomoeda específica, clique sob seu nome, conforme o exemplo da Figura 38.

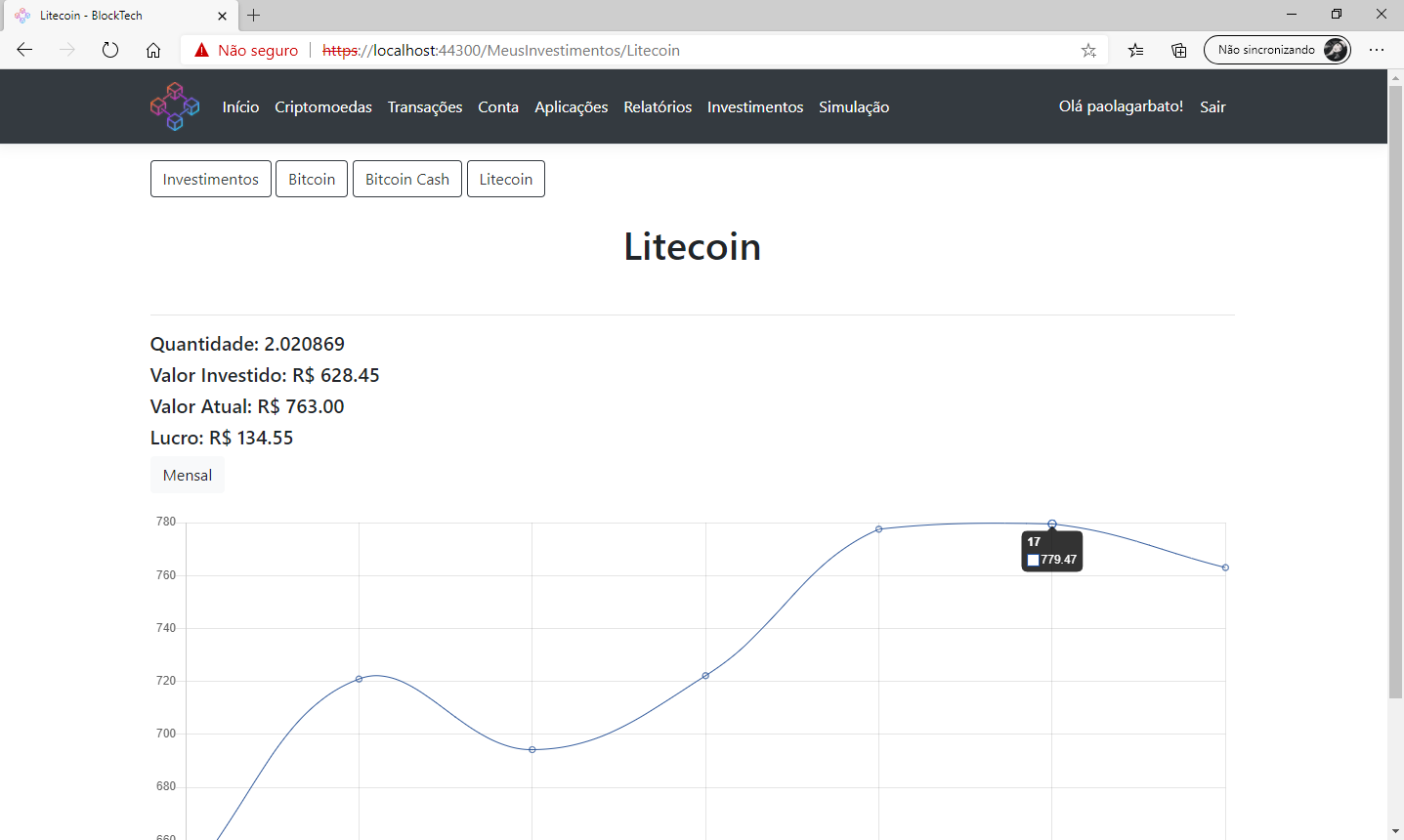
Figura 38 - Aplicativo Web - Investimento - Acessando Litecoin



Fonte: Paola Regina Garbato (2020)

4. Após acessar a criptomoeda desejada, poderá visualizar um gráfico dos últimos 7 dias de seu investimento, além de sua quantidade, valor investido e valor atual. Além disso, também será exibido o seu lucro ou perda após investir na criptomoeda, conforme a Figura 39.

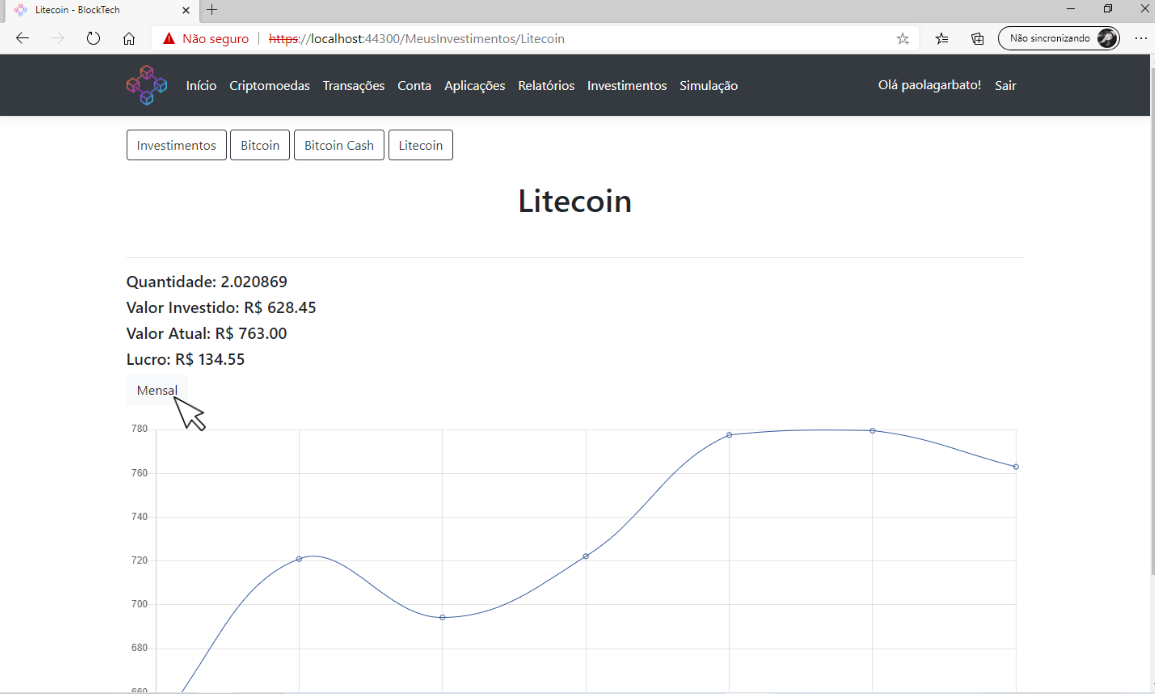
Figura 39 - Aplicativo Web - Investimentos - Litecoin



Fonte: Paola Regina Garbato (2020)

5. Para visualizar um gráfico dos últimos 30 dias, clique em “Mensal”, conforme a Figura 40.

Figura 40 - Aplicativo Web - Investimento – Acessando Gráfico Mensal



Fonte: Paola Regina Garbato (2020)

6. Caso tenha iniciado seu investimento há menos de 30 dias, será exibido o gráfico desde o dia em que investiu, conforme a Figura 41.

Figura 41 - Aplicativo Web - Investimentos - Gráfico Mensal



Fonte: Paola Regina Garbato (2020)

**Simulação**

1. Para realizar uma simulação de compra de criptomoedas, clique em “Simulação”, conforme a Figura 42.

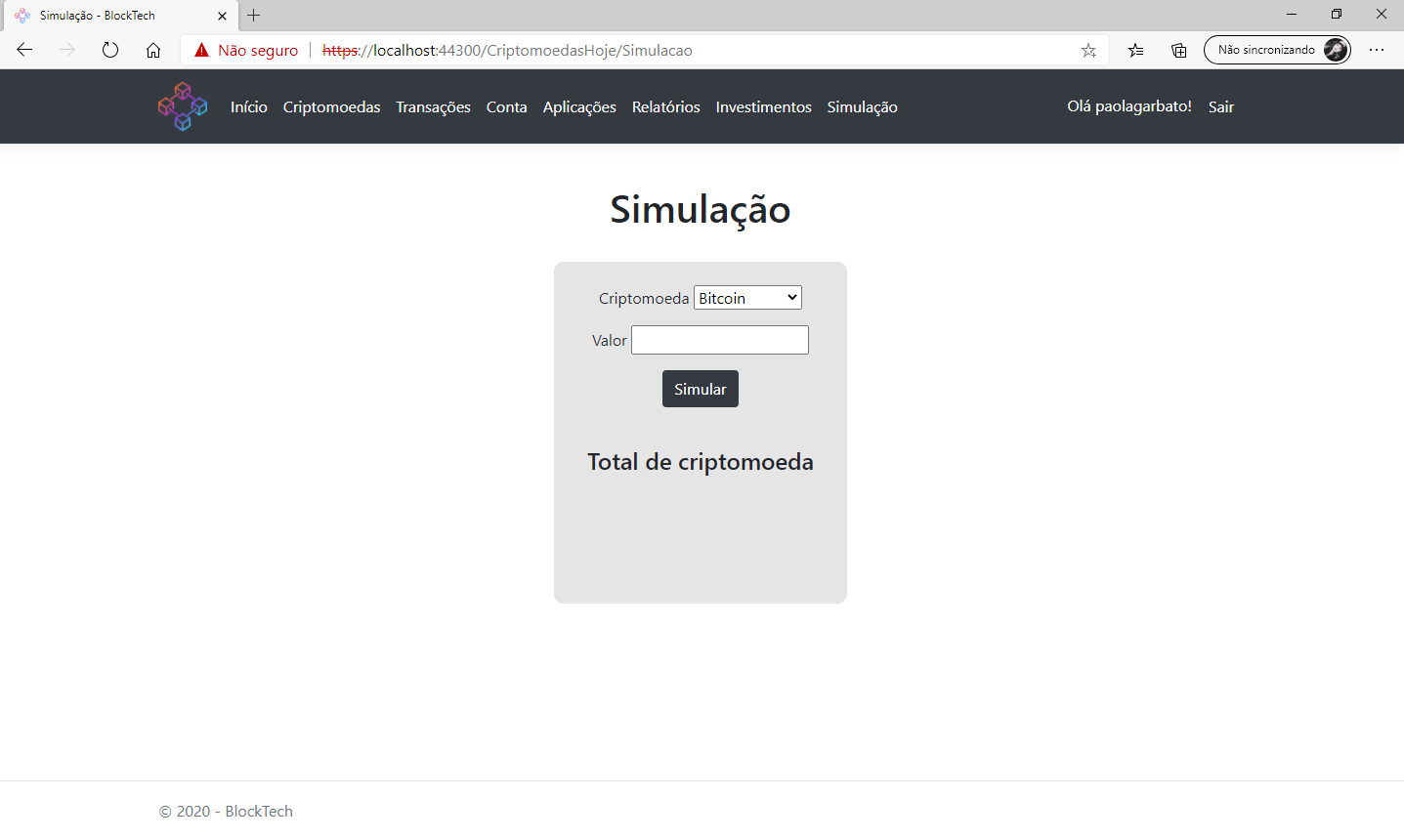
Figura 42 - Aplicativo Web - Simulação



Fonte: Paola Regina Garbato (2020)

2. Em simulação, como pode ser visto na Figura 43, poderá selecionar a criptomoeda desejada, o valor em reais, e visualizar a quantidade de criptomoeda que seria adquirida.

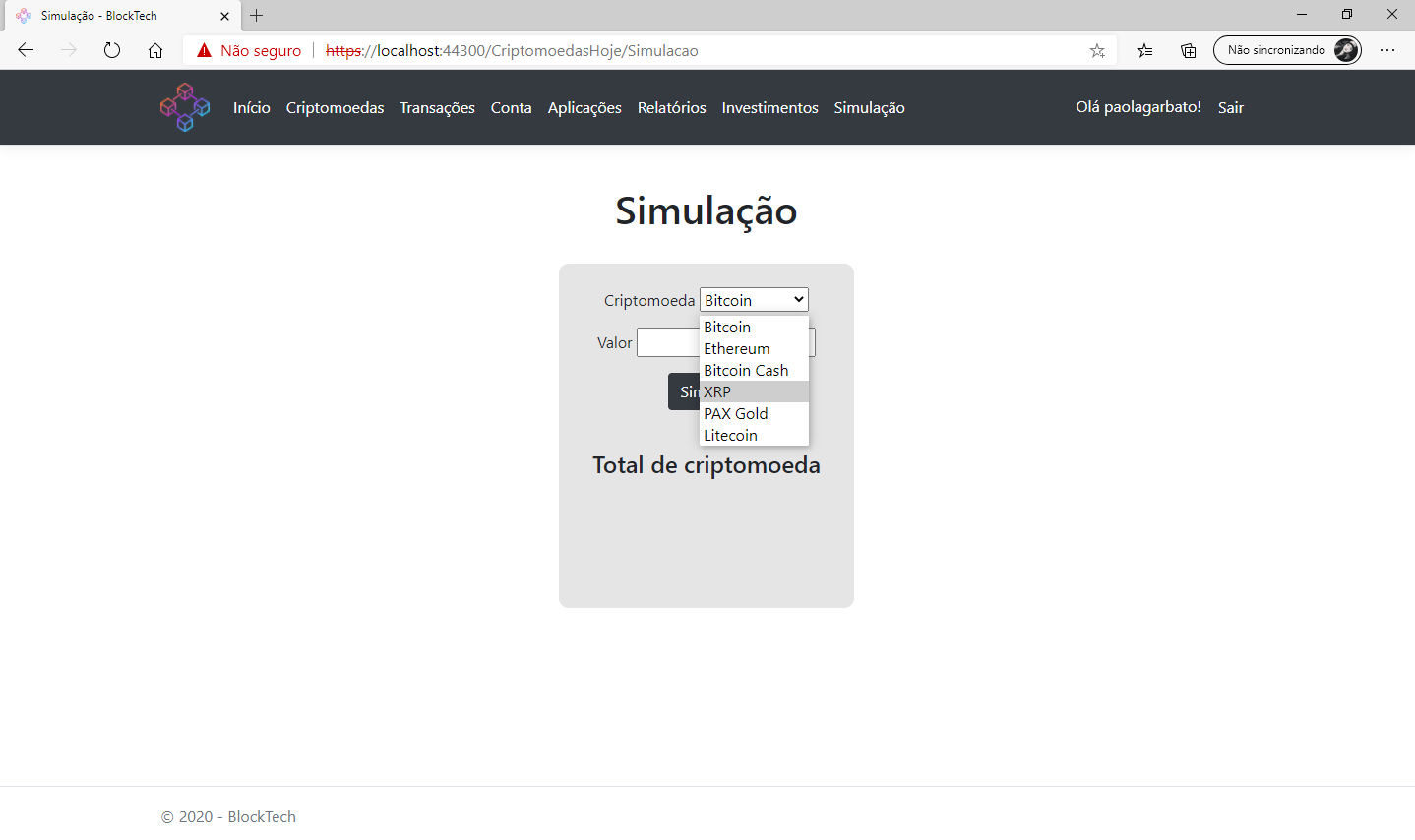
Figura 43 - Aplicativo Web - Visualizando Página de Simulação



Fonte: Paola Regina Garbato (2020)

3. Primeiro, selecione a criptomoeda desejada, conforme a Figura 44.

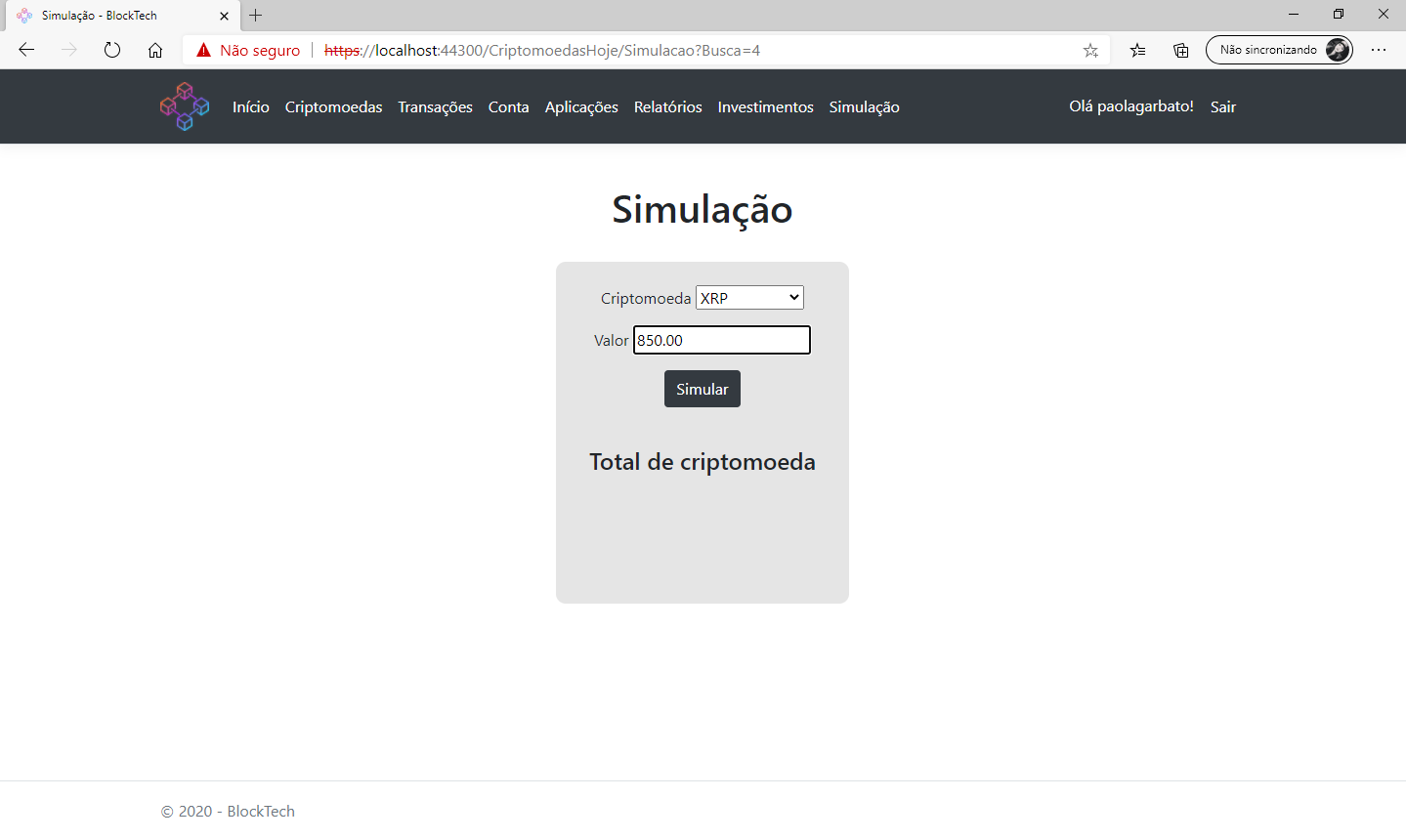
Figura 44 - Aplicativo Web - Simulação – Selecionando Criptomoeda



Fonte: Paola Regina Garbato (2020)

4. Insira o valor que deseja investir, conforme a Figura 45.

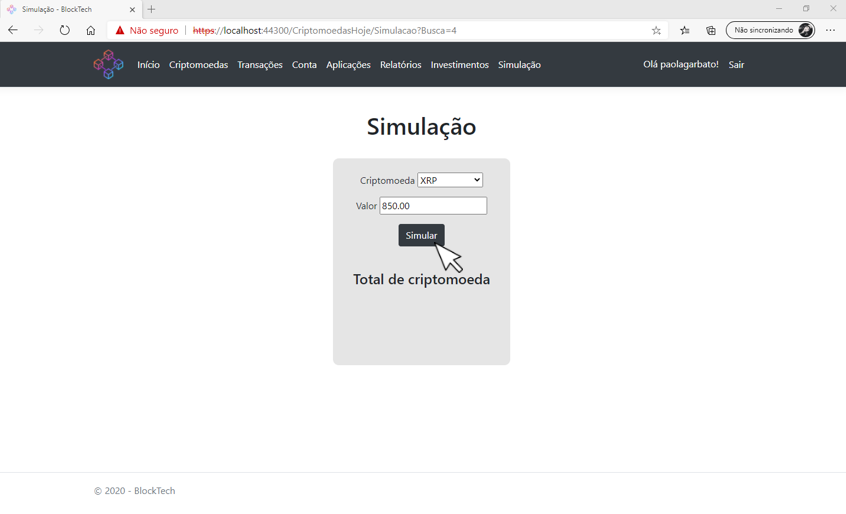
Figura 45 - Aplicativo Web – Simulação - Inserindo Valor



Fonte: Paola Regina Garbato (2020)

5. Após inserir os dados, clique em “Simular”, conforme a Figura 46.

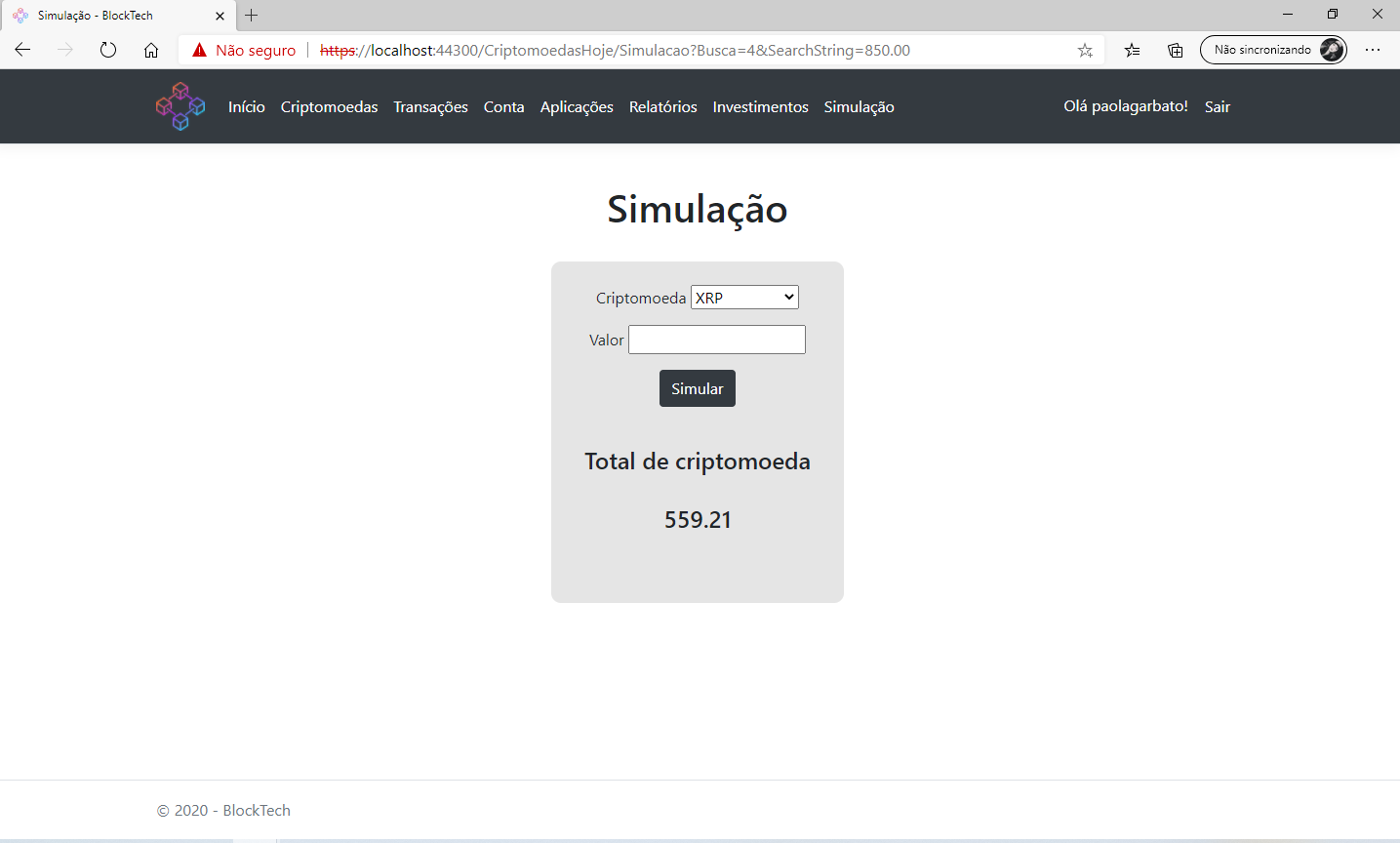
Figura 46 - Aplicativo Web – Simulando



Fonte: Paola Regina Garbato (2020)

6. Então, será exibido o total de criptomoeda que obterá com o valor investido, conforme a Figura 47.

Figura 47 - Aplicativo Web - Resultado da Simulação



Fonte: Paola Regina Garbato (2020)

**Alterar dados**

1. Para alterar seu telefone ou senha, clique em seu nome de usuário, conforme a Figura 48.

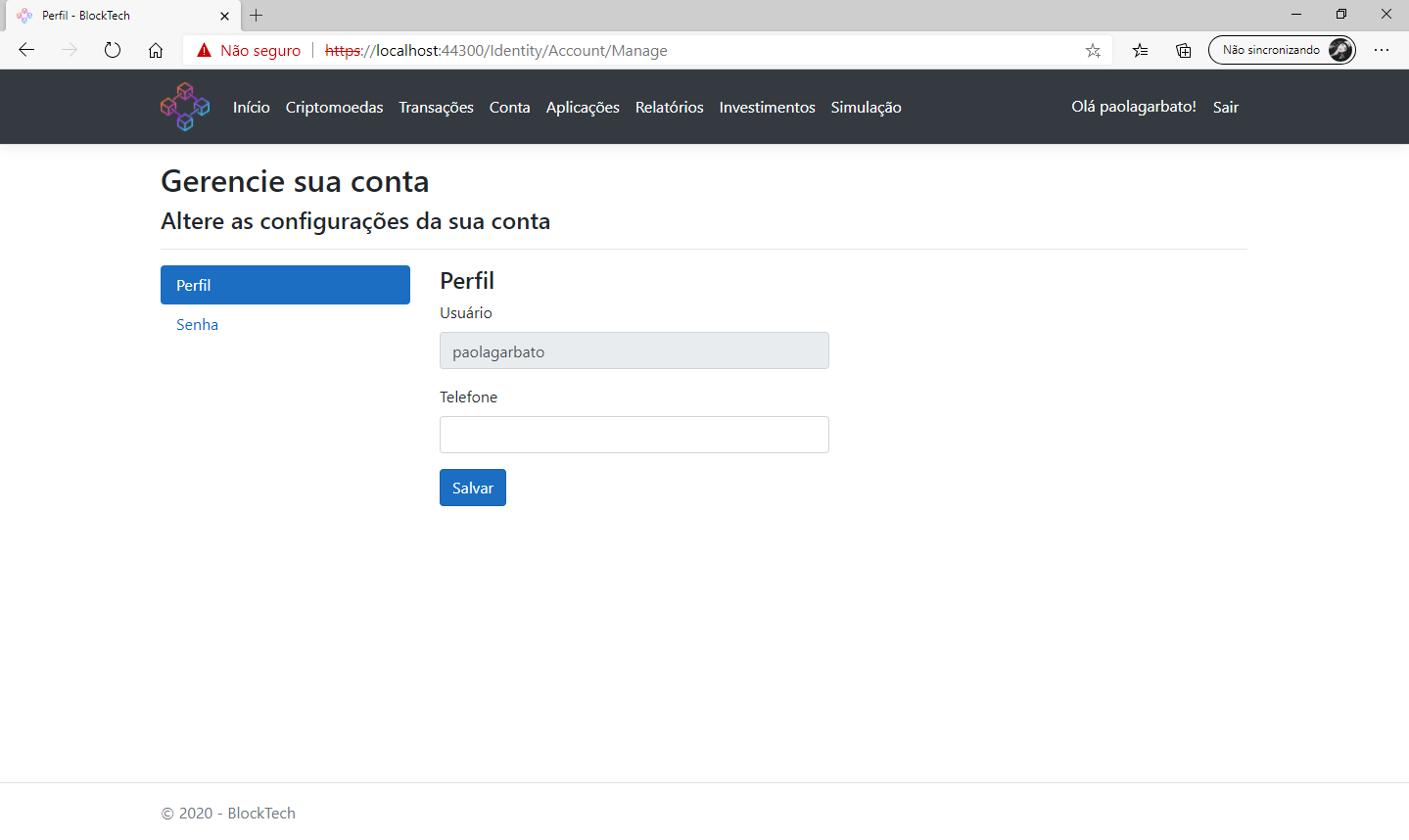
Figura 48 - Aplicativo Web - Alterando Dados



Fonte: Paola Regina Garbato (2020)

2. Será aberto o gerenciamento de sua conta, conforme a Figura 49.

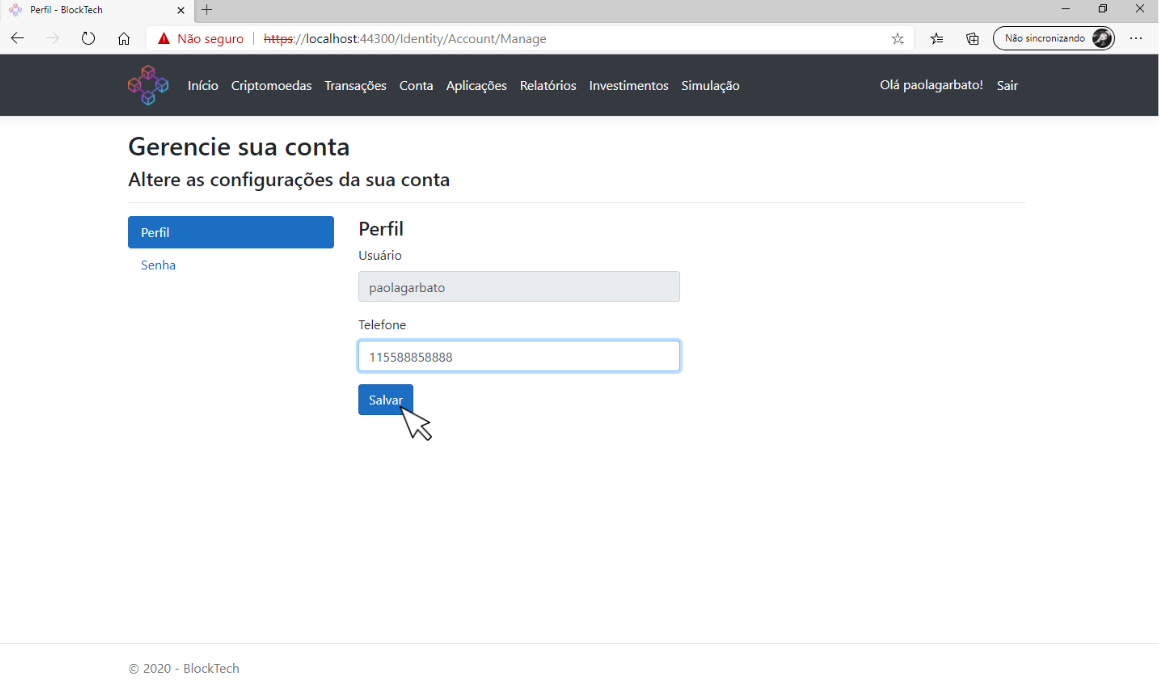
Figura 49 - Aplicativo Web - Gerenciamento de Dados



Fonte: Paola Regina Garbato (2020)

3. Para alterar seu telefone, insira o novo no campo “Telefone”, e clique em “Salvar”, conforme a Figura 50.

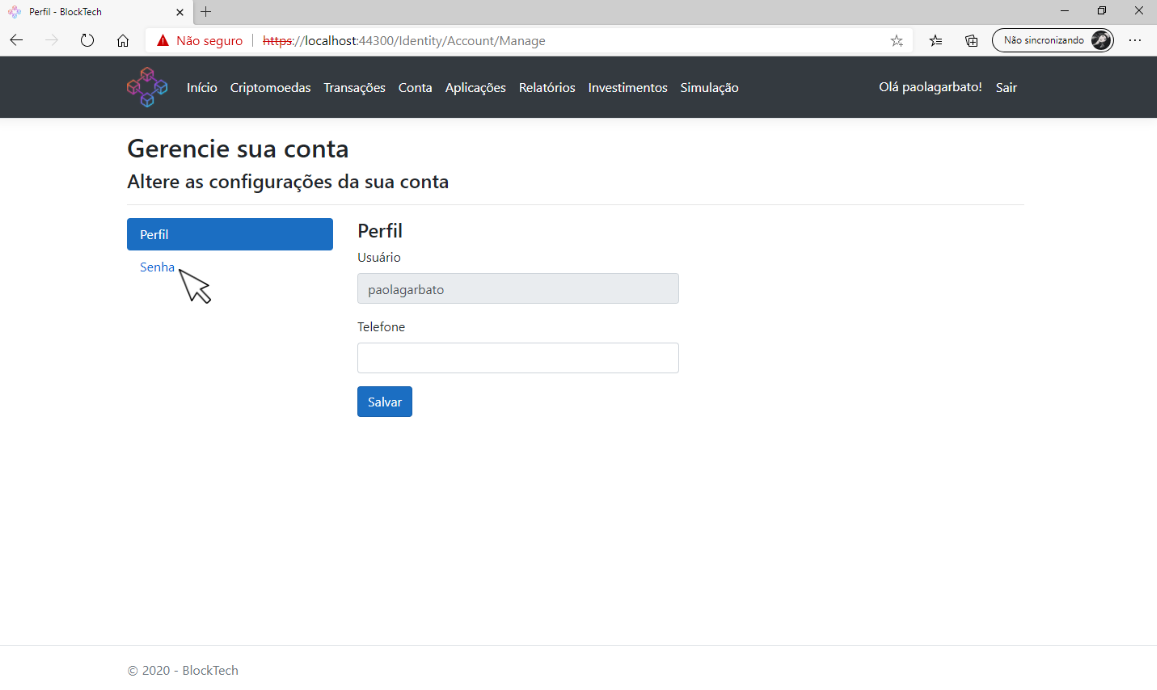
Figura 50 - Aplicativo Web - Alterando o Telefone



Fonte: Paola Regina Garbato (2020)

4. Para alterar a senha, clique em “Senha”, conforme a Figura 51.

Figura 51 - Aplicativo Web - Acessando Alteração de Senha



Fonte: Paola Regina Garbato (2020)

5. Para alterar a senha, insira a senha atual, e a nova senha – essa, precisa ser confirmada, conforme mostram os campos da Figura 52.

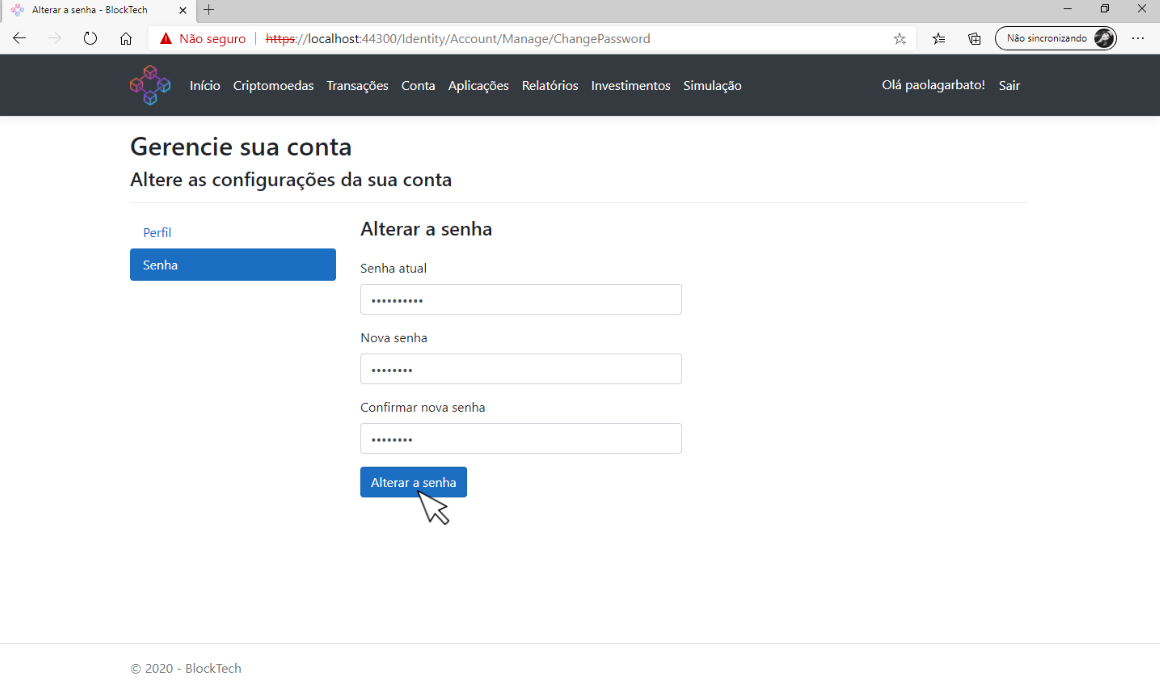
Figura 52 - Aplicativo Web - Alterar Senha



Fonte: Paola Regina Garbato (2020)

6. Para finalizar a alteração de senha, após inserir os dados, clique em “Alterar Senha”, conforme a Figura 53.

Figura 53 - Aplicativo Web - Alterar Senha



Fonte: Paola Regina Garbato (2020)

**Sair**

1. Para sair de sua conta, clique em “Sair”, conforme a Figura 54.

Figura 54 - Aplicativo Web - Sair



Fonte: Paola Regina Garbato (2020)

**APÊNDICE C - SCRIPT DE CRIAÇÃO DE TABELAS SQL SERVER**

USE [BlockTechMVC]

GO

/\*\*\*\*\*\* Object: Table [dbo].[AspNetRoles] Script Date: 04/11/2020 16:31:03 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[AspNetRoles](

[Id] [nvarchar](450) NOT NULL,

[Name] [nvarchar](256) NULL,

[NormalizedName] [nvarchar](256) NULL,

[ConcurrencyStamp] [nvarchar](max) NULL,

CONSTRAINT [PK\_AspNetRoles] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[AspNetUserRoles] Script Date: 04/11/2020 16:31:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[AspNetUserRoles](

[UserId] [nvarchar](450) NOT NULL,

[RoleId] [nvarchar](450) NOT NULL,

CONSTRAINT [PK\_AspNetUserRoles] PRIMARY KEY CLUSTERED

(

[UserId] ASC,

[RoleId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[AspNetUsers] Script Date: 04/11/2020 16:31:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[AspNetUsers](

[Id] [nvarchar](450) NOT NULL,

[UserName] [nvarchar](256) NOT NULL,

[NormalizedUserName] [nvarchar](256) NULL,

[Email] [nvarchar](256) NOT NULL,

[NormalizedEmail] [nvarchar](256) NULL,

[EmailConfirmed] [bit] NOT NULL,

[PasswordHash] [nvarchar](max) NULL,

[SecurityStamp] [nvarchar](max) NULL,

[ConcurrencyStamp] [nvarchar](max) NULL,

[PhoneNumber] [nvarchar](max) NULL,

[PhoneNumberConfirmed] [bit] NOT NULL,

[TwoFactorEnabled] [bit] NOT NULL,

[LockoutEnd] [datetimeoffset](7) NULL,

[LockoutEnabled] [bit] NOT NULL,

[AccessFailedCount] [int] NOT NULL,

[Nome] [nvarchar](60) NULL,

[Documento] [nvarchar](14) NULL,

[Cep] [nvarchar](9) NULL,

[Uf] [nvarchar](2) NULL,

[Cidade] [nvarchar](58) NULL,

[Rua] [nvarchar](max) NOT NULL,

[Numero] [nvarchar](10) NULL,

[Telefone] [nvarchar](15) NULL,

CONSTRAINT [PK\_AspNetUsers] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY],

UNIQUE NONCLUSTERED

(

[Email] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY],

UNIQUE NONCLUSTERED

(

[UserName] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Conta] Script Date: 04/11/2020 16:31:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Conta](

[Id] [int] IDENTITY(1,1) NOT NULL,

[Banco] [nvarchar](max) NOT NULL,

[Agencia] [nvarchar](4) NULL,

[NumeroConta] [nvarchar](max) NOT NULL,

[TipoConta] [int] NOT NULL,

CONSTRAINT [PK\_Conta] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[ContaCliente] Script Date: 04/11/2020 16:31:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[ContaCliente](

[Id] [int] NOT NULL,

[NumeroConta] [int] IDENTITY(27,123412) NOT NULL,

[DataAbertura] [datetime2](7) NOT NULL,

[ContaId] [int] NULL,

[ApplicationUserId] [nvarchar](450) NULL,

CONSTRAINT [PK\_ContaCliente] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Criptomoeda] Script Date: 04/11/2020 16:31:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Criptomoeda](

[Id] [int] IDENTITY(1,1) NOT NULL,

[Nome] [nvarchar](max) NULL,

[Simbolo] [nvarchar](5) NULL,

[Cadastro] [datetime2](7) NOT NULL,

CONSTRAINT [PK\_Criptomoeda] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[CriptomoedaHoje] Script Date: 04/11/2020 16:31:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[CriptomoedaHoje](

[Id] [int] IDENTITY(1,1) NOT NULL,

[Data] [datetime2](7) NOT NULL,

[Valor] [decimal](20, 2) NOT NULL,

[CriptomoedaId] [int] NOT NULL,

CONSTRAINT [PK\_CriptomoedaHoje] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[CriptoSaldo] Script Date: 04/11/2020 16:31:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[CriptoSaldo](

[Id] [int] IDENTITY(1,1) NOT NULL,

[Criptomoeda] [nvarchar](max) NULL,

[Quantidade] [decimal](20, 6) NOT NULL,

[ContaClienteId] [int] NULL,

CONSTRAINT [PK\_CriptoSaldo] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Saldo] Script Date: 04/11/2020 16:31:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Saldo](

[Id] [int] IDENTITY(1,1) NOT NULL,

[SaldoAtualRS] [decimal](18, 2) NULL,

[ContaClienteId] [int] NULL,

CONSTRAINT [PK\_Saldo] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Transacao] Script Date: 04/11/2020 16:31:04 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Transacao](

[Id] [int] IDENTITY(1,1) NOT NULL,

[Tipo] [int] NOT NULL,

[Data] [datetime2](7) NOT NULL,

[Valor] [float] NOT NULL,

[CriptomoedaHojeId] [int] NOT NULL,

[ContaClienteId] [int] NOT NULL,

[SaldoId] [int] NULL,

[CriptoSaldoId] [int] NULL,

CONSTRAINT [PK\_Transacao] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

ALTER TABLE [dbo].[AspNetUserRoles] WITH CHECK ADD CONSTRAINT [FK\_AspNetUserRoles\_AspNetRoles\_RoleId] FOREIGN KEY([RoleId])

REFERENCES [dbo].[AspNetRoles] ([Id])

ON DELETE CASCADE

GO

ALTER TABLE [dbo].[AspNetUserRoles] CHECK CONSTRAINT [FK\_AspNetUserRoles\_AspNetRoles\_RoleId]

GO

ALTER TABLE [dbo].[AspNetUserRoles] WITH CHECK ADD CONSTRAINT [FK\_AspNetUserRoles\_AspNetUsers\_UserId] FOREIGN KEY([UserId])

REFERENCES [dbo].[AspNetUsers] ([Id])

ON DELETE CASCADE

GO

ALTER TABLE [dbo].[AspNetUserRoles] CHECK CONSTRAINT [FK\_AspNetUserRoles\_AspNetUsers\_UserId]

GO

ALTER TABLE [dbo].[ContaCliente] WITH CHECK ADD FOREIGN KEY([ApplicationUserId])

REFERENCES [dbo].[AspNetUsers] ([Id])

GO

ALTER TABLE [dbo].[ContaCliente] WITH CHECK ADD CONSTRAINT [FK\_ContaCliente\_Conta\_ContaId] FOREIGN KEY([ContaId])

REFERENCES [dbo].[Conta] ([Id])

ON DELETE CASCADE

GO

ALTER TABLE [dbo].[ContaCliente] CHECK CONSTRAINT [FK\_ContaCliente\_Conta\_ContaId]

GO

ALTER TABLE [dbo].[CriptomoedaHoje] WITH CHECK ADD CONSTRAINT [FK\_CriptomoedaHoje\_Criptomoeda\_CriptomoedaId] FOREIGN KEY([CriptomoedaId])

REFERENCES [dbo].[Criptomoeda] ([Id])

ON DELETE CASCADE

GO

ALTER TABLE [dbo].[CriptomoedaHoje] CHECK CONSTRAINT [FK\_CriptomoedaHoje\_Criptomoeda\_CriptomoedaId]

GO

ALTER TABLE [dbo].[CriptoSaldo] WITH CHECK ADD FOREIGN KEY([ContaClienteId])

REFERENCES [dbo].[ContaCliente] ([Id])

GO

ALTER TABLE [dbo].[Saldo] WITH CHECK ADD FOREIGN KEY([ContaClienteId])

REFERENCES [dbo].[ContaCliente] ([Id])

GO

ALTER TABLE [dbo].[Transacao] WITH CHECK ADD FOREIGN KEY([CriptoSaldoId])

REFERENCES [dbo].[CriptoSaldo] ([Id])

GO

ALTER TABLE [dbo].[Transacao] WITH CHECK ADD CONSTRAINT [FK\_Transacao\_ContaCliente\_ContaClienteId] FOREIGN KEY([ContaClienteId])

REFERENCES [dbo].[ContaCliente] ([Id])

ON DELETE CASCADE

GO

ALTER TABLE [dbo].[Transacao] CHECK CONSTRAINT [FK\_Transacao\_ContaCliente\_ContaClienteId]

GO

ALTER TABLE [dbo].[Transacao] WITH CHECK ADD CONSTRAINT [FK\_Transacao\_CriptomoedaHoje\_CriptomoedaHojeId] FOREIGN KEY([CriptomoedaHojeId])

REFERENCES [dbo].[CriptomoedaHoje] ([Id])

ON DELETE CASCADE

GO

ALTER TABLE [dbo].[Transacao] CHECK CONSTRAINT [FK\_Transacao\_CriptomoedaHoje\_CriptomoedaHojeId]

GO

ALTER TABLE [dbo].[Transacao] WITH CHECK ADD CONSTRAINT [FK\_Transacao\_Saldo\_SaldoId] FOREIGN KEY([SaldoId])

REFERENCES [dbo].[Saldo] ([Id])

GO

ALTER TABLE [dbo].[Transacao] CHECK CONSTRAINT [FK\_Transacao\_Saldo\_SaldoId]

GO

**APÊNDICE D – APLICATIVO WEB**

**CSS**

/\* Please see documentation at https://docs.microsoft.com/aspnet/core/client-side/bundling-and-minification

for details on configuring this project to bundle and minify static web assets. \*/

a.navbar-brand {

white-space: normal;

text-align: center;

word-break: break-all;

}

/\* Provide sufficient contrast against white background \*/

a {

color: #0366d6;

}

.btn-primary {

color: #fff;

background-color: #1b6ec2;

border-color: #1861ac;

}

.nav-pills .nav-link.active, .nav-pills .show > .nav-link {

color: #fff;

background-color: #1b6ec2;

border-color: #1861ac;

}

/\* Sticky footer styles

-------------------------------------------------- \*/

html {

font-size: 14px;

}

@media (min-width: 768px) {

html {

font-size: 16px;

}

}

.border-top {

border-top: 1px solid #e5e5e5;

}

.border-bottom {

border-bottom: 1px solid #e5e5e5;

}

.box-shadow {

box-shadow: 0 .25rem .75rem rgba(0, 0, 0, .05);

}

button.accept-policy {

font-size: 1rem;

line-height: inherit;

}

/\* Sticky footer styles

-------------------------------------------------- \*/

html {

position: relative;

min-height: 100%;

}

body {

/\* Margin bottom by footer height \*/

margin-bottom: 60px;

}

.footer {

position: absolute;

bottom: 0;

width: 100%;

white-space: nowrap;

line-height: 60px; /\* Vertically center the text there \*/

}

.home {

font-size: 150%;

line-height: 160%;

max-width: 85%;

}

.line {

font-size: 140%;

line-height: 160%;

max-width: 80%;

}

.imgOrder {

background-image: url('../images/ordenar.png');

background-repeat: no-repeat;

background-size: 15px;

padding-left: 18px;

color: black;

}

#bitcoinLegenda, #ethereumLegenda, #bitcoinCashLegenda, #xrpLegenda, #paxGoldLegenda, #litecoinLegenda {

width: 40px;

height: 12px;

margin: 6px 5px 0px 0;

}

#bitcoinLegenda {

background-color: rgb(250, 150, 26);

}

#ethereumLegenda {

background-color: rgb(107, 108, 109);

}

#bitcoinCashLegenda {

background-color: rgb(141, 195, 81);

}

#xrpLegenda {

background-color: rgb(35, 41, 47);

}

#paxGoldLegenda {

background-color: rgb(250, 244, 15);

}

#litecoinLegenda {

background-color: rgb(52, 92, 157);

}

.textoLegenda {

margin: 0 5px;

}

.legendas {

display: flex;

flex-wrap: nowrap;

}

.menu {

margin: 5px;

}

.criptomoedasImg {

display: flex;

flex-direction: row;

margin-bottom: 35px;

}

.simboloCripto {

display: flex;

justify-content: center;

margin-right: 15px;

max-width: 90%;

}

.bg-light-d {

background-color: #e5e5e5;

width: 300px;

height: 350px;

border-radius: 10px;

margin: 0 auto;

}

#graficoMes, #botaoSemana {

display: none;

}

**JavaScript**

function TrocarGrafico() {

document.getElementById('graficoSemana').style.display = 'none'

document.getElementById('graficoMes').style.display = 'block'

document.getElementById('botaoMes').style.display = 'none'

document.getElementById('botaoSemana').style.display = 'block'

}

function TrocarGraficoSemana() {

document.getElementById('graficoSemana').style.display = 'block'

document.getElementById('graficoMes').style.display = 'none'

document.getElementById('botaoMes').style.display = 'block'

document.getElementById('botaoSemana').style.display = 'none'

}

**ApplicationDbContext**

using System;

using System.Collections.Generic;

using System.Text;

using BlockTechMVC.Models;

using Microsoft.AspNetCore.Identity.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore;

namespace BlockTechMVC.Data

{

public class ApplicationDbContext : IdentityDbContext<ApplicationUser>

{

public ApplicationDbContext(DbContextOptions<ApplicationDbContext> options)

: base(options)

{

}

public DbSet<BlockTechMVC.Models.Transacao> Transacao { get; set; }

public DbSet<BlockTechMVC.Models.Criptomoeda> Criptomoeda { get; set; }

public DbSet<BlockTechMVC.Models.CriptomoedaHoje> CriptomoedaHoje { get; set; }

public DbSet<BlockTechMVC.Models.ApplicationUser> ApplicationUser { get; set; }

public DbSet<BlockTechMVC.Models.Conta> Conta { get; set; }

public DbSet<BlockTechMVC.Models.ContaCliente> ContaCliente { get; set; }

public DbSet<BlockTechMVC.Models.Saldo> Saldo { get; set; }

public DbSet<BlockTechMVC.Models.CriptoSaldo> CriptoSaldo { get; set; }

}

}

**Migrations – ApplicationDbContextModelSnapshot**

// <auto-generated />

using System;

using BlockTechMVC.Data;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Infrastructure;

using Microsoft.EntityFrameworkCore.Metadata;

using Microsoft.EntityFrameworkCore.Storage.ValueConversion;

namespace BlockTechMVC.Migrations

{

[DbContext(typeof(ApplicationDbContext))]

partial class ApplicationDbContextModelSnapshot : ModelSnapshot

{

protected override void BuildModel(ModelBuilder modelBuilder)

{

#pragma warning disable 612, 618

modelBuilder

.HasAnnotation("ProductVersion", "3.1.9")

.HasAnnotation("Relational:MaxIdentifierLength", 128)

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

modelBuilder.Entity("BlockTechMVC.Models.ApplicationUser", b =>

{

b.Property<string>("Id")

.HasColumnType("nvarchar(450)");

b.Property<int>("AccessFailedCount")

.HasColumnType("int");

b.Property<string>("Cep")

.HasColumnType("nvarchar(9)")

.HasMaxLength(9);

b.Property<string>("Cidade")

.HasColumnType("nvarchar(58)")

.HasMaxLength(58);

b.Property<string>("ConcurrencyStamp")

.IsConcurrencyToken()

.HasColumnType("nvarchar(max)");

b.Property<string>("Documento")

.HasColumnType("nvarchar(14)")

.HasMaxLength(14);

b.Property<string>("Email")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<bool>("EmailConfirmed")

.HasColumnType("bit");

b.Property<bool>("LockoutEnabled")

.HasColumnType("bit");

b.Property<DateTimeOffset?>("LockoutEnd")

.HasColumnType("datetimeoffset");

b.Property<string>("Nome")

.HasColumnType("nvarchar(60)")

.HasMaxLength(60);

b.Property<string>("NormalizedEmail")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<string>("NormalizedUserName")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<string>("Numero")

.HasColumnType("nvarchar(10)")

.HasMaxLength(10);

b.Property<string>("PasswordHash")

.HasColumnType("nvarchar(max)");

b.Property<string>("PhoneNumber")

.HasColumnType("nvarchar(max)");

b.Property<bool>("PhoneNumberConfirmed")

.HasColumnType("bit");

b.Property<string>("Rua")

.HasColumnType("nvarchar(max)");

b.Property<string>("SecurityStamp")

.HasColumnType("nvarchar(max)");

b.Property<string>("Telefone")

.HasColumnType("nvarchar(15)")

.HasMaxLength(15);

b.Property<bool>("TwoFactorEnabled")

.HasColumnType("bit");

b.Property<string>("Uf")

.HasColumnType("nvarchar(2)")

.HasMaxLength(2);

b.Property<string>("UserName")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.HasKey("Id");

b.HasIndex("NormalizedEmail")

.HasName("EmailIndex");

b.HasIndex("NormalizedUserName")

.IsUnique()

.HasName("UserNameIndex")

.HasFilter("[NormalizedUserName] IS NOT NULL");

b.ToTable("AspNetUsers");

});

modelBuilder.Entity("BlockTechMVC.Models.Conta", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("Agencia")

.HasColumnType("nvarchar(4)")

.HasMaxLength(4);

b.Property<string>("Banco")

.HasColumnType("nvarchar(max)");

b.Property<string>("NumeroConta")

.HasColumnType("nvarchar(max)");

b.Property<int>("TipoConta")

.HasColumnType("int");

b.HasKey("Id");

b.ToTable("Conta");

});

modelBuilder.Entity("BlockTechMVC.Models.ContaCliente", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("ApplicationUserID")

.HasColumnType("nvarchar(450)");

b.Property<int>("ContaId")

.HasColumnType("int");

b.Property<DateTime>("DataAbertura")

.HasColumnType("datetime2");

b.Property<int>("NumeroConta")

.HasColumnType("int");

b.HasKey("Id");

b.HasIndex("ApplicationUserID");

b.HasIndex("ContaId");

b.ToTable("ContaCliente");

});

modelBuilder.Entity("BlockTechMVC.Models.CriptoSaldo", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<int>("ContaClienteId")

.HasColumnType("int");

b.Property<string>("Criptomoeda")

.HasColumnType("nvarchar(max)");

b.Property<decimal>("Quantidade")

.HasColumnType("decimal(20,2)");

b.HasKey("Id");

b.HasIndex("ContaClienteId");

b.ToTable("CriptoSaldo");

});

modelBuilder.Entity("BlockTechMVC.Models.Criptomoeda", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<DateTime>("Cadastro")

.HasColumnType("datetime2");

b.Property<string>("Nome")

.HasColumnType("nvarchar(max)");

b.Property<string>("Simbolo")

.HasColumnType("nvarchar(5)")

.HasMaxLength(5);

b.HasKey("Id");

b.ToTable("Criptomoeda");

});

modelBuilder.Entity("BlockTechMVC.Models.CriptomoedaHoje", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<int>("CriptomoedaId")

.HasColumnType("int");

b.Property<DateTime>("Data")

.HasColumnType("datetime2");

b.Property<decimal>("Valor")

.HasColumnType("decimal(20,2)");

b.HasKey("Id");

b.HasIndex("CriptomoedaId");

b.ToTable("CriptomoedaHoje");

});

modelBuilder.Entity("BlockTechMVC.Models.Saldo", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<int>("ContaClienteId")

.HasColumnType("int");

b.Property<decimal>("SaldoAtualRS")

.HasColumnType("decimal(20,2)");

b.HasKey("Id");

b.HasIndex("ContaClienteId");

b.ToTable("Saldo");

});

modelBuilder.Entity("BlockTechMVC.Models.Transacao", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<int>("ContaClienteId")

.HasColumnType("int");

b.Property<int>("CriptoSaldoId")

.HasColumnType("int");

b.Property<int>("CriptomoedaHojeId")

.HasColumnType("int");

b.Property<DateTime>("Data")

.HasColumnType("datetime2");

b.Property<int>("SaldoId")

.HasColumnType("int");

b.Property<int>("Tipo")

.HasColumnType("int");

b.Property<double>("Valor")

.HasColumnType("float");

b.HasKey("Id");

b.HasIndex("ContaClienteId");

b.HasIndex("CriptoSaldoId");

b.HasIndex("CriptomoedaHojeId");

b.HasIndex("SaldoId");

b.ToTable("Transacao");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityRole", b =>

{

b.Property<string>("Id")

.HasColumnType("nvarchar(450)");

b.Property<string>("ConcurrencyStamp")

.IsConcurrencyToken()

.HasColumnType("nvarchar(max)");

b.Property<string>("Name")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<string>("NormalizedName")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.HasKey("Id");

b.HasIndex("NormalizedName")

.IsUnique()

.HasName("RoleNameIndex")

.HasFilter("[NormalizedName] IS NOT NULL");

b.ToTable("AspNetRoles");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityRoleClaim<string>", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("ClaimType")

.HasColumnType("nvarchar(max)");

b.Property<string>("ClaimValue")

.HasColumnType("nvarchar(max)");

b.Property<string>("RoleId")

.IsRequired()

.HasColumnType("nvarchar(450)");

b.HasKey("Id");

b.HasIndex("RoleId");

b.ToTable("AspNetRoleClaims");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserClaim<string>", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("ClaimType")

.HasColumnType("nvarchar(max)");

b.Property<string>("ClaimValue")

.HasColumnType("nvarchar(max)");

b.Property<string>("UserId")

.IsRequired()

.HasColumnType("nvarchar(450)");

b.HasKey("Id");

b.HasIndex("UserId");

b.ToTable("AspNetUserClaims");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserLogin<string>", b =>

{

b.Property<string>("LoginProvider")

.HasColumnType("nvarchar(128)")

.HasMaxLength(128);

b.Property<string>("ProviderKey")

.HasColumnType("nvarchar(128)")

.HasMaxLength(128);

b.Property<string>("ProviderDisplayName")

.HasColumnType("nvarchar(max)");

b.Property<string>("UserId")

.IsRequired()

.HasColumnType("nvarchar(450)");

b.HasKey("LoginProvider", "ProviderKey");

b.HasIndex("UserId");

b.ToTable("AspNetUserLogins");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserRole<string>", b =>

{

b.Property<string>("UserId")

.HasColumnType("nvarchar(450)");

b.Property<string>("RoleId")

.HasColumnType("nvarchar(450)");

b.HasKey("UserId", "RoleId");

b.HasIndex("RoleId");

b.ToTable("AspNetUserRoles");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserToken<string>", b =>

{

b.Property<string>("UserId")

.HasColumnType("nvarchar(450)");

b.Property<string>("LoginProvider")

.HasColumnType("nvarchar(128)")

.HasMaxLength(128);

b.Property<string>("Name")

.HasColumnType("nvarchar(128)")

.HasMaxLength(128);

b.Property<string>("Value")

.HasColumnType("nvarchar(max)");

b.HasKey("UserId", "LoginProvider", "Name");

b.ToTable("AspNetUserTokens");

});

modelBuilder.Entity("BlockTechMVC.Models.ContaCliente", b =>

{

b.HasOne("BlockTechMVC.Models.ApplicationUser", "ApplicationUser")

.WithMany()

.HasForeignKey("ApplicationUserID");

b.HasOne("BlockTechMVC.Models.Conta", "Conta")

.WithMany()

.HasForeignKey("ContaId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("BlockTechMVC.Models.CriptoSaldo", b =>

{

b.HasOne("BlockTechMVC.Models.ContaCliente", "ContaCliente")

.WithMany()

.HasForeignKey("ContaClienteId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("BlockTechMVC.Models.CriptomoedaHoje", b =>

{

b.HasOne("BlockTechMVC.Models.Criptomoeda", "Criptomoeda")

.WithMany()

.HasForeignKey("CriptomoedaId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("BlockTechMVC.Models.Saldo", b =>

{

b.HasOne("BlockTechMVC.Models.ContaCliente", "ContaCliente")

.WithMany()

.HasForeignKey("ContaClienteId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("BlockTechMVC.Models.Transacao", b =>

{

b.HasOne("BlockTechMVC.Models.ContaCliente", "ContaCliente")

.WithMany()

.HasForeignKey("ContaClienteId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("BlockTechMVC.Models.CriptoSaldo", "CriptoSaldo")

.WithMany()

.HasForeignKey("CriptoSaldoId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("BlockTechMVC.Models.CriptomoedaHoje", "CriptomoedaHoje")

.WithMany()

.HasForeignKey("CriptomoedaHojeId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("BlockTechMVC.Models.Saldo", "Saldo")

.WithMany()

.HasForeignKey("SaldoId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityRoleClaim<string>", b =>

{

b.HasOne("Microsoft.AspNetCore.Identity.IdentityRole", null)

.WithMany()

.HasForeignKey("RoleId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserClaim<string>", b =>

{

b.HasOne("BlockTechMVC.Models.ApplicationUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserLogin<string>", b =>

{

b.HasOne("BlockTechMVC.Models.ApplicationUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserRole<string>", b =>

{

b.HasOne("Microsoft.AspNetCore.Identity.IdentityRole", null)

.WithMany()

.HasForeignKey("RoleId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("BlockTechMVC.Models.ApplicationUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserToken<string>", b =>

{

b.HasOne("BlockTechMVC.Models.ApplicationUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

#pragma warning restore 612, 618

}

}

}

**Migrations - 20201026225003\_Transacoes.Designer**

// <auto-generated />

using System;

using BlockTechMVC.Data;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Infrastructure;

using Microsoft.EntityFrameworkCore.Metadata;

using Microsoft.EntityFrameworkCore.Migrations;

using Microsoft.EntityFrameworkCore.Storage.ValueConversion;

namespace BlockTechMVC.Migrations

{

[DbContext(typeof(ApplicationDbContext))]

[Migration("20201026225003\_Transacoes")]

partial class Transacoes

{

protected override void BuildTargetModel(ModelBuilder modelBuilder)

{

#pragma warning disable 612, 618

modelBuilder

.HasAnnotation("ProductVersion", "3.1.9")

.HasAnnotation("Relational:MaxIdentifierLength", 128)

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

modelBuilder.Entity("BlockTechMVC.Models.ApplicationUser", b =>

{

b.Property<string>("Id")

.HasColumnType("nvarchar(450)");

b.Property<int>("AccessFailedCount")

.HasColumnType("int");

b.Property<string>("Cep")

.HasColumnType("nvarchar(9)")

.HasMaxLength(9);

b.Property<string>("Cidade")

.HasColumnType("nvarchar(58)")

.HasMaxLength(58);

b.Property<string>("ConcurrencyStamp")

.IsConcurrencyToken()

.HasColumnType("nvarchar(max)");

b.Property<string>("Documento")

.HasColumnType("nvarchar(14)")

.HasMaxLength(14);

b.Property<string>("Email")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<bool>("EmailConfirmed")

.HasColumnType("bit");

b.Property<bool>("LockoutEnabled")

.HasColumnType("bit");

b.Property<DateTimeOffset?>("LockoutEnd")

.HasColumnType("datetimeoffset");

b.Property<string>("Nome")

.HasColumnType("nvarchar(60)")

.HasMaxLength(60);

b.Property<string>("NormalizedEmail")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<string>("NormalizedUserName")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<string>("Numero")

.HasColumnType("nvarchar(10)")

.HasMaxLength(10);

b.Property<string>("PasswordHash")

.HasColumnType("nvarchar(max)");

b.Property<string>("PhoneNumber")

.HasColumnType("nvarchar(max)");

b.Property<bool>("PhoneNumberConfirmed")

.HasColumnType("bit");

b.Property<string>("Rua")

.HasColumnType("nvarchar(max)");

b.Property<string>("SecurityStamp")

.HasColumnType("nvarchar(max)");

b.Property<string>("Telefone")

.HasColumnType("nvarchar(15)")

.HasMaxLength(15);

b.Property<bool>("TwoFactorEnabled")

.HasColumnType("bit");

b.Property<string>("Uf")

.HasColumnType("nvarchar(2)")

.HasMaxLength(2);

b.Property<string>("UserName")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.HasKey("Id");

b.HasIndex("NormalizedEmail")

.HasName("EmailIndex");

b.HasIndex("NormalizedUserName")

.IsUnique()

.HasName("UserNameIndex")

.HasFilter("[NormalizedUserName] IS NOT NULL");

b.ToTable("AspNetUsers");

});

modelBuilder.Entity("BlockTechMVC.Models.Conta", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("Agencia")

.HasColumnType("nvarchar(4)")

.HasMaxLength(4);

b.Property<string>("Banco")

.HasColumnType("nvarchar(max)");

b.Property<string>("NumeroConta")

.HasColumnType("nvarchar(max)");

b.Property<int>("TipoConta")

.HasColumnType("int");

b.HasKey("Id");

b.ToTable("Conta");

});

modelBuilder.Entity("BlockTechMVC.Models.ContaCliente", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("ApplicationUserID")

.HasColumnType("nvarchar(450)");

b.Property<int>("ContaId")

.HasColumnType("int");

b.Property<DateTime>("DataAbertura")

.HasColumnType("datetime2");

b.Property<int>("NumeroConta")

.HasColumnType("int");

b.HasKey("Id");

b.HasIndex("ApplicationUserID");

b.HasIndex("ContaId");

b.ToTable("ContaCliente");

});

modelBuilder.Entity("BlockTechMVC.Models.CriptoSaldo", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<int>("ContaClienteId")

.HasColumnType("int");

b.Property<string>("Criptomoeda")

.HasColumnType("nvarchar(max)");

b.Property<decimal>("Quantidade")

.HasColumnType("decimal(20,2)");

b.HasKey("Id");

b.HasIndex("ContaClienteId");

b.ToTable("CriptoSaldo");

});

modelBuilder.Entity("BlockTechMVC.Models.Criptomoeda", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<DateTime>("Cadastro")

.HasColumnType("datetime2");

b.Property<string>("Nome")

.HasColumnType("nvarchar(max)");

b.Property<string>("Simbolo")

.HasColumnType("nvarchar(5)")

.HasMaxLength(5);

b.HasKey("Id");

b.ToTable("Criptomoeda");

});

modelBuilder.Entity("BlockTechMVC.Models.CriptomoedaHoje", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<int>("CriptomoedaId")

.HasColumnType("int");

b.Property<DateTime>("Data")

.HasColumnType("datetime2");

b.Property<decimal>("Valor")

.HasColumnType("decimal(20,2)");

b.HasKey("Id");

b.HasIndex("CriptomoedaId");

b.ToTable("CriptomoedaHoje");

});

modelBuilder.Entity("BlockTechMVC.Models.Saldo", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<int>("ContaClienteId")

.HasColumnType("int");

b.Property<decimal>("SaldoAtualRS")

.HasColumnType("decimal(20,2)");

b.HasKey("Id");

b.HasIndex("ContaClienteId");

b.ToTable("Saldo");

});

modelBuilder.Entity("BlockTechMVC.Models.Transacao", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<int>("ContaClienteId")

.HasColumnType("int");

b.Property<int>("CriptoSaldoId")

.HasColumnType("int");

b.Property<int>("CriptomoedaHojeId")

.HasColumnType("int");

b.Property<DateTime>("Data")

.HasColumnType("datetime2");

b.Property<int>("SaldoId")

.HasColumnType("int");

b.Property<int>("Tipo")

.HasColumnType("int");

b.Property<double>("Valor")

.HasColumnType("float");

b.HasKey("Id");

b.HasIndex("ContaClienteId");

b.HasIndex("CriptoSaldoId");

b.HasIndex("CriptomoedaHojeId");

b.HasIndex("SaldoId");

b.ToTable("Transacao");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityRole", b =>

{

b.Property<string>("Id")

.HasColumnType("nvarchar(450)");

b.Property<string>("ConcurrencyStamp")

.IsConcurrencyToken()

.HasColumnType("nvarchar(max)");

b.Property<string>("Name")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<string>("NormalizedName")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.HasKey("Id");

b.HasIndex("NormalizedName")

.IsUnique()

.HasName("RoleNameIndex")

.HasFilter("[NormalizedName] IS NOT NULL");

b.ToTable("AspNetRoles");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityRoleClaim<string>", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("ClaimType")

.HasColumnType("nvarchar(max)");

b.Property<string>("ClaimValue")

.HasColumnType("nvarchar(max)");

b.Property<string>("RoleId")

.IsRequired()

.HasColumnType("nvarchar(450)");

b.HasKey("Id");

b.HasIndex("RoleId");

b.ToTable("AspNetRoleClaims");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserClaim<string>", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("ClaimType")

.HasColumnType("nvarchar(max)");

b.Property<string>("ClaimValue")

.HasColumnType("nvarchar(max)");

b.Property<string>("UserId")

.IsRequired()

.HasColumnType("nvarchar(450)");

b.HasKey("Id");

b.HasIndex("UserId");

b.ToTable("AspNetUserClaims");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserLogin<string>", b =>

{

b.Property<string>("LoginProvider")

.HasColumnType("nvarchar(128)")

.HasMaxLength(128);

b.Property<string>("ProviderKey")

.HasColumnType("nvarchar(128)")

.HasMaxLength(128);

b.Property<string>("ProviderDisplayName")

.HasColumnType("nvarchar(max)");

b.Property<string>("UserId")

.IsRequired()

.HasColumnType("nvarchar(450)");

b.HasKey("LoginProvider", "ProviderKey");

b.HasIndex("UserId");

b.ToTable("AspNetUserLogins");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserRole<string>", b =>

{

b.Property<string>("UserId")

.HasColumnType("nvarchar(450)");

b.Property<string>("RoleId")

.HasColumnType("nvarchar(450)");

b.HasKey("UserId", "RoleId");

b.HasIndex("RoleId");

b.ToTable("AspNetUserRoles");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserToken<string>", b =>

{

b.Property<string>("UserId")

.HasColumnType("nvarchar(450)");

b.Property<string>("LoginProvider")

.HasColumnType("nvarchar(128)")

.HasMaxLength(128);

b.Property<string>("Name")

.HasColumnType("nvarchar(128)")

.HasMaxLength(128);

b.Property<string>("Value")

.HasColumnType("nvarchar(max)");

b.HasKey("UserId", "LoginProvider", "Name");

b.ToTable("AspNetUserTokens");

});

modelBuilder.Entity("BlockTechMVC.Models.ContaCliente", b =>

{

b.HasOne("BlockTechMVC.Models.ApplicationUser", "ApplicationUser")

.WithMany()

.HasForeignKey("ApplicationUserID");

b.HasOne("BlockTechMVC.Models.Conta", "Conta")

.WithMany()

.HasForeignKey("ContaId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("BlockTechMVC.Models.CriptoSaldo", b =>

{

b.HasOne("BlockTechMVC.Models.ContaCliente", "ContaCliente")

.WithMany()

.HasForeignKey("ContaClienteId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("BlockTechMVC.Models.CriptomoedaHoje", b =>

{

b.HasOne("BlockTechMVC.Models.Criptomoeda", "Criptomoeda")

.WithMany()

.HasForeignKey("CriptomoedaId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("BlockTechMVC.Models.Saldo", b =>

{

b.HasOne("BlockTechMVC.Models.ContaCliente", "ContaCliente")

.WithMany()

.HasForeignKey("ContaClienteId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("BlockTechMVC.Models.Transacao", b =>

{

b.HasOne("BlockTechMVC.Models.ContaCliente", "ContaCliente")

.WithMany()

.HasForeignKey("ContaClienteId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("BlockTechMVC.Models.CriptoSaldo", "CriptoSaldo")

.WithMany()

.HasForeignKey("CriptoSaldoId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("BlockTechMVC.Models.CriptomoedaHoje", "CriptomoedaHoje")

.WithMany()

.HasForeignKey("CriptomoedaHojeId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("BlockTechMVC.Models.Saldo", "Saldo")

.WithMany()

.HasForeignKey("SaldoId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityRoleClaim<string>", b =>

{

b.HasOne("Microsoft.AspNetCore.Identity.IdentityRole", null)

.WithMany()

.HasForeignKey("RoleId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserClaim<string>", b =>

{

b.HasOne("BlockTechMVC.Models.ApplicationUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserLogin<string>", b =>

{

b.HasOne("BlockTechMVC.Models.ApplicationUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserRole<string>", b =>

{

b.HasOne("Microsoft.AspNetCore.Identity.IdentityRole", null)

.WithMany()

.HasForeignKey("RoleId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("BlockTechMVC.Models.ApplicationUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserToken<string>", b =>

{

b.HasOne("BlockTechMVC.Models.ApplicationUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

#pragma warning restore 612, 618

}

}

}

**Migrations - 20201023164722\_Transacao\_Saldo\_CriptoSaldo.Designer**

// <auto-generated />

using System;

using BlockTechMVC.Data;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Infrastructure;

using Microsoft.EntityFrameworkCore.Metadata;

using Microsoft.EntityFrameworkCore.Migrations;

using Microsoft.EntityFrameworkCore.Storage.ValueConversion;

namespace BlockTechMVC.Migrations

{

[DbContext(typeof(ApplicationDbContext))]

[Migration("20201023164722\_Transacao\_Saldo\_CriptoSaldo")]

partial class Transacao\_Saldo\_CriptoSaldo

{

protected override void BuildTargetModel(ModelBuilder modelBuilder)

{

#pragma warning disable 612, 618

modelBuilder

.HasAnnotation("ProductVersion", "3.1.9")

.HasAnnotation("Relational:MaxIdentifierLength", 128)

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

modelBuilder.Entity("BlockTechMVC.Models.ApplicationUser", b =>

{

b.Property<string>("Id")

.HasColumnType("nvarchar(450)");

b.Property<int>("AccessFailedCount")

.HasColumnType("int");

b.Property<string>("Cep")

.HasColumnType("nvarchar(9)")

.HasMaxLength(9);

b.Property<string>("Cidade")

.HasColumnType("nvarchar(58)")

.HasMaxLength(58);

b.Property<string>("ConcurrencyStamp")

.IsConcurrencyToken()

.HasColumnType("nvarchar(max)");

b.Property<string>("Documento")

.HasColumnType("nvarchar(14)")

.HasMaxLength(14);

b.Property<string>("Email")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<bool>("EmailConfirmed")

.HasColumnType("bit");

b.Property<bool>("LockoutEnabled")

.HasColumnType("bit");

b.Property<DateTimeOffset?>("LockoutEnd")

.HasColumnType("datetimeoffset");

b.Property<string>("Nome")

.HasColumnType("nvarchar(60)")

.HasMaxLength(60);

b.Property<string>("NormalizedEmail")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<string>("NormalizedUserName")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<string>("Numero")

.HasColumnType("nvarchar(10)")

.HasMaxLength(10);

b.Property<string>("PasswordHash")

.HasColumnType("nvarchar(max)");

b.Property<string>("PhoneNumber")

.HasColumnType("nvarchar(max)");

b.Property<bool>("PhoneNumberConfirmed")

.HasColumnType("bit");

b.Property<string>("Rua")

.HasColumnType("nvarchar(max)");

b.Property<string>("SecurityStamp")

.HasColumnType("nvarchar(max)");

b.Property<string>("Telefone")

.HasColumnType("nvarchar(15)")

.HasMaxLength(15);

b.Property<bool>("TwoFactorEnabled")

.HasColumnType("bit");

b.Property<string>("Uf")

.HasColumnType("nvarchar(2)")

.HasMaxLength(2);

b.Property<string>("UserName")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.HasKey("Id");

b.HasIndex("NormalizedEmail")

.HasName("EmailIndex");

b.HasIndex("NormalizedUserName")

.IsUnique()

.HasName("UserNameIndex")

.HasFilter("[NormalizedUserName] IS NOT NULL");

b.ToTable("AspNetUsers");

});

modelBuilder.Entity("BlockTechMVC.Models.Conta", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("Agencia")

.HasColumnType("nvarchar(4)")

.HasMaxLength(4);

b.Property<string>("Banco")

.HasColumnType("nvarchar(max)");

b.Property<string>("NumeroConta")

.HasColumnType("nvarchar(max)");

b.Property<int>("TipoConta")

.HasColumnType("int");

b.HasKey("Id");

b.ToTable("Conta");

});

modelBuilder.Entity("BlockTechMVC.Models.ContaCliente", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("ApplicationUserID")

.HasColumnType("nvarchar(450)");

b.Property<int>("ContaId")

.HasColumnType("int");

b.Property<DateTime>("DataAbertura")

.HasColumnType("datetime2");

b.Property<int>("NumeroConta")

.HasColumnType("int");

b.HasKey("Id");

b.HasIndex("ApplicationUserID");

b.HasIndex("ContaId");

b.ToTable("ContaCliente");

});

modelBuilder.Entity("BlockTechMVC.Models.CriptoSaldo", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<int>("ContaClienteId")

.HasColumnType("int");

b.Property<string>("Criptomoeda")

.HasColumnType("nvarchar(max)");

b.Property<decimal>("Quantidade")

.HasColumnType("decimal(20,2)");

b.HasKey("Id");

b.HasIndex("ContaClienteId");

b.ToTable("CriptoSaldo");

});

modelBuilder.Entity("BlockTechMVC.Models.Criptomoeda", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<DateTime>("Cadastro")

.HasColumnType("datetime2");

b.Property<string>("Nome")

.HasColumnType("nvarchar(max)");

b.Property<string>("Simbolo")

.HasColumnType("nvarchar(5)")

.HasMaxLength(5);

b.HasKey("Id");

b.ToTable("Criptomoeda");

});

modelBuilder.Entity("BlockTechMVC.Models.CriptomoedaHoje", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<int>("CriptomoedaId")

.HasColumnType("int");

b.Property<DateTime>("Data")

.HasColumnType("datetime2");

b.Property<decimal>("Valor")

.HasColumnType("decimal(20,2)");

b.HasKey("Id");

b.HasIndex("CriptomoedaId");

b.ToTable("CriptomoedaHoje");

});

modelBuilder.Entity("BlockTechMVC.Models.Saldo", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<int>("ContaClienteId")

.HasColumnType("int");

b.Property<decimal>("SaldoAtualRS")

.HasColumnType("decimal(20,2)");

b.HasKey("Id");

b.HasIndex("ContaClienteId");

b.ToTable("Saldo");

});

modelBuilder.Entity("BlockTechMVC.Models.Transacao", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<int>("ContaClienteId")

.HasColumnType("int");

b.Property<int>("CriptoSaldoId")

.HasColumnType("int");

b.Property<int>("CriptomoedaHojeId")

.HasColumnType("int");

b.Property<DateTime>("Data")

.HasColumnType("datetime2");

b.Property<int>("SaldoId")

.HasColumnType("int");

b.Property<int>("Tipo")

.HasColumnType("int");

b.Property<decimal>("Valor")

.HasColumnType("decimal(20,2)");

b.HasKey("Id");

b.HasIndex("ContaClienteId");

b.HasIndex("CriptoSaldoId");

b.HasIndex("CriptomoedaHojeId");

b.HasIndex("SaldoId");

b.ToTable("Transacao");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityRole", b =>

{

b.Property<string>("Id")

.HasColumnType("nvarchar(450)");

b.Property<string>("ConcurrencyStamp")

.IsConcurrencyToken()

.HasColumnType("nvarchar(max)");

b.Property<string>("Name")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<string>("NormalizedName")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.HasKey("Id");

b.HasIndex("NormalizedName")

.IsUnique()

.HasName("RoleNameIndex")

.HasFilter("[NormalizedName] IS NOT NULL");

b.ToTable("AspNetRoles");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityRoleClaim<string>", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("ClaimType")

.HasColumnType("nvarchar(max)");

b.Property<string>("ClaimValue")

.HasColumnType("nvarchar(max)");

b.Property<string>("RoleId")

.IsRequired()

.HasColumnType("nvarchar(450)");

b.HasKey("Id");

b.HasIndex("RoleId");

b.ToTable("AspNetRoleClaims");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserClaim<string>", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("ClaimType")

.HasColumnType("nvarchar(max)");

b.Property<string>("ClaimValue")

.HasColumnType("nvarchar(max)");

b.Property<string>("UserId")

.IsRequired()

.HasColumnType("nvarchar(450)");

b.HasKey("Id");

b.HasIndex("UserId");

b.ToTable("AspNetUserClaims");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserLogin<string>", b =>

{

b.Property<string>("LoginProvider")

.HasColumnType("nvarchar(128)")

.HasMaxLength(128);

b.Property<string>("ProviderKey")

.HasColumnType("nvarchar(128)")

.HasMaxLength(128);

b.Property<string>("ProviderDisplayName")

.HasColumnType("nvarchar(max)");

b.Property<string>("UserId")

.IsRequired()

.HasColumnType("nvarchar(450)");

b.HasKey("LoginProvider", "ProviderKey");

b.HasIndex("UserId");

b.ToTable("AspNetUserLogins");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserRole<string>", b =>

{

b.Property<string>("UserId")

.HasColumnType("nvarchar(450)");

b.Property<string>("RoleId")

.HasColumnType("nvarchar(450)");

b.HasKey("UserId", "RoleId");

b.HasIndex("RoleId");

b.ToTable("AspNetUserRoles");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserToken<string>", b =>

{

b.Property<string>("UserId")

.HasColumnType("nvarchar(450)");

b.Property<string>("LoginProvider")

.HasColumnType("nvarchar(128)")

.HasMaxLength(128);

b.Property<string>("Name")

.HasColumnType("nvarchar(128)")

.HasMaxLength(128);

b.Property<string>("Value")

.HasColumnType("nvarchar(max)");

b.HasKey("UserId", "LoginProvider", "Name");

b.ToTable("AspNetUserTokens");

});

modelBuilder.Entity("BlockTechMVC.Models.ContaCliente", b =>

{

b.HasOne("BlockTechMVC.Models.ApplicationUser", "ApplicationUser")

.WithMany()

.HasForeignKey("ApplicationUserID");

b.HasOne("BlockTechMVC.Models.Conta", "Conta")

.WithMany()

.HasForeignKey("ContaId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("BlockTechMVC.Models.CriptoSaldo", b =>

{

b.HasOne("BlockTechMVC.Models.ContaCliente", "ContaCliente")

.WithMany()

.HasForeignKey("ContaClienteId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("BlockTechMVC.Models.CriptomoedaHoje", b =>

{

b.HasOne("BlockTechMVC.Models.Criptomoeda", "Criptomoeda")

.WithMany()

.HasForeignKey("CriptomoedaId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("BlockTechMVC.Models.Saldo", b =>

{

b.HasOne("BlockTechMVC.Models.ContaCliente", "ContaCliente")

.WithMany()

.HasForeignKey("ContaClienteId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("BlockTechMVC.Models.Transacao", b =>

{

b.HasOne("BlockTechMVC.Models.ContaCliente", "ContaCliente")

.WithMany()

.HasForeignKey("ContaClienteId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("BlockTechMVC.Models.CriptoSaldo", "CriptoSaldo")

.WithMany()

.HasForeignKey("CriptoSaldoId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("BlockTechMVC.Models.CriptomoedaHoje", "CriptomoedaHoje")

.WithMany()

.HasForeignKey("CriptomoedaHojeId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("BlockTechMVC.Models.Saldo", "Saldo")

.WithMany()

.HasForeignKey("SaldoId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityRoleClaim<string>", b =>

{

b.HasOne("Microsoft.AspNetCore.Identity.IdentityRole", null)

.WithMany()

.HasForeignKey("RoleId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserClaim<string>", b =>

{

b.HasOne("BlockTechMVC.Models.ApplicationUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserLogin<string>", b =>

{

b.HasOne("BlockTechMVC.Models.ApplicationUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserRole<string>", b =>

{

b.HasOne("Microsoft.AspNetCore.Identity.IdentityRole", null)

.WithMany()

.HasForeignKey("RoleId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("BlockTechMVC.Models.ApplicationUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserToken<string>", b =>

{

b.HasOne("BlockTechMVC.Models.ApplicationUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

#pragma warning restore 612, 618

}

}

}

**Migrations - 20201022210505\_Conta-int.Designer**

// <auto-generated />

using System;

using BlockTechMVC.Data;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Infrastructure;

using Microsoft.EntityFrameworkCore.Metadata;

using Microsoft.EntityFrameworkCore.Migrations;

using Microsoft.EntityFrameworkCore.Storage.ValueConversion;

namespace BlockTechMVC.Migrations

{

[DbContext(typeof(ApplicationDbContext))]

[Migration("20201022210505\_Conta-int")]

partial class Containt

{

protected override void BuildTargetModel(ModelBuilder modelBuilder)

{

#pragma warning disable 612, 618

modelBuilder

.HasAnnotation("ProductVersion", "3.1.9")

.HasAnnotation("Relational:MaxIdentifierLength", 128)

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

modelBuilder.Entity("BlockTechMVC.Models.ApplicationUser", b =>

{

b.Property<string>("Id")

.HasColumnType("nvarchar(450)");

b.Property<int>("AccessFailedCount")

.HasColumnType("int");

b.Property<string>("Cep")

.HasColumnType("nvarchar(9)")

.HasMaxLength(9);

b.Property<string>("Cidade")

.HasColumnType("nvarchar(58)")

.HasMaxLength(58);

b.Property<string>("ConcurrencyStamp")

.IsConcurrencyToken()

.HasColumnType("nvarchar(max)");

b.Property<string>("Documento")

.HasColumnType("nvarchar(14)")

.HasMaxLength(14);

b.Property<string>("Email")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<bool>("EmailConfirmed")

.HasColumnType("bit");

b.Property<bool>("LockoutEnabled")

.HasColumnType("bit");

b.Property<DateTimeOffset?>("LockoutEnd")

.HasColumnType("datetimeoffset");

b.Property<string>("Nome")

.HasColumnType("nvarchar(60)")

.HasMaxLength(60);

b.Property<string>("NormalizedEmail")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<string>("NormalizedUserName")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<string>("Numero")

.HasColumnType("nvarchar(10)")

.HasMaxLength(10);

b.Property<string>("PasswordHash")

.HasColumnType("nvarchar(max)");

b.Property<string>("PhoneNumber")

.HasColumnType("nvarchar(max)");

b.Property<bool>("PhoneNumberConfirmed")

.HasColumnType("bit");

b.Property<string>("Rua")

.HasColumnType("nvarchar(max)");

b.Property<string>("SecurityStamp")

.HasColumnType("nvarchar(max)");

b.Property<string>("Telefone")

.HasColumnType("nvarchar(15)")

.HasMaxLength(15);

b.Property<bool>("TwoFactorEnabled")

.HasColumnType("bit");

b.Property<string>("Uf")

.HasColumnType("nvarchar(2)")

.HasMaxLength(2);

b.Property<string>("UserName")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.HasKey("Id");

b.HasIndex("NormalizedEmail")

.HasName("EmailIndex");

b.HasIndex("NormalizedUserName")

.IsUnique()

.HasName("UserNameIndex")

.HasFilter("[NormalizedUserName] IS NOT NULL");

b.ToTable("AspNetUsers");

});

modelBuilder.Entity("BlockTechMVC.Models.Conta", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("Agencia")

.HasColumnType("nvarchar(4)")

.HasMaxLength(4);

b.Property<string>("Banco")

.HasColumnType("nvarchar(max)");

b.Property<string>("NumeroConta")

.HasColumnType("nvarchar(max)");

b.Property<int>("TipoConta")

.HasColumnType("int");

b.HasKey("Id");

b.ToTable("Conta");

});

modelBuilder.Entity("BlockTechMVC.Models.ContaCliente", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("ApplicationUserID")

.HasColumnType("nvarchar(450)");

b.Property<int>("ContaId")

.HasColumnType("int");

b.Property<DateTime>("DataAbertura")

.HasColumnType("datetime2");

b.Property<int>("NumeroConta")

.HasColumnType("int");

b.HasKey("Id");

b.HasIndex("ApplicationUserID");

b.HasIndex("ContaId");

b.ToTable("ContaCliente");

});

modelBuilder.Entity("BlockTechMVC.Models.CriptoSaldo", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("Criptomoeda")

.HasColumnType("nvarchar(max)");

b.Property<decimal>("Quantidade")

.HasColumnType("decimal(20,2)");

b.HasKey("Id");

b.ToTable("CriptoSaldo");

});

modelBuilder.Entity("BlockTechMVC.Models.Criptomoeda", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<DateTime>("Cadastro")

.HasColumnType("datetime2");

b.Property<string>("Nome")

.HasColumnType("nvarchar(max)");

b.Property<string>("Simbolo")

.HasColumnType("nvarchar(5)")

.HasMaxLength(5);

b.HasKey("Id");

b.ToTable("Criptomoeda");

});

modelBuilder.Entity("BlockTechMVC.Models.CriptomoedaHoje", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<int>("CriptomoedaId")

.HasColumnType("int");

b.Property<DateTime>("Data")

.HasColumnType("datetime2");

b.Property<decimal>("Valor")

.HasColumnType("decimal(20,2)");

b.HasKey("Id");

b.HasIndex("CriptomoedaId");

b.ToTable("CriptomoedaHoje");

});

modelBuilder.Entity("BlockTechMVC.Models.Saldo", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<int>("CriptoSaldoId")

.HasColumnType("int");

b.Property<decimal>("SaldoAtualRS")

.HasColumnType("decimal(20,2)");

b.Property<int>("TransacaoId")

.HasColumnType("int");

b.HasKey("Id");

b.HasIndex("CriptoSaldoId");

b.ToTable("Saldo");

});

modelBuilder.Entity("BlockTechMVC.Models.Transacao", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<int>("ContaClienteId")

.HasColumnType("int");

b.Property<int>("CriptomoedaHojeId")

.HasColumnType("int");

b.Property<DateTime>("Data")

.HasColumnType("datetime2");

b.Property<int?>("SaldoId")

.HasColumnType("int");

b.Property<int>("Tipo")

.HasColumnType("int");

b.Property<double>("Valor")

.HasColumnType("float");

b.HasKey("Id");

b.HasIndex("ContaClienteId");

b.HasIndex("CriptomoedaHojeId");

b.HasIndex("SaldoId");

b.ToTable("Transacao");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityRole", b =>

{

b.Property<string>("Id")

.HasColumnType("nvarchar(450)");

b.Property<string>("ConcurrencyStamp")

.IsConcurrencyToken()

.HasColumnType("nvarchar(max)");

b.Property<string>("Name")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<string>("NormalizedName")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.HasKey("Id");

b.HasIndex("NormalizedName")

.IsUnique()

.HasName("RoleNameIndex")

.HasFilter("[NormalizedName] IS NOT NULL");

b.ToTable("AspNetRoles");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityRoleClaim<string>", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("ClaimType")

.HasColumnType("nvarchar(max)");

b.Property<string>("ClaimValue")

.HasColumnType("nvarchar(max)");

b.Property<string>("RoleId")

.IsRequired()

.HasColumnType("nvarchar(450)");

b.HasKey("Id");

b.HasIndex("RoleId");

b.ToTable("AspNetRoleClaims");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserClaim<string>", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("ClaimType")

.HasColumnType("nvarchar(max)");

b.Property<string>("ClaimValue")

.HasColumnType("nvarchar(max)");

b.Property<string>("UserId")

.IsRequired()

.HasColumnType("nvarchar(450)");

b.HasKey("Id");

b.HasIndex("UserId");

b.ToTable("AspNetUserClaims");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserLogin<string>", b =>

{

b.Property<string>("LoginProvider")

.HasColumnType("nvarchar(128)")

.HasMaxLength(128);

b.Property<string>("ProviderKey")

.HasColumnType("nvarchar(128)")

.HasMaxLength(128);

b.Property<string>("ProviderDisplayName")

.HasColumnType("nvarchar(max)");

b.Property<string>("UserId")

.IsRequired()

.HasColumnType("nvarchar(450)");

b.HasKey("LoginProvider", "ProviderKey");

b.HasIndex("UserId");

b.ToTable("AspNetUserLogins");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserRole<string>", b =>

{

b.Property<string>("UserId")

.HasColumnType("nvarchar(450)");

b.Property<string>("RoleId")

.HasColumnType("nvarchar(450)");

b.HasKey("UserId", "RoleId");

b.HasIndex("RoleId");

b.ToTable("AspNetUserRoles");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserToken<string>", b =>

{

b.Property<string>("UserId")

.HasColumnType("nvarchar(450)");

b.Property<string>("LoginProvider")

.HasColumnType("nvarchar(128)")

.HasMaxLength(128);

b.Property<string>("Name")

.HasColumnType("nvarchar(128)")

.HasMaxLength(128);

b.Property<string>("Value")

.HasColumnType("nvarchar(max)");

b.HasKey("UserId", "LoginProvider", "Name");

b.ToTable("AspNetUserTokens");

});

modelBuilder.Entity("BlockTechMVC.Models.ContaCliente", b =>

{

b.HasOne("BlockTechMVC.Models.ApplicationUser", "ApplicationUser")

.WithMany()

.HasForeignKey("ApplicationUserID");

b.HasOne("BlockTechMVC.Models.Conta", "Conta")

.WithMany()

.HasForeignKey("ContaId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("BlockTechMVC.Models.CriptomoedaHoje", b =>

{

b.HasOne("BlockTechMVC.Models.Criptomoeda", "Criptomoeda")

.WithMany()

.HasForeignKey("CriptomoedaId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("BlockTechMVC.Models.Saldo", b =>

{

b.HasOne("BlockTechMVC.Models.CriptoSaldo", "CriptoSaldo")

.WithMany()

.HasForeignKey("CriptoSaldoId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("BlockTechMVC.Models.Transacao", b =>

{

b.HasOne("BlockTechMVC.Models.ContaCliente", "ContaCliente")

.WithMany()

.HasForeignKey("ContaClienteId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("BlockTechMVC.Models.CriptomoedaHoje", "CriptomoedaHoje")

.WithMany()

.HasForeignKey("CriptomoedaHojeId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("BlockTechMVC.Models.Saldo", null)

.WithMany("Transacao")

.HasForeignKey("SaldoId");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityRoleClaim<string>", b =>

{

b.HasOne("Microsoft.AspNetCore.Identity.IdentityRole", null)

.WithMany()

.HasForeignKey("RoleId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserClaim<string>", b =>

{

b.HasOne("BlockTechMVC.Models.ApplicationUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserLogin<string>", b =>

{

b.HasOne("BlockTechMVC.Models.ApplicationUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserRole<string>", b =>

{

b.HasOne("Microsoft.AspNetCore.Identity.IdentityRole", null)

.WithMany()

.HasForeignKey("RoleId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("BlockTechMVC.Models.ApplicationUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserToken<string>", b =>

{

b.HasOne("BlockTechMVC.Models.ApplicationUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

#pragma warning restore 612, 618

}

}

}

**Migrations - 20201022185953\_FK\_user\_conta.Designer**

// <auto-generated />

using System;

using BlockTechMVC.Data;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Infrastructure;

using Microsoft.EntityFrameworkCore.Metadata;

using Microsoft.EntityFrameworkCore.Migrations;

using Microsoft.EntityFrameworkCore.Storage.ValueConversion;

namespace BlockTechMVC.Migrations

{

[DbContext(typeof(ApplicationDbContext))]

[Migration("20201022185953\_FK\_user\_conta")]

partial class FK\_user\_conta

{

protected override void BuildTargetModel(ModelBuilder modelBuilder)

{

#pragma warning disable 612, 618

modelBuilder

.HasAnnotation("ProductVersion", "3.1.9")

.HasAnnotation("Relational:MaxIdentifierLength", 128)

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

modelBuilder.Entity("BlockTechMVC.Models.ApplicationUser", b =>

{

b.Property<string>("Id")

.HasColumnType("nvarchar(450)");

b.Property<int>("AccessFailedCount")

.HasColumnType("int");

b.Property<string>("Cep")

.HasColumnType("nvarchar(9)")

.HasMaxLength(9);

b.Property<string>("Cidade")

.HasColumnType("nvarchar(58)")

.HasMaxLength(58);

b.Property<string>("ConcurrencyStamp")

.IsConcurrencyToken()

.HasColumnType("nvarchar(max)");

b.Property<string>("Documento")

.HasColumnType("nvarchar(14)")

.HasMaxLength(14);

b.Property<string>("Email")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<bool>("EmailConfirmed")

.HasColumnType("bit");

b.Property<bool>("LockoutEnabled")

.HasColumnType("bit");

b.Property<DateTimeOffset?>("LockoutEnd")

.HasColumnType("datetimeoffset");

b.Property<string>("Nome")

.HasColumnType("nvarchar(60)")

.HasMaxLength(60);

b.Property<string>("NormalizedEmail")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<string>("NormalizedUserName")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<string>("Numero")

.HasColumnType("nvarchar(10)")

.HasMaxLength(10);

b.Property<string>("PasswordHash")

.HasColumnType("nvarchar(max)");

b.Property<string>("PhoneNumber")

.HasColumnType("nvarchar(max)");

b.Property<bool>("PhoneNumberConfirmed")

.HasColumnType("bit");

b.Property<string>("Rua")

.HasColumnType("nvarchar(max)");

b.Property<string>("SecurityStamp")

.HasColumnType("nvarchar(max)");

b.Property<string>("Telefone")

.HasColumnType("nvarchar(15)")

.HasMaxLength(15);

b.Property<bool>("TwoFactorEnabled")

.HasColumnType("bit");

b.Property<string>("Uf")

.HasColumnType("nvarchar(2)")

.HasMaxLength(2);

b.Property<string>("UserName")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.HasKey("Id");

b.HasIndex("NormalizedEmail")

.HasName("EmailIndex");

b.HasIndex("NormalizedUserName")

.IsUnique()

.HasName("UserNameIndex")

.HasFilter("[NormalizedUserName] IS NOT NULL");

b.ToTable("AspNetUsers");

});

modelBuilder.Entity("BlockTechMVC.Models.Conta", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("Agencia")

.HasColumnType("nvarchar(4)")

.HasMaxLength(4);

b.Property<string>("Banco")

.HasColumnType("nvarchar(max)");

b.Property<string>("NumeroConta")

.HasColumnType("nvarchar(max)");

b.Property<int>("TipoConta")

.HasColumnType("int");

b.HasKey("Id");

b.ToTable("Conta");

});

modelBuilder.Entity("BlockTechMVC.Models.ContaCliente", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("ApplicationUserID")

.HasColumnType("nvarchar(450)");

b.Property<int>("ContaId")

.HasColumnType("int");

b.Property<DateTime>("DataAbertura")

.HasColumnType("datetime2");

b.Property<string>("NumeroConta")

.HasColumnType("nvarchar(max)");

b.HasKey("Id");

b.HasIndex("ApplicationUserID");

b.HasIndex("ContaId");

b.ToTable("ContaCliente");

});

modelBuilder.Entity("BlockTechMVC.Models.CriptoSaldo", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("Criptomoeda")

.HasColumnType("nvarchar(max)");

b.Property<decimal>("Quantidade")

.HasColumnType("decimal(20,2)");

b.HasKey("Id");

b.ToTable("CriptoSaldo");

});

modelBuilder.Entity("BlockTechMVC.Models.Criptomoeda", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<DateTime>("Cadastro")

.HasColumnType("datetime2");

b.Property<string>("Nome")

.HasColumnType("nvarchar(max)");

b.Property<string>("Simbolo")

.HasColumnType("nvarchar(5)")

.HasMaxLength(5);

b.HasKey("Id");

b.ToTable("Criptomoeda");

});

modelBuilder.Entity("BlockTechMVC.Models.CriptomoedaHoje", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<int>("CriptomoedaId")

.HasColumnType("int");

b.Property<DateTime>("Data")

.HasColumnType("datetime2");

b.Property<decimal>("Valor")

.HasColumnType("decimal(20,2)");

b.HasKey("Id");

b.HasIndex("CriptomoedaId");

b.ToTable("CriptomoedaHoje");

});

modelBuilder.Entity("BlockTechMVC.Models.Saldo", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<int>("CriptoSaldoId")

.HasColumnType("int");

b.Property<decimal>("SaldoAtualRS")

.HasColumnType("decimal(20,2)");

b.Property<int>("TransacaoId")

.HasColumnType("int");

b.HasKey("Id");

b.HasIndex("CriptoSaldoId");

b.ToTable("Saldo");

});

modelBuilder.Entity("BlockTechMVC.Models.Transacao", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<int>("ContaClienteId")

.HasColumnType("int");

b.Property<int>("CriptomoedaHojeId")

.HasColumnType("int");

b.Property<DateTime>("Data")

.HasColumnType("datetime2");

b.Property<int?>("SaldoId")

.HasColumnType("int");

b.Property<int>("Tipo")

.HasColumnType("int");

b.Property<double>("Valor")

.HasColumnType("float");

b.HasKey("Id");

b.HasIndex("ContaClienteId");

b.HasIndex("CriptomoedaHojeId");

b.HasIndex("SaldoId");

b.ToTable("Transacao");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityRole", b =>

{

b.Property<string>("Id")

.HasColumnType("nvarchar(450)");

b.Property<string>("ConcurrencyStamp")

.IsConcurrencyToken()

.HasColumnType("nvarchar(max)");

b.Property<string>("Name")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<string>("NormalizedName")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.HasKey("Id");

b.HasIndex("NormalizedName")

.IsUnique()

.HasName("RoleNameIndex")

.HasFilter("[NormalizedName] IS NOT NULL");

b.ToTable("AspNetRoles");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityRoleClaim<string>", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("ClaimType")

.HasColumnType("nvarchar(max)");

b.Property<string>("ClaimValue")

.HasColumnType("nvarchar(max)");

b.Property<string>("RoleId")

.IsRequired()

.HasColumnType("nvarchar(450)");

b.HasKey("Id");

b.HasIndex("RoleId");

b.ToTable("AspNetRoleClaims");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserClaim<string>", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("ClaimType")

.HasColumnType("nvarchar(max)");

b.Property<string>("ClaimValue")

.HasColumnType("nvarchar(max)");

b.Property<string>("UserId")

.IsRequired()

.HasColumnType("nvarchar(450)");

b.HasKey("Id");

b.HasIndex("UserId");

b.ToTable("AspNetUserClaims");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserLogin<string>", b =>

{

b.Property<string>("LoginProvider")

.HasColumnType("nvarchar(128)")

.HasMaxLength(128);

b.Property<string>("ProviderKey")

.HasColumnType("nvarchar(128)")

.HasMaxLength(128);

b.Property<string>("ProviderDisplayName")

.HasColumnType("nvarchar(max)");

b.Property<string>("UserId")

.IsRequired()

.HasColumnType("nvarchar(450)");

b.HasKey("LoginProvider", "ProviderKey");

b.HasIndex("UserId");

b.ToTable("AspNetUserLogins");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserRole<string>", b =>

{

b.Property<string>("UserId")

.HasColumnType("nvarchar(450)");

b.Property<string>("RoleId")

.HasColumnType("nvarchar(450)");

b.HasKey("UserId", "RoleId");

b.HasIndex("RoleId");

b.ToTable("AspNetUserRoles");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserToken<string>", b =>

{

b.Property<string>("UserId")

.HasColumnType("nvarchar(450)");

b.Property<string>("LoginProvider")

.HasColumnType("nvarchar(128)")

.HasMaxLength(128);

b.Property<string>("Name")

.HasColumnType("nvarchar(128)")

.HasMaxLength(128);

b.Property<string>("Value")

.HasColumnType("nvarchar(max)");

b.HasKey("UserId", "LoginProvider", "Name");

b.ToTable("AspNetUserTokens");

});

modelBuilder.Entity("BlockTechMVC.Models.ContaCliente", b =>

{

b.HasOne("BlockTechMVC.Models.ApplicationUser", "ApplicationUser")

.WithMany()

.HasForeignKey("ApplicationUserID");

b.HasOne("BlockTechMVC.Models.Conta", "Conta")

.WithMany()

.HasForeignKey("ContaId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("BlockTechMVC.Models.CriptomoedaHoje", b =>

{

b.HasOne("BlockTechMVC.Models.Criptomoeda", "Criptomoeda")

.WithMany()

.HasForeignKey("CriptomoedaId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("BlockTechMVC.Models.Saldo", b =>

{

b.HasOne("BlockTechMVC.Models.CriptoSaldo", "CriptoSaldo")

.WithMany()

.HasForeignKey("CriptoSaldoId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("BlockTechMVC.Models.Transacao", b =>

{

b.HasOne("BlockTechMVC.Models.ContaCliente", "ContaCliente")

.WithMany()

.HasForeignKey("ContaClienteId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("BlockTechMVC.Models.CriptomoedaHoje", "CriptomoedaHoje")

.WithMany()

.HasForeignKey("CriptomoedaHojeId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("BlockTechMVC.Models.Saldo", null)

.WithMany("Transacao")

.HasForeignKey("SaldoId");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityRoleClaim<string>", b =>

{

b.HasOne("Microsoft.AspNetCore.Identity.IdentityRole", null)

.WithMany()

.HasForeignKey("RoleId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserClaim<string>", b =>

{

b.HasOne("BlockTechMVC.Models.ApplicationUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserLogin<string>", b =>

{

b.HasOne("BlockTechMVC.Models.ApplicationUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserRole<string>", b =>

{

b.HasOne("Microsoft.AspNetCore.Identity.IdentityRole", null)

.WithMany()

.HasForeignKey("RoleId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("BlockTechMVC.Models.ApplicationUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserToken<string>", b =>

{

b.HasOne("BlockTechMVC.Models.ApplicationUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

#pragma warning restore 612, 618

}

}

}

**Migrations - 20201021005607\_Criptomoeda-Saldo.Designer**

// <auto-generated />

using System;

using BlockTechMVC.Data;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Infrastructure;

using Microsoft.EntityFrameworkCore.Metadata;

using Microsoft.EntityFrameworkCore.Migrations;

using Microsoft.EntityFrameworkCore.Storage.ValueConversion;

namespace BlockTechMVC.Migrations

{

[DbContext(typeof(ApplicationDbContext))]

[Migration("20201021005607\_Criptomoeda-Saldo")]

partial class CriptomoedaSaldo

{

protected override void BuildTargetModel(ModelBuilder modelBuilder)

{

#pragma warning disable 612, 618

modelBuilder

.HasAnnotation("ProductVersion", "3.1.9")

.HasAnnotation("Relational:MaxIdentifierLength", 128)

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

modelBuilder.Entity("BlockTechMVC.Models.ApplicationUser", b =>

{

b.Property<string>("Id")

.HasColumnType("nvarchar(450)");

b.Property<int>("AccessFailedCount")

.HasColumnType("int");

b.Property<string>("Cep")

.HasColumnType("nvarchar(9)")

.HasMaxLength(9);

b.Property<string>("Cidade")

.HasColumnType("nvarchar(58)")

.HasMaxLength(58);

b.Property<string>("ConcurrencyStamp")

.IsConcurrencyToken()

.HasColumnType("nvarchar(max)");

b.Property<string>("Documento")

.HasColumnType("nvarchar(14)")

.HasMaxLength(14);

b.Property<string>("Email")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<bool>("EmailConfirmed")

.HasColumnType("bit");

b.Property<bool>("LockoutEnabled")

.HasColumnType("bit");

b.Property<DateTimeOffset?>("LockoutEnd")

.HasColumnType("datetimeoffset");

b.Property<string>("Nome")

.HasColumnType("nvarchar(60)")

.HasMaxLength(60);

b.Property<string>("NormalizedEmail")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<string>("NormalizedUserName")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<string>("Numero")

.HasColumnType("nvarchar(10)")

.HasMaxLength(10);

b.Property<string>("PasswordHash")

.HasColumnType("nvarchar(max)");

b.Property<string>("PhoneNumber")

.HasColumnType("nvarchar(max)");

b.Property<bool>("PhoneNumberConfirmed")

.HasColumnType("bit");

b.Property<string>("Rua")

.HasColumnType("nvarchar(max)");

b.Property<string>("SecurityStamp")

.HasColumnType("nvarchar(max)");

b.Property<string>("Telefone")

.HasColumnType("nvarchar(15)")

.HasMaxLength(15);

b.Property<bool>("TwoFactorEnabled")

.HasColumnType("bit");

b.Property<string>("Uf")

.HasColumnType("nvarchar(2)")

.HasMaxLength(2);

b.Property<string>("UserName")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.HasKey("Id");

b.HasIndex("NormalizedEmail")

.HasName("EmailIndex");

b.HasIndex("NormalizedUserName")

.IsUnique()

.HasName("UserNameIndex")

.HasFilter("[NormalizedUserName] IS NOT NULL");

b.ToTable("AspNetUsers");

});

modelBuilder.Entity("BlockTechMVC.Models.Conta", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("Agencia")

.HasColumnType("nvarchar(4)")

.HasMaxLength(4);

b.Property<string>("Banco")

.HasColumnType("nvarchar(max)");

b.Property<string>("NumeroConta")

.HasColumnType("nvarchar(max)");

b.Property<int>("TipoConta")

.HasColumnType("int");

b.HasKey("Id");

b.ToTable("Conta");

});

modelBuilder.Entity("BlockTechMVC.Models.ContaCliente", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("ApplicationUserId")

.HasColumnType("nvarchar(450)");

b.Property<int>("ContaId")

.HasColumnType("int");

b.Property<DateTime>("DataAbertura")

.HasColumnType("datetime2");

b.Property<string>("NumeroConta")

.HasColumnType("nvarchar(max)");

b.HasKey("Id");

b.HasIndex("ApplicationUserId");

b.HasIndex("ContaId");

b.ToTable("ContaCliente");

});

modelBuilder.Entity("BlockTechMVC.Models.CriptoSaldo", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("Criptomoeda")

.HasColumnType("nvarchar(50)");

b.Property<decimal>("Quantidade")

.HasColumnType("decimal(20,2)");

b.HasKey("Id");

b.ToTable("CriptoSaldo");

});

modelBuilder.Entity("BlockTechMVC.Models.Criptomoeda", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<DateTime>("Cadastro")

.HasColumnType("datetime2")

.HasMaxLength(20);

b.Property<string>("Nome")

.HasColumnType("nvarchar(max)");

b.Property<string>("Simbolo")

.HasColumnType("nvarchar(5)")

.HasMaxLength(5);

b.HasKey("Id");

b.ToTable("Criptomoeda");

});

modelBuilder.Entity("BlockTechMVC.Models.CriptomoedaHoje", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<int>("CriptomoedaId")

.HasColumnType("int");

b.Property<DateTime>("Data")

.HasColumnType("datetime2");

b.Property<decimal>("Valor")

.HasColumnType("decimal(20,2)");

b.HasKey("Id");

b.HasIndex("CriptomoedaId");

b.ToTable("CriptomoedaHoje");

});

modelBuilder.Entity("BlockTechMVC.Models.Saldo", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<int>("CriptoSaldoId")

.HasColumnType("int");

b.Property<decimal>("SaldoAtualRS")

.HasColumnType("decimal(20,2)");

b.Property<int>("TransacaoId")

.HasColumnType("int");

b.HasKey("Id");

b.HasIndex("CriptoSaldoId");

b.ToTable("Saldo");

});

modelBuilder.Entity("BlockTechMVC.Models.Transacao", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<int>("ContaClienteId")

.HasColumnType("int");

b.Property<int>("CriptomoedaHojeId")

.HasColumnType("int");

b.Property<DateTime>("Data")

.HasColumnType("datetime2");

b.Property<int?>("SaldoId")

.HasColumnType("int");

b.Property<int>("Tipo")

.HasColumnType("int");

b.Property<double>("Valor")

.HasColumnType("float");

b.HasKey("Id");

b.HasIndex("ContaClienteId");

b.HasIndex("CriptomoedaHojeId");

b.HasIndex("SaldoId");

b.ToTable("Transacao");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityRole", b =>

{

b.Property<string>("Id")

.HasColumnType("nvarchar(450)");

b.Property<string>("ConcurrencyStamp")

.IsConcurrencyToken()

.HasColumnType("nvarchar(max)");

b.Property<string>("Name")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<string>("NormalizedName")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.HasKey("Id");

b.HasIndex("NormalizedName")

.IsUnique()

.HasName("RoleNameIndex")

.HasFilter("[NormalizedName] IS NOT NULL");

b.ToTable("AspNetRoles");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityRoleClaim<string>", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("ClaimType")

.HasColumnType("nvarchar(max)");

b.Property<string>("ClaimValue")

.HasColumnType("nvarchar(max)");

b.Property<string>("RoleId")

.IsRequired()

.HasColumnType("nvarchar(450)");

b.HasKey("Id");

b.HasIndex("RoleId");

b.ToTable("AspNetRoleClaims");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserClaim<string>", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("ClaimType")

.HasColumnType("nvarchar(max)");

b.Property<string>("ClaimValue")

.HasColumnType("nvarchar(max)");

b.Property<string>("UserId")

.IsRequired()

.HasColumnType("nvarchar(450)");

b.HasKey("Id");

b.HasIndex("UserId");

b.ToTable("AspNetUserClaims");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserLogin<string>", b =>

{

b.Property<string>("LoginProvider")

.HasColumnType("nvarchar(128)")

.HasMaxLength(128);

b.Property<string>("ProviderKey")

.HasColumnType("nvarchar(128)")

.HasMaxLength(128);

b.Property<string>("ProviderDisplayName")

.HasColumnType("nvarchar(max)");

b.Property<string>("UserId")

.IsRequired()

.HasColumnType("nvarchar(450)");

b.HasKey("LoginProvider", "ProviderKey");

b.HasIndex("UserId");

b.ToTable("AspNetUserLogins");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserRole<string>", b =>

{

b.Property<string>("UserId")

.HasColumnType("nvarchar(450)");

b.Property<string>("RoleId")

.HasColumnType("nvarchar(450)");

b.HasKey("UserId", "RoleId");

b.HasIndex("RoleId");

b.ToTable("AspNetUserRoles");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserToken<string>", b =>

{

b.Property<string>("UserId")

.HasColumnType("nvarchar(450)");

b.Property<string>("LoginProvider")

.HasColumnType("nvarchar(128)")

.HasMaxLength(128);

b.Property<string>("Name")

.HasColumnType("nvarchar(128)")

.HasMaxLength(128);

b.Property<string>("Value")

.HasColumnType("nvarchar(max)");

b.HasKey("UserId", "LoginProvider", "Name");

b.ToTable("AspNetUserTokens");

});

modelBuilder.Entity("BlockTechMVC.Models.ContaCliente", b =>

{

b.HasOne("BlockTechMVC.Models.ApplicationUser", "ApplicationUser")

.WithMany()

.HasForeignKey("ApplicationUserId");

b.HasOne("BlockTechMVC.Models.Conta", "Conta")

.WithMany()

.HasForeignKey("ContaId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("BlockTechMVC.Models.CriptomoedaHoje", b =>

{

b.HasOne("BlockTechMVC.Models.Criptomoeda", "Criptomoeda")

.WithMany()

.HasForeignKey("CriptomoedaId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("BlockTechMVC.Models.Saldo", b =>

{

b.HasOne("BlockTechMVC.Models.CriptoSaldo", "CriptoSaldo")

.WithMany()

.HasForeignKey("CriptoSaldoId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("BlockTechMVC.Models.Transacao", b =>

{

b.HasOne("BlockTechMVC.Models.ContaCliente", "ContaCliente")

.WithMany()

.HasForeignKey("ContaClienteId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("BlockTechMVC.Models.CriptomoedaHoje", "CriptomoedaHoje")

.WithMany()

.HasForeignKey("CriptomoedaHojeId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("BlockTechMVC.Models.Saldo", null)

.WithMany("Transacao")

.HasForeignKey("SaldoId");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityRoleClaim<string>", b =>

{

b.HasOne("Microsoft.AspNetCore.Identity.IdentityRole", null)

.WithMany()

.HasForeignKey("RoleId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserClaim<string>", b =>

{

b.HasOne("BlockTechMVC.Models.ApplicationUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserLogin<string>", b =>

{

b.HasOne("BlockTechMVC.Models.ApplicationUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserRole<string>", b =>

{

b.HasOne("Microsoft.AspNetCore.Identity.IdentityRole", null)

.WithMany()

.HasForeignKey("RoleId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("BlockTechMVC.Models.ApplicationUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserToken<string>", b =>

{

b.HasOne("BlockTechMVC.Models.ApplicationUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

#pragma warning restore 612, 618

}

}

}

**Migrations - 20201019172645\_MaxLength.Designer**

// <auto-generated />

using System;

using BlockTechMVC.Data;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Infrastructure;

using Microsoft.EntityFrameworkCore.Metadata;

using Microsoft.EntityFrameworkCore.Migrations;

using Microsoft.EntityFrameworkCore.Storage.ValueConversion;

namespace BlockTechMVC.Migrations

{

[DbContext(typeof(ApplicationDbContext))]

[Migration("20201019172645\_MaxLength")]

partial class MaxLength

{

protected override void BuildTargetModel(ModelBuilder modelBuilder)

{

#pragma warning disable 612, 618

modelBuilder

.HasAnnotation("ProductVersion", "3.1.9")

.HasAnnotation("Relational:MaxIdentifierLength", 128)

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

modelBuilder.Entity("BlockTechMVC.Models.ApplicationUser", b =>

{

b.Property<string>("Id")

.HasColumnType("nvarchar(450)");

b.Property<int>("AccessFailedCount")

.HasColumnType("int");

b.Property<string>("Cep")

.HasColumnType("nvarchar(9)")

.HasMaxLength(9);

b.Property<string>("Cidade")

.HasColumnType("nvarchar(58)")

.HasMaxLength(58);

b.Property<string>("ConcurrencyStamp")

.IsConcurrencyToken()

.HasColumnType("nvarchar(max)");

b.Property<string>("Documento")

.HasColumnType("nvarchar(14)")

.HasMaxLength(14);

b.Property<string>("Email")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<bool>("EmailConfirmed")

.HasColumnType("bit");

b.Property<bool>("LockoutEnabled")

.HasColumnType("bit");

b.Property<DateTimeOffset?>("LockoutEnd")

.HasColumnType("datetimeoffset");

b.Property<string>("Nome")

.HasColumnType("nvarchar(60)")

.HasMaxLength(60);

b.Property<string>("NormalizedEmail")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<string>("NormalizedUserName")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<string>("Numero")

.HasColumnType("nvarchar(10)")

.HasMaxLength(10);

b.Property<string>("PasswordHash")

.HasColumnType("nvarchar(max)");

b.Property<string>("PhoneNumber")

.HasColumnType("nvarchar(max)");

b.Property<bool>("PhoneNumberConfirmed")

.HasColumnType("bit");

b.Property<string>("Rua")

.HasColumnType("nvarchar(max)");

b.Property<string>("SecurityStamp")

.HasColumnType("nvarchar(max)");

b.Property<string>("Telefone")

.HasColumnType("nvarchar(15)")

.HasMaxLength(15);

b.Property<bool>("TwoFactorEnabled")

.HasColumnType("bit");

b.Property<string>("Uf")

.HasColumnType("nvarchar(2)")

.HasMaxLength(2);

b.Property<string>("UserName")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.HasKey("Id");

b.HasIndex("NormalizedEmail")

.HasName("EmailIndex");

b.HasIndex("NormalizedUserName")

.IsUnique()

.HasName("UserNameIndex")

.HasFilter("[NormalizedUserName] IS NOT NULL");

b.ToTable("AspNetUsers");

});

modelBuilder.Entity("BlockTechMVC.Models.Conta", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("Agencia")

.HasColumnType("nvarchar(4)")

.HasMaxLength(4);

b.Property<string>("Banco")

.HasColumnType("nvarchar(max)");

b.Property<string>("NumeroConta")

.HasColumnType("nvarchar(max)");

b.Property<int>("TipoConta")

.HasColumnType("int");

b.HasKey("Id");

b.ToTable("Conta");

});

modelBuilder.Entity("BlockTechMVC.Models.ContaCliente", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("ApplicationUserId")

.HasColumnType("nvarchar(450)");

b.Property<int>("ContaId")

.HasColumnType("int");

b.Property<DateTime>("DataAbertura")

.HasColumnType("datetime2");

b.Property<string>("NumeroConta")

.HasColumnType("nvarchar(max)");

b.HasKey("Id");

b.HasIndex("ApplicationUserId");

b.HasIndex("ContaId");

b.ToTable("ContaCliente");

});

modelBuilder.Entity("BlockTechMVC.Models.Criptomoeda", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<DateTime>("Cadastro")

.HasColumnType("datetime2")

.HasMaxLength(20);

b.Property<string>("Nome")

.HasColumnType("nvarchar(max)");

b.Property<string>("Simbolo")

.HasColumnType("nvarchar(5)")

.HasMaxLength(5);

b.HasKey("Id");

b.ToTable("Criptomoeda");

});

modelBuilder.Entity("BlockTechMVC.Models.CriptomoedaHoje", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<int>("CriptomoedaId")

.HasColumnType("int");

b.Property<DateTime>("Data")

.HasColumnType("datetime2");

b.Property<decimal>("Valor")

.HasColumnType("decimal(20,2)");

b.HasKey("Id");

b.HasIndex("CriptomoedaId");

b.ToTable("CriptomoedaHoje");

});

modelBuilder.Entity("BlockTechMVC.Models.Saldo", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<decimal>("SaldoAtualRS")

.HasColumnType("decimal(20,2)");

b.Property<int>("TransacaoId")

.HasColumnType("int");

b.Property<decimal>("quantidadeCripo")

.HasColumnType("decimal(20,2)");

b.HasKey("Id");

b.ToTable("Saldo");

});

modelBuilder.Entity("BlockTechMVC.Models.Transacao", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<int>("ContaClienteId")

.HasColumnType("int");

b.Property<int>("CriptomoedaHojeId")

.HasColumnType("int");

b.Property<DateTime>("Data")

.HasColumnType("datetime2");

b.Property<int?>("SaldoId")

.HasColumnType("int");

b.Property<int>("Tipo")

.HasColumnType("int");

b.Property<double>("Valor")

.HasColumnType("float");

b.HasKey("Id");

b.HasIndex("ContaClienteId");

b.HasIndex("CriptomoedaHojeId");

b.HasIndex("SaldoId");

b.ToTable("Transacao");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityRole", b =>

{

b.Property<string>("Id")

.HasColumnType("nvarchar(450)");

b.Property<string>("ConcurrencyStamp")

.IsConcurrencyToken()

.HasColumnType("nvarchar(max)");

b.Property<string>("Name")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<string>("NormalizedName")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.HasKey("Id");

b.HasIndex("NormalizedName")

.IsUnique()

.HasName("RoleNameIndex")

.HasFilter("[NormalizedName] IS NOT NULL");

b.ToTable("AspNetRoles");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityRoleClaim<string>", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("ClaimType")

.HasColumnType("nvarchar(max)");

b.Property<string>("ClaimValue")

.HasColumnType("nvarchar(max)");

b.Property<string>("RoleId")

.IsRequired()

.HasColumnType("nvarchar(450)");

b.HasKey("Id");

b.HasIndex("RoleId");

b.ToTable("AspNetRoleClaims");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserClaim<string>", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("ClaimType")

.HasColumnType("nvarchar(max)");

b.Property<string>("ClaimValue")

.HasColumnType("nvarchar(max)");

b.Property<string>("UserId")

.IsRequired()

.HasColumnType("nvarchar(450)");

b.HasKey("Id");

b.HasIndex("UserId");

b.ToTable("AspNetUserClaims");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserLogin<string>", b =>

{

b.Property<string>("LoginProvider")

.HasColumnType("nvarchar(128)")

.HasMaxLength(128);

b.Property<string>("ProviderKey")

.HasColumnType("nvarchar(128)")

.HasMaxLength(128);

b.Property<string>("ProviderDisplayName")

.HasColumnType("nvarchar(max)");

b.Property<string>("UserId")

.IsRequired()

.HasColumnType("nvarchar(450)");

b.HasKey("LoginProvider", "ProviderKey");

b.HasIndex("UserId");

b.ToTable("AspNetUserLogins");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserRole<string>", b =>

{

b.Property<string>("UserId")

.HasColumnType("nvarchar(450)");

b.Property<string>("RoleId")

.HasColumnType("nvarchar(450)");

b.HasKey("UserId", "RoleId");

b.HasIndex("RoleId");

b.ToTable("AspNetUserRoles");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserToken<string>", b =>

{

b.Property<string>("UserId")

.HasColumnType("nvarchar(450)");

b.Property<string>("LoginProvider")

.HasColumnType("nvarchar(128)")

.HasMaxLength(128);

b.Property<string>("Name")

.HasColumnType("nvarchar(128)")

.HasMaxLength(128);

b.Property<string>("Value")

.HasColumnType("nvarchar(max)");

b.HasKey("UserId", "LoginProvider", "Name");

b.ToTable("AspNetUserTokens");

});

modelBuilder.Entity("BlockTechMVC.Models.ContaCliente", b =>

{

b.HasOne("BlockTechMVC.Models.ApplicationUser", "ApplicationUser")

.WithMany()

.HasForeignKey("ApplicationUserId");

b.HasOne("BlockTechMVC.Models.Conta", "Conta")

.WithMany()

.HasForeignKey("ContaId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("BlockTechMVC.Models.CriptomoedaHoje", b =>

{

b.HasOne("BlockTechMVC.Models.Criptomoeda", "Criptomoeda")

.WithMany()

.HasForeignKey("CriptomoedaId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("BlockTechMVC.Models.Transacao", b =>

{

b.HasOne("BlockTechMVC.Models.ContaCliente", "ContaCliente")

.WithMany()

.HasForeignKey("ContaClienteId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("BlockTechMVC.Models.CriptomoedaHoje", "CriptomoedaHoje")

.WithMany()

.HasForeignKey("CriptomoedaHojeId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("BlockTechMVC.Models.Saldo", null)

.WithMany("Transacao")

.HasForeignKey("SaldoId");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityRoleClaim<string>", b =>

{

b.HasOne("Microsoft.AspNetCore.Identity.IdentityRole", null)

.WithMany()

.HasForeignKey("RoleId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserClaim<string>", b =>

{

b.HasOne("BlockTechMVC.Models.ApplicationUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserLogin<string>", b =>

{

b.HasOne("BlockTechMVC.Models.ApplicationUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserRole<string>", b =>

{

b.HasOne("Microsoft.AspNetCore.Identity.IdentityRole", null)

.WithMany()

.HasForeignKey("RoleId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("BlockTechMVC.Models.ApplicationUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserToken<string>", b =>

{

b.HasOne("BlockTechMVC.Models.ApplicationUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

#pragma warning restore 612, 618

}

}

}

**Migrations - 20201019121451\_AtlzIdntt.Designer**

// <auto-generated />

using System;

using BlockTechMVC.Data;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Infrastructure;

using Microsoft.EntityFrameworkCore.Metadata;

using Microsoft.EntityFrameworkCore.Migrations;

using Microsoft.EntityFrameworkCore.Storage.ValueConversion;

namespace BlockTechMVC.Migrations

{

[DbContext(typeof(ApplicationDbContext))]

[Migration("20201019121451\_AtlzIdntt")]

partial class AtlzIdntt

{

protected override void BuildTargetModel(ModelBuilder modelBuilder)

{

#pragma warning disable 612, 618

modelBuilder

.HasAnnotation("ProductVersion", "3.1.9")

.HasAnnotation("Relational:MaxIdentifierLength", 128)

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

modelBuilder.Entity("BlockTechMVC.Models.ApplicationUser", b =>

{

b.Property<string>("Id")

.HasColumnType("nvarchar(450)");

b.Property<int>("AccessFailedCount")

.HasColumnType("int");

b.Property<string>("Cep")

.HasColumnType("nvarchar(max)");

b.Property<string>("Cidade")

.HasColumnType("nvarchar(max)");

b.Property<string>("ConcurrencyStamp")

.IsConcurrencyToken()

.HasColumnType("nvarchar(max)");

b.Property<string>("Documento")

.HasColumnType("nvarchar(max)");

b.Property<string>("Email")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<bool>("EmailConfirmed")

.HasColumnType("bit");

b.Property<bool>("LockoutEnabled")

.HasColumnType("bit");

b.Property<DateTimeOffset?>("LockoutEnd")

.HasColumnType("datetimeoffset");

b.Property<string>("Nome")

.HasColumnType("nvarchar(max)");

b.Property<string>("NormalizedEmail")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<string>("NormalizedUserName")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<string>("Numero")

.HasColumnType("nvarchar(max)");

b.Property<string>("PasswordHash")

.HasColumnType("nvarchar(max)");

b.Property<string>("PhoneNumber")

.HasColumnType("nvarchar(max)");

b.Property<bool>("PhoneNumberConfirmed")

.HasColumnType("bit");

b.Property<string>("Rua")

.HasColumnType("nvarchar(max)");

b.Property<string>("SecurityStamp")

.HasColumnType("nvarchar(max)");

b.Property<string>("Telefone")

.HasColumnType("nvarchar(max)");

b.Property<bool>("TwoFactorEnabled")

.HasColumnType("bit");

b.Property<string>("Uf")

.HasColumnType("nvarchar(max)");

b.Property<string>("UserName")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.HasKey("Id");

b.HasIndex("NormalizedEmail")

.HasName("EmailIndex");

b.HasIndex("NormalizedUserName")

.IsUnique()

.HasName("UserNameIndex")

.HasFilter("[NormalizedUserName] IS NOT NULL");

b.ToTable("AspNetUsers");

});

modelBuilder.Entity("BlockTechMVC.Models.Conta", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("Agencia")

.HasColumnType("nvarchar(max)");

b.Property<string>("Banco")

.HasColumnType("nvarchar(max)");

b.Property<string>("NumeroConta")

.HasColumnType("nvarchar(max)");

b.Property<int>("TipoConta")

.HasColumnType("int");

b.HasKey("Id");

b.ToTable("Conta");

});

modelBuilder.Entity("BlockTechMVC.Models.ContaCliente", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("ApplicationUserId")

.HasColumnType("nvarchar(450)");

b.Property<int>("ContaId")

.HasColumnType("int");

b.Property<DateTime>("DataAbertura")

.HasColumnType("datetime2");

b.Property<string>("NumeroConta")

.HasColumnType("nvarchar(max)");

b.HasKey("Id");

b.HasIndex("ApplicationUserId");

b.HasIndex("ContaId");

b.ToTable("ContaCliente");

});

modelBuilder.Entity("BlockTechMVC.Models.Criptomoeda", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<DateTime>("Cadastro")

.HasColumnType("datetime2");

b.Property<string>("Nome")

.HasColumnType("nvarchar(max)");

b.Property<string>("Simbolo")

.HasColumnType("nvarchar(max)");

b.HasKey("Id");

b.ToTable("Criptomoeda");

});

modelBuilder.Entity("BlockTechMVC.Models.CriptomoedaHoje", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<int>("CriptomoedaId")

.HasColumnType("int");

b.Property<DateTime>("Data")

.HasColumnType("datetime2");

b.Property<decimal>("Valor")

.HasColumnType("decimal(20,2)");

b.HasKey("Id");

b.HasIndex("CriptomoedaId");

b.ToTable("CriptomoedaHoje");

});

modelBuilder.Entity("BlockTechMVC.Models.Saldo", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<decimal>("SaldoAtualRS")

.HasColumnType("decimal(20,2)");

b.Property<int>("TransacaoId")

.HasColumnType("int");

b.Property<decimal>("quantidadeCripo")

.HasColumnType("decimal(20,2)");

b.HasKey("Id");

b.ToTable("Saldo");

});

modelBuilder.Entity("BlockTechMVC.Models.Transacao", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<int>("ContaClienteId")

.HasColumnType("int");

b.Property<int>("CriptomoedaHojeId")

.HasColumnType("int");

b.Property<DateTime>("Data")

.HasColumnType("datetime2");

b.Property<int?>("SaldoId")

.HasColumnType("int");

b.Property<int>("Tipo")

.HasColumnType("int");

b.Property<double>("Valor")

.HasColumnType("float");

b.HasKey("Id");

b.HasIndex("ContaClienteId");

b.HasIndex("CriptomoedaHojeId");

b.HasIndex("SaldoId");

b.ToTable("Transacao");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityRole", b =>

{

b.Property<string>("Id")

.HasColumnType("nvarchar(450)");

b.Property<string>("ConcurrencyStamp")

.IsConcurrencyToken()

.HasColumnType("nvarchar(max)");

b.Property<string>("Name")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<string>("NormalizedName")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.HasKey("Id");

b.HasIndex("NormalizedName")

.IsUnique()

.HasName("RoleNameIndex")

.HasFilter("[NormalizedName] IS NOT NULL");

b.ToTable("AspNetRoles");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityRoleClaim<string>", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("ClaimType")

.HasColumnType("nvarchar(max)");

b.Property<string>("ClaimValue")

.HasColumnType("nvarchar(max)");

b.Property<string>("RoleId")

.IsRequired()

.HasColumnType("nvarchar(450)");

b.HasKey("Id");

b.HasIndex("RoleId");

b.ToTable("AspNetRoleClaims");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserClaim<string>", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("ClaimType")

.HasColumnType("nvarchar(max)");

b.Property<string>("ClaimValue")

.HasColumnType("nvarchar(max)");

b.Property<string>("UserId")

.IsRequired()

.HasColumnType("nvarchar(450)");

b.HasKey("Id");

b.HasIndex("UserId");

b.ToTable("AspNetUserClaims");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserLogin<string>", b =>

{

b.Property<string>("LoginProvider")

.HasColumnType("nvarchar(128)")

.HasMaxLength(128);

b.Property<string>("ProviderKey")

.HasColumnType("nvarchar(128)")

.HasMaxLength(128);

b.Property<string>("ProviderDisplayName")

.HasColumnType("nvarchar(max)");

b.Property<string>("UserId")

.IsRequired()

.HasColumnType("nvarchar(450)");

b.HasKey("LoginProvider", "ProviderKey");

b.HasIndex("UserId");

b.ToTable("AspNetUserLogins");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserRole<string>", b =>

{

b.Property<string>("UserId")

.HasColumnType("nvarchar(450)");

b.Property<string>("RoleId")

.HasColumnType("nvarchar(450)");

b.HasKey("UserId", "RoleId");

b.HasIndex("RoleId");

b.ToTable("AspNetUserRoles");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserToken<string>", b =>

{

b.Property<string>("UserId")

.HasColumnType("nvarchar(450)");

b.Property<string>("LoginProvider")

.HasColumnType("nvarchar(128)")

.HasMaxLength(128);

b.Property<string>("Name")

.HasColumnType("nvarchar(128)")

.HasMaxLength(128);

b.Property<string>("Value")

.HasColumnType("nvarchar(max)");

b.HasKey("UserId", "LoginProvider", "Name");

b.ToTable("AspNetUserTokens");

});

modelBuilder.Entity("BlockTechMVC.Models.ContaCliente", b =>

{

b.HasOne("BlockTechMVC.Models.ApplicationUser", "ApplicationUser")

.WithMany()

.HasForeignKey("ApplicationUserId");

b.HasOne("BlockTechMVC.Models.Conta", "Conta")

.WithMany()

.HasForeignKey("ContaId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("BlockTechMVC.Models.CriptomoedaHoje", b =>

{

b.HasOne("BlockTechMVC.Models.Criptomoeda", "Criptomoeda")

.WithMany()

.HasForeignKey("CriptomoedaId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("BlockTechMVC.Models.Transacao", b =>

{

b.HasOne("BlockTechMVC.Models.ContaCliente", "ContaCliente")

.WithMany()

.HasForeignKey("ContaClienteId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("BlockTechMVC.Models.CriptomoedaHoje", "CriptomoedaHoje")

.WithMany()

.HasForeignKey("CriptomoedaHojeId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("BlockTechMVC.Models.Saldo", null)

.WithMany("Transacao")

.HasForeignKey("SaldoId");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityRoleClaim<string>", b =>

{

b.HasOne("Microsoft.AspNetCore.Identity.IdentityRole", null)

.WithMany()

.HasForeignKey("RoleId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserClaim<string>", b =>

{

b.HasOne("BlockTechMVC.Models.ApplicationUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserLogin<string>", b =>

{

b.HasOne("BlockTechMVC.Models.ApplicationUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserRole<string>", b =>

{

b.HasOne("Microsoft.AspNetCore.Identity.IdentityRole", null)

.WithMany()

.HasForeignKey("RoleId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("BlockTechMVC.Models.ApplicationUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserToken<string>", b =>

{

b.HasOne("BlockTechMVC.Models.ApplicationUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

#pragma warning restore 612, 618

}

}

}

**Migrations - 20201019112957\_Atl.Designer**

// <auto-generated />

using System;

using BlockTechMVC.Data;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Infrastructure;

using Microsoft.EntityFrameworkCore.Metadata;

using Microsoft.EntityFrameworkCore.Migrations;

using Microsoft.EntityFrameworkCore.Storage.ValueConversion;

namespace BlockTechMVC.Migrations

{

[DbContext(typeof(ApplicationDbContext))]

[Migration("20201019112957\_Atl")]

partial class Atl

{

protected override void BuildTargetModel(ModelBuilder modelBuilder)

{

#pragma warning disable 612, 618

modelBuilder

.HasAnnotation("ProductVersion", "3.1.9")

.HasAnnotation("Relational:MaxIdentifierLength", 128)

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

modelBuilder.Entity("BlockTechMVC.Models.ApplicationUser", b =>

{

b.Property<string>("Id")

.HasColumnType("nvarchar(450)");

b.Property<int>("AccessFailedCount")

.HasColumnType("int");

b.Property<string>("Cep")

.HasColumnType("nvarchar(max)");

b.Property<string>("Cidade")

.HasColumnType("nvarchar(max)");

b.Property<string>("ConcurrencyStamp")

.IsConcurrencyToken()

.HasColumnType("nvarchar(max)");

b.Property<string>("Documento")

.HasColumnType("nvarchar(max)");

b.Property<string>("Email")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<bool>("EmailConfirmed")

.HasColumnType("bit");

b.Property<bool>("LockoutEnabled")

.HasColumnType("bit");

b.Property<DateTimeOffset?>("LockoutEnd")

.HasColumnType("datetimeoffset");

b.Property<string>("Nome")

.HasColumnType("nvarchar(max)");

b.Property<string>("NormalizedEmail")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<string>("NormalizedUserName")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<string>("Numero")

.HasColumnType("nvarchar(max)");

b.Property<string>("PasswordHash")

.HasColumnType("nvarchar(max)");

b.Property<string>("PhoneNumber")

.HasColumnType("nvarchar(max)");

b.Property<bool>("PhoneNumberConfirmed")

.HasColumnType("bit");

b.Property<string>("Rua")

.HasColumnType("nvarchar(max)");

b.Property<string>("SecurityStamp")

.HasColumnType("nvarchar(max)");

b.Property<string>("Telefone")

.HasColumnType("nvarchar(max)");

b.Property<bool>("TwoFactorEnabled")

.HasColumnType("bit");

b.Property<string>("Uf")

.HasColumnType("nvarchar(max)");

b.Property<string>("UserName")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.HasKey("Id");

b.HasIndex("NormalizedEmail")

.HasName("EmailIndex");

b.HasIndex("NormalizedUserName")

.IsUnique()

.HasName("UserNameIndex")

.HasFilter("[NormalizedUserName] IS NOT NULL");

b.ToTable("AspNetUsers");

});

modelBuilder.Entity("BlockTechMVC.Models.Conta", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("Agencia")

.HasColumnType("nvarchar(max)");

b.Property<string>("Banco")

.HasColumnType("nvarchar(max)");

b.Property<string>("NumeroConta")

.HasColumnType("nvarchar(max)");

b.Property<int>("TipoConta")

.HasColumnType("int");

b.HasKey("Id");

b.ToTable("Conta");

});

modelBuilder.Entity("BlockTechMVC.Models.ContaCliente", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("ApplicationUserId")

.HasColumnType("nvarchar(450)");

b.Property<int>("ContaId")

.HasColumnType("int");

b.Property<DateTime>("DataAbertura")

.HasColumnType("datetime2");

b.Property<string>("NumeroConta")

.HasColumnType("nvarchar(max)");

b.HasKey("Id");

b.HasIndex("ApplicationUserId");

b.HasIndex("ContaId");

b.ToTable("ContaCliente");

});

modelBuilder.Entity("BlockTechMVC.Models.Criptomoeda", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<DateTime>("Cadastro")

.HasColumnType("datetime2");

b.Property<string>("Nome")

.HasColumnType("nvarchar(max)");

b.Property<string>("Simbolo")

.HasColumnType("nvarchar(max)");

b.HasKey("Id");

b.ToTable("Criptomoeda");

});

modelBuilder.Entity("BlockTechMVC.Models.CriptomoedaHoje", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<int>("CriptomoedaId")

.HasColumnType("int");

b.Property<DateTime>("Data")

.HasColumnType("datetime2");

b.Property<decimal>("Valor")

.HasColumnType("decimal(20,2)");

b.HasKey("Id");

b.HasIndex("CriptomoedaId");

b.ToTable("CriptomoedaHoje");

});

modelBuilder.Entity("BlockTechMVC.Models.Saldo", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<decimal>("SaldoAtualRS")

.HasColumnType("decimal(20,2)");

b.Property<int>("TransacaoId")

.HasColumnType("int");

b.Property<decimal>("quantidadeCripo")

.HasColumnType("decimal(20,2)");

b.HasKey("Id");

b.ToTable("Saldo");

});

modelBuilder.Entity("BlockTechMVC.Models.Transacao", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<int>("ContaClienteId")

.HasColumnType("int");

b.Property<int>("CriptomoedaHojeId")

.HasColumnType("int");

b.Property<DateTime>("Data")

.HasColumnType("datetime2");

b.Property<int?>("SaldoId")

.HasColumnType("int");

b.Property<int>("Tipo")

.HasColumnType("int");

b.Property<double>("Valor")

.HasColumnType("float");

b.HasKey("Id");

b.HasIndex("ContaClienteId");

b.HasIndex("CriptomoedaHojeId");

b.HasIndex("SaldoId");

b.ToTable("Transacao");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityRole", b =>

{

b.Property<string>("Id")

.HasColumnType("nvarchar(450)");

b.Property<string>("ConcurrencyStamp")

.IsConcurrencyToken()

.HasColumnType("nvarchar(max)");

b.Property<string>("Name")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<string>("NormalizedName")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.HasKey("Id");

b.HasIndex("NormalizedName")

.IsUnique()

.HasName("RoleNameIndex")

.HasFilter("[NormalizedName] IS NOT NULL");

b.ToTable("AspNetRoles");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityRoleClaim<string>", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("ClaimType")

.HasColumnType("nvarchar(max)");

b.Property<string>("ClaimValue")

.HasColumnType("nvarchar(max)");

b.Property<string>("RoleId")

.IsRequired()

.HasColumnType("nvarchar(450)");

b.HasKey("Id");

b.HasIndex("RoleId");

b.ToTable("AspNetRoleClaims");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserClaim<string>", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("ClaimType")

.HasColumnType("nvarchar(max)");

b.Property<string>("ClaimValue")

.HasColumnType("nvarchar(max)");

b.Property<string>("UserId")

.IsRequired()

.HasColumnType("nvarchar(450)");

b.HasKey("Id");

b.HasIndex("UserId");

b.ToTable("AspNetUserClaims");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserLogin<string>", b =>

{

b.Property<string>("LoginProvider")

.HasColumnType("nvarchar(128)")

.HasMaxLength(128);

b.Property<string>("ProviderKey")

.HasColumnType("nvarchar(128)")

.HasMaxLength(128);

b.Property<string>("ProviderDisplayName")

.HasColumnType("nvarchar(max)");

b.Property<string>("UserId")

.IsRequired()

.HasColumnType("nvarchar(450)");

b.HasKey("LoginProvider", "ProviderKey");

b.HasIndex("UserId");

b.ToTable("AspNetUserLogins");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserRole<string>", b =>

{

b.Property<string>("UserId")

.HasColumnType("nvarchar(450)");

b.Property<string>("RoleId")

.HasColumnType("nvarchar(450)");

b.HasKey("UserId", "RoleId");

b.HasIndex("RoleId");

b.ToTable("AspNetUserRoles");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserToken<string>", b =>

{

b.Property<string>("UserId")

.HasColumnType("nvarchar(450)");

b.Property<string>("LoginProvider")

.HasColumnType("nvarchar(128)")

.HasMaxLength(128);

b.Property<string>("Name")

.HasColumnType("nvarchar(128)")

.HasMaxLength(128);

b.Property<string>("Value")

.HasColumnType("nvarchar(max)");

b.HasKey("UserId", "LoginProvider", "Name");

b.ToTable("AspNetUserTokens");

});

modelBuilder.Entity("BlockTechMVC.Models.ContaCliente", b =>

{

b.HasOne("BlockTechMVC.Models.ApplicationUser", "ApplicationUser")

.WithMany()

.HasForeignKey("ApplicationUserId");

b.HasOne("BlockTechMVC.Models.Conta", "Conta")

.WithMany()

.HasForeignKey("ContaId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("BlockTechMVC.Models.CriptomoedaHoje", b =>

{

b.HasOne("BlockTechMVC.Models.Criptomoeda", "Criptomoeda")

.WithMany()

.HasForeignKey("CriptomoedaId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("BlockTechMVC.Models.Transacao", b =>

{

b.HasOne("BlockTechMVC.Models.ContaCliente", "ContaCliente")

.WithMany()

.HasForeignKey("ContaClienteId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("BlockTechMVC.Models.CriptomoedaHoje", "CriptomoedaHoje")

.WithMany()

.HasForeignKey("CriptomoedaHojeId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("BlockTechMVC.Models.Saldo", null)

.WithMany("Transacao")

.HasForeignKey("SaldoId");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityRoleClaim<string>", b =>

{

b.HasOne("Microsoft.AspNetCore.Identity.IdentityRole", null)

.WithMany()

.HasForeignKey("RoleId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserClaim<string>", b =>

{

b.HasOne("BlockTechMVC.Models.ApplicationUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserLogin<string>", b =>

{

b.HasOne("BlockTechMVC.Models.ApplicationUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserRole<string>", b =>

{

b.HasOne("Microsoft.AspNetCore.Identity.IdentityRole", null)

.WithMany()

.HasForeignKey("RoleId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("BlockTechMVC.Models.ApplicationUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserToken<string>", b =>

{

b.HasOne("BlockTechMVC.Models.ApplicationUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

#pragma warning restore 612, 618

}

}

}

**Migrations - 20201018182900\_BD.Designer**

// <auto-generated />

using System;

using BlockTechMVC.Data;

using Microsoft.EntityFrameworkCore;

using Microsoft.EntityFrameworkCore.Infrastructure;

using Microsoft.EntityFrameworkCore.Metadata;

using Microsoft.EntityFrameworkCore.Migrations;

using Microsoft.EntityFrameworkCore.Storage.ValueConversion;

namespace BlockTechMVC.Migrations

{

[DbContext(typeof(ApplicationDbContext))]

[Migration("20201018182900\_BD")]

partial class BD

{

protected override void BuildTargetModel(ModelBuilder modelBuilder)

{

#pragma warning disable 612, 618

modelBuilder

.HasAnnotation("ProductVersion", "3.1.9")

.HasAnnotation("Relational:MaxIdentifierLength", 128)

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

modelBuilder.Entity("BlockTechMVC.Models.ApplicationUser", b =>

{

b.Property<string>("Id")

.HasColumnType("nvarchar(450)");

b.Property<int>("AccessFailedCount")

.HasColumnType("int");

b.Property<string>("Cep")

.HasColumnType("nvarchar(max)");

b.Property<string>("Cidade")

.HasColumnType("nvarchar(max)");

b.Property<string>("ConcurrencyStamp")

.IsConcurrencyToken()

.HasColumnType("nvarchar(max)");

b.Property<string>("Documento")

.HasColumnType("nvarchar(max)");

b.Property<string>("Email")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<bool>("EmailConfirmed")

.HasColumnType("bit");

b.Property<bool>("LockoutEnabled")

.HasColumnType("bit");

b.Property<DateTimeOffset?>("LockoutEnd")

.HasColumnType("datetimeoffset");

b.Property<string>("Nome")

.HasColumnType("nvarchar(max)");

b.Property<string>("NormalizedEmail")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<string>("NormalizedUserName")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<string>("Numero")

.HasColumnType("nvarchar(max)");

b.Property<string>("PasswordHash")

.HasColumnType("nvarchar(max)");

b.Property<string>("PhoneNumber")

.HasColumnType("nvarchar(max)");

b.Property<bool>("PhoneNumberConfirmed")

.HasColumnType("bit");

b.Property<string>("Rua")

.HasColumnType("nvarchar(max)");

b.Property<string>("SecurityStamp")

.HasColumnType("nvarchar(max)");

b.Property<string>("Telefone")

.HasColumnType("nvarchar(max)");

b.Property<bool>("TwoFactorEnabled")

.HasColumnType("bit");

b.Property<string>("Uf")

.HasColumnType("nvarchar(max)");

b.Property<string>("UserName")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.HasKey("Id");

b.HasIndex("NormalizedEmail")

.HasName("EmailIndex");

b.HasIndex("NormalizedUserName")

.IsUnique()

.HasName("UserNameIndex")

.HasFilter("[NormalizedUserName] IS NOT NULL");

b.ToTable("AspNetUsers");

});

modelBuilder.Entity("BlockTechMVC.Models.Conta", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("Agencia")

.HasColumnType("nvarchar(max)");

b.Property<string>("Banco")

.HasColumnType("nvarchar(max)");

b.Property<string>("NomeDestinatario")

.HasColumnType("nvarchar(max)");

b.Property<string>("NumeroConta")

.HasColumnType("nvarchar(max)");

b.Property<int>("TipoConta")

.HasColumnType("int");

b.HasKey("Id");

b.ToTable("Conta");

});

modelBuilder.Entity("BlockTechMVC.Models.ContaCliente", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("ApplicationUserId")

.HasColumnType("nvarchar(450)");

b.Property<int>("ContaId")

.HasColumnType("int");

b.Property<DateTime>("DataAbertura")

.HasColumnType("datetime2");

b.Property<string>("NumeroConta")

.HasColumnType("nvarchar(max)");

b.HasKey("Id");

b.HasIndex("ApplicationUserId");

b.HasIndex("ContaId");

b.ToTable("ContaCliente");

});

modelBuilder.Entity("BlockTechMVC.Models.Criptomoeda", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<DateTime>("Cadastro")

.HasColumnType("datetime2");

b.Property<string>("Nome")

.HasColumnType("nvarchar(max)");

b.Property<string>("Simbolo")

.HasColumnType("nvarchar(max)");

b.HasKey("Id");

b.ToTable("Criptomoeda");

});

modelBuilder.Entity("BlockTechMVC.Models.CriptomoedaHoje", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<int>("CriptomoedaId")

.HasColumnType("int");

b.Property<DateTime>("Data")

.HasColumnType("datetime2");

b.Property<double>("Valor")

.HasColumnType("float");

b.HasKey("Id");

b.HasIndex("CriptomoedaId");

b.ToTable("CriptomoedaHoje");

});

modelBuilder.Entity("BlockTechMVC.Models.Saldo", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<double>("SaldoAtualRS")

.HasColumnType("float");

b.Property<int>("TransacaoId")

.HasColumnType("int");

b.Property<double>("quantidadeCripo")

.HasColumnType("float");

b.HasKey("Id");

b.ToTable("Saldo");

});

modelBuilder.Entity("BlockTechMVC.Models.Transacao", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<int>("ContaClienteId")

.HasColumnType("int");

b.Property<int>("CriptomoedaHojeId")

.HasColumnType("int");

b.Property<DateTime>("Data")

.HasColumnType("datetime2");

b.Property<int?>("SaldoId")

.HasColumnType("int");

b.Property<int>("Tipo")

.HasColumnType("int");

b.Property<double>("Valor")

.HasColumnType("float");

b.HasKey("Id");

b.HasIndex("ContaClienteId");

b.HasIndex("CriptomoedaHojeId");

b.HasIndex("SaldoId");

b.ToTable("Transacao");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityRole", b =>

{

b.Property<string>("Id")

.HasColumnType("nvarchar(450)");

b.Property<string>("ConcurrencyStamp")

.IsConcurrencyToken()

.HasColumnType("nvarchar(max)");

b.Property<string>("Name")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.Property<string>("NormalizedName")

.HasColumnType("nvarchar(256)")

.HasMaxLength(256);

b.HasKey("Id");

b.HasIndex("NormalizedName")

.IsUnique()

.HasName("RoleNameIndex")

.HasFilter("[NormalizedName] IS NOT NULL");

b.ToTable("AspNetRoles");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityRoleClaim<string>", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("ClaimType")

.HasColumnType("nvarchar(max)");

b.Property<string>("ClaimValue")

.HasColumnType("nvarchar(max)");

b.Property<string>("RoleId")

.IsRequired()

.HasColumnType("nvarchar(450)");

b.HasKey("Id");

b.HasIndex("RoleId");

b.ToTable("AspNetRoleClaims");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserClaim<string>", b =>

{

b.Property<int>("Id")

.ValueGeneratedOnAdd()

.HasColumnType("int")

.HasAnnotation("SqlServer:ValueGenerationStrategy", SqlServerValueGenerationStrategy.IdentityColumn);

b.Property<string>("ClaimType")

.HasColumnType("nvarchar(max)");

b.Property<string>("ClaimValue")

.HasColumnType("nvarchar(max)");

b.Property<string>("UserId")

.IsRequired()

.HasColumnType("nvarchar(450)");

b.HasKey("Id");

b.HasIndex("UserId");

b.ToTable("AspNetUserClaims");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserLogin<string>", b =>

{

b.Property<string>("LoginProvider")

.HasColumnType("nvarchar(128)")

.HasMaxLength(128);

b.Property<string>("ProviderKey")

.HasColumnType("nvarchar(128)")

.HasMaxLength(128);

b.Property<string>("ProviderDisplayName")

.HasColumnType("nvarchar(max)");

b.Property<string>("UserId")

.IsRequired()

.HasColumnType("nvarchar(450)");

b.HasKey("LoginProvider", "ProviderKey");

b.HasIndex("UserId");

b.ToTable("AspNetUserLogins");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserRole<string>", b =>

{

b.Property<string>("UserId")

.HasColumnType("nvarchar(450)");

b.Property<string>("RoleId")

.HasColumnType("nvarchar(450)");

b.HasKey("UserId", "RoleId");

b.HasIndex("RoleId");

b.ToTable("AspNetUserRoles");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserToken<string>", b =>

{

b.Property<string>("UserId")

.HasColumnType("nvarchar(450)");

b.Property<string>("LoginProvider")

.HasColumnType("nvarchar(128)")

.HasMaxLength(128);

b.Property<string>("Name")

.HasColumnType("nvarchar(128)")

.HasMaxLength(128);

b.Property<string>("Value")

.HasColumnType("nvarchar(max)");

b.HasKey("UserId", "LoginProvider", "Name");

b.ToTable("AspNetUserTokens");

});

modelBuilder.Entity("BlockTechMVC.Models.ContaCliente", b =>

{

b.HasOne("BlockTechMVC.Models.ApplicationUser", "ApplicationUser")

.WithMany()

.HasForeignKey("ApplicationUserId");

b.HasOne("BlockTechMVC.Models.Conta", "Conta")

.WithMany()

.HasForeignKey("ContaId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("BlockTechMVC.Models.CriptomoedaHoje", b =>

{

b.HasOne("BlockTechMVC.Models.Criptomoeda", "Criptomoeda")

.WithMany()

.HasForeignKey("CriptomoedaId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("BlockTechMVC.Models.Transacao", b =>

{

b.HasOne("BlockTechMVC.Models.ContaCliente", "ContaCliente")

.WithMany()

.HasForeignKey("ContaClienteId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("BlockTechMVC.Models.CriptomoedaHoje", "CriptomoedaHoje")

.WithMany()

.HasForeignKey("CriptomoedaHojeId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("BlockTechMVC.Models.Saldo", null)

.WithMany("Transacao")

.HasForeignKey("SaldoId");

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityRoleClaim<string>", b =>

{

b.HasOne("Microsoft.AspNetCore.Identity.IdentityRole", null)

.WithMany()

.HasForeignKey("RoleId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserClaim<string>", b =>

{

b.HasOne("BlockTechMVC.Models.ApplicationUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserLogin<string>", b =>

{

b.HasOne("BlockTechMVC.Models.ApplicationUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserRole<string>", b =>

{

b.HasOne("Microsoft.AspNetCore.Identity.IdentityRole", null)

.WithMany()

.HasForeignKey("RoleId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

b.HasOne("BlockTechMVC.Models.ApplicationUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

modelBuilder.Entity("Microsoft.AspNetCore.Identity.IdentityUserToken<string>", b =>

{

b.HasOne("BlockTechMVC.Models.ApplicationUser", null)

.WithMany()

.HasForeignKey("UserId")

.OnDelete(DeleteBehavior.Cascade)

.IsRequired();

});

#pragma warning restore 612, 618

}

}

}

**Models – Enums – TipoConta**

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Linq;

using System.Threading.Tasks;

namespace BlockTechMVC.Models.Enums

{

public enum TipoConta : int

{

[Description("Conta Corrente")]

ContaCorrente = 0,

[Description("Conta Poupança")]

ContaPoupanca = 1,

}

}

**Models – Enums - TipoTransacao**

using System.ComponentModel;

namespace BlockTechMVC.Models.Enums

{

public enum TipoConta : int

{

[Description("Conta Corrente")]

ContaCorrente = 0,

[Description("Conta Poupança")]

ContaPoupanca = 1,

}

}

**Model – ErrorViewModel**

using System.ComponentModel;

namespace BlockTechMVC.Models.Enums

{

public enum TipoConta : int

{

[Description("Conta Corrente")]

ContaCorrente = 0,

[Description("Conta Poupança")]

ContaPoupanca = 1,

}

}

**Model – ApplicationUser**

using Microsoft.AspNetCore.Identity;

using System.ComponentModel.DataAnnotations;

namespace BlockTechMVC.Models

{

public class ApplicationUser : IdentityUser

{

[Display(Name = "Nome/Razão Social")]

[RegularExpression(@"^[a-zA-Z]+$", ErrorMessage = "Use apenas caracteres alfabéticos.")]

[Required(ErrorMessage = "O campo 'Nome/Razão Social' está vazio.")]

[StringLength(60)]

public string Nome { get; set; } //Razão Social ou Nome

[StringLength(14, ErrorMessage = "CNPJ Inexistente.")]

[Display(Name = "CPF/CNPJ")]

[Required(ErrorMessage = "O campo 'Documento' está vazio.")]

public string Documento { get; set; } //CPF ou CNPJ

[StringLength(9, ErrorMessage = "CEP Inválido.")]

[Required(ErrorMessage = "O campo 'CEP' está vazio.")]

public string Cep { get; set; }

[StringLength(2, ErrorMessage = "UF Inválida.")]

[Required(ErrorMessage = "O campo 'UF' está vazio.")]

public string Uf { get; set; }

[StringLength(58, ErrorMessage = "Essa cidade não existe.")] //O maior nome de cidade possui 58 caracteres

[Required(ErrorMessage = "O campo 'Cidade' está vazio.")]

public string Cidade { get; set; }

[Required(ErrorMessage = "O campo 'Rua' está vazio.")]

public string Rua { get; set; }

[StringLength(10)]

[Required(ErrorMessage = "O campo 'Número' está vazio.")]

[Display(Name ="Número")]

public string Numero { get; set; } //Número da casa

[Required(ErrorMessage = "O campo 'Telefone' está vazio.")]

[StringLength(15, ErrorMessage = "Telefone inválido.")]

public string Telefone { get; set; }

public ApplicationUser()

{

}

public ApplicationUser(string email, string nome, string documento, string cep, string uf, string cidade, string rua, string numero, string telefone, string password) : base()

{

Email = email;

Nome = nome;

Documento = documento;

Cep = cep;

Uf = uf;

Cidade = cidade;

Rua = rua;

Numero = numero;

Telefone = telefone;

PasswordHash = password;

}

}

}

**Models – Conta**

using BlockTechMVC.Models.Enums;

using System.ComponentModel.DataAnnotations;

namespace BlockTechMVC.Models

{

public class Conta

{

public int Id { get; set; }

public string Banco { get; set; }

[StringLength(4, ErrorMessage = "Número Inválido.")]

[Display(Name = "Agência")]

public string Agencia { get; set; }

[Display(Name ="Número da Conta")]

public string NumeroConta { get; set; }

[Display(Name ="Tipo")]

public TipoConta TipoConta { get; set; }

public Conta()

{

}

public Conta(string banco, string agencia, string numeroConta, TipoConta tipoConta)

{

Banco = banco;

Agencia = agencia;

NumeroConta = numeroConta;

TipoConta = tipoConta;

}

}

}

**Models – ContaCliente**

using System;

using System.ComponentModel.DataAnnotations;

namespace BlockTechMVC.Models

{

public class ContaCliente

{

public int Id { get; set; }

[Required(ErrorMessage = "Insira o número da conta!", AllowEmptyStrings = false)]

[Display(Name="Número da Conta")]

[DisplayFormat(DataFormatString = "{0:###-##}")]

public int NumeroConta { get; set; } //IDENTITY

[Display(Name ="Data de Abertura")]

[DataType(DataType.Date)]

[DisplayFormat(DataFormatString = "{0:dd-MM-yyyy}", ApplyFormatInEditMode = true)]

public DateTime DataAbertura { get; set; }

[Display(Name = "Nome/Razão Social")]

public virtual ApplicationUser ApplicationUser { get; set; }

[Display(Name = "Nome/Razão Social")]

public string ApplicationUserID { get; set; }

[Display(Name = "Conta Associada")]

public virtual Conta Conta { get; set; }

[Display(Name = "Conta Associada")]

public int ContaId { get; set; }

public ContaCliente()

{

}

public ContaCliente(DateTime dataAbertura, ApplicationUser applicationUser, Conta conta)

{

DataAbertura = dataAbertura;

ApplicationUser = applicationUser;

Conta = conta;

}

}

}

**Models – Criptomoeda**

using System;

using System.ComponentModel.DataAnnotations;

namespace BlockTechMVC.Models

{

public class Criptomoeda

{

public int Id { get; set; }

[Display(Name ="Nome")]

public string Nome { get; set; }

[Display(Name = "Símbolo")]

[StringLength(5, ErrorMessage = "Esse símbolo não existe.")]

public string Simbolo { get; set; }

[Display(Name ="Data de Cadastro")]

[DataType(DataType.Date)]

[DisplayFormat(DataFormatString = "{0:dd-MM-yyyy}", ApplyFormatInEditMode = true)]

public DateTime Cadastro { get; set; }

public Criptomoeda()

{

}

public Criptomoeda(string nome, string simbolo, DateTime cadastro)

{

Nome = nome;

Simbolo = simbolo;

Cadastro = cadastro;

}

}

}

**Models – CriptomoedaHoje**

using System;

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace BlockTechMVC.Models

{

public class CriptomoedaHoje

{

public int Id { get; set; }

[DataType(DataType.Date)]

[DisplayFormat(DataFormatString = "{0:dd-MM-yyyy}", ApplyFormatInEditMode = true)]

public DateTime Data { get; set; }

[Column(TypeName = "decimal(20,2)")]

[DisplayFormat(DataFormatString ="{0:F2}")]

public double Valor { get; set; }

public Criptomoeda Criptomoeda { get; set; }

[Display(Name ="Criptomoeda")]

public int CriptomoedaId { get; set; }

public CriptomoedaHoje()

{

}

public CriptomoedaHoje(DateTime data, double valor, Criptomoeda criptomoeda)

{

Data = data;

Valor = valor;

Criptomoeda = criptomoeda;

}

}

}

**Models – CriptoSaldo**

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace BlockTechMVC.Models

{

public class CriptoSaldo

{

public int Id { get; set; }

public string Criptomoeda { get; set; } //Nome

//[Display(Name = "Saldo em Criptomoeda")]

[Display(Name = "Quantidade")]

[Column(TypeName = "decimal(20,6)")]

[DisplayFormat(DataFormatString = "{0:F6}")]

public double Quantidade { get; set; } //Quantidade de Criptomoedas

public ContaCliente ContaCliente { get; set; }

public int ContaClienteId { get; set; }

public CriptoSaldo()

{

}

public CriptoSaldo(string criptomoeda, double quantidade)

{

Criptomoeda = criptomoeda;

Quantidade = quantidade;

}

}

}

**Models – Saldo**

using System.ComponentModel.DataAnnotations;

using System.ComponentModel.DataAnnotations.Schema;

namespace BlockTechMVC.Models

{

public class Saldo

{

public int Id { get; set; }

[Display(Name = "Saldo Atual")]

[Column(TypeName = "decimal(20,2)")]

[DisplayFormat(DataFormatString = "{0:F2}")]

public double SaldoAtualRS { get; set; } = 0;

public ContaCliente ContaCliente { get; set; }

public int ContaClienteId { get; set; }

public Saldo()

{

}

public Saldo(double saldoAtualRS, ContaCliente contaCliente)

{

ContaCliente = contaCliente;

SaldoAtualRS = saldoAtualRS;

}

}

}

**Models – Transacao**

using BlockTechMVC.Models.Enums;

using System;

using System.ComponentModel.DataAnnotations;

namespace BlockTechMVC.Models

{

public class Transacao

{

public int Id { get; set; }

[Display(Name ="Transação")]

public TipoTransacao Tipo { get; set; }

[DataType(DataType.Date)]

[DisplayFormat(DataFormatString = "{0:dd-MM-yyyy}", ApplyFormatInEditMode = true)]

public DateTime Data { get; set; }

[DisplayFormat(DataFormatString = "{0:F2}")]

public double Valor { get; set; }

[Display(Name = "Criptomoeda")]

public int CriptomoedaHojeId { get; set; }

[Display(Name = "Criptomoeda")]

public CriptomoedaHoje CriptomoedaHoje { get; set; }

[Display(Name = "Conta")]

public int ContaClienteId { get; set; }

[Display(Name = "Conta")]

public ContaCliente ContaCliente { get; set; }

[Display(Name = "Saldo Criptomoeda")]

public CriptoSaldo CriptoSaldo { get; set; }

[Display(Name = "Saldo Criptomoeda")]

public int CriptoSaldoId { get; set; }

[Display(Name = "Saldo")]

public int SaldoId { get; set; }

[Display(Name = "Saldo")]

public Saldo Saldo { get; set; }

public Transacao()

{

}

public Transacao(TipoTransacao tipo, DateTime data, double valor, CriptomoedaHoje criptomoedaHoje, ContaCliente contaCliente)

{

Tipo = tipo;

Data = data;

Valor = valor;

CriptomoedaHoje = criptomoedaHoje;

ContaCliente = contaCliente;

}

}

}

**Views – ContasClientes – Index**

@model IEnumerable<BlockTechMVC.Models.Saldo>

@{

ViewData["Title"] = "Conta";

var total = ViewBag.Total;

int i = 0;

double[] totalGeral = new double[20];

if (User.IsInRole("Admin")) {

foreach (var item in ViewBag.TotalAdm)

{

totalGeral[i] = item;

i++;

}

i = 0;

}

}

@if (!User.IsInRole("Admin"))

{

<h1>@ViewData["Title"]</h1>

<br />

}

@if (User.IsInRole("Admin"))

{

<br />

<h1>Contas</h1>

<br />

<form asp-controller="ContasClientes" asp-action="Index" method="get">

<p>

Nome/Razão Social: <input type="text" name="SearchString">

<input type="submit" value="Filtrar" />

</p>

</form>

}

<div class="table-responsive">

<table class="table table-hover">

@if (User.IsInRole("Admin"))

{

<caption>Informações das Contas</caption>

}

else

{

<caption>Informações da Conta</caption>

}

<thead>

<tr>

<th>

@Html.DisplayNameFor(model => model.ContaCliente.DataAbertura)

</th>

@if (User.IsInRole("Admin"))

{

<th>

@Html.ActionLink("Nome", "Index", new { sortOrder = ViewBag.NameSortParm }, new { @class = "imgOrder" })

</th>

}

<th>

@Html.DisplayNameFor(model => model.ContaCliente.NumeroConta)

</th>

@if (User.IsInRole("Admin"))

{

<th>

@Html.ActionLink("Saldo", "Index", new { sortOrder = ViewBag.ValueSortParm }, new { @class = "imgOrder" })

</th>

}

else

{

<th>

@Html.DisplayNameFor(model => model.SaldoAtualRS)

</th>

}

<th></th>

<th style="white-space: nowrap"></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model)

{

<tr>

<td>

@Html.DisplayFor(modelItem => item.ContaCliente.DataAbertura)

</td>

@if (User.IsInRole("Admin"))

{

<td>

@Html.DisplayFor(modelItem => item.ContaCliente.ApplicationUser.Nome)

</td>

}

<td>

@Html.DisplayFor(modelItem => item.ContaCliente.NumeroConta)

</td>

@if (User.IsInRole("Admin"))

{

<td>

@totalGeral[i]

@{ i++; }

</td>

}

else

{

<td>

R$ @total

</td>

}

<td>

<a asp-action="Details" asp-route-id="@item.Id" class="btn btn-outline-dark">Detalhes</a>

</td>

</tr>

}

</tbody>

</table>

</div>

**Views – ContaClientes – Details**

@model BlockTechMVC.Models.Transacao

@{

ViewData["Title"] = "Detalhes";

var total = ViewBag.Total;

var totalAdm = ViewBag.TotalAdm;

}

<br />

<h1>@ViewData["Title"]</h1>

<br />

<div>

<hr />

<dl class="row">

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.ContaCliente.DataAbertura)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.ContaCliente.DataAbertura)

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.ContaCliente.ApplicationUser.Nome)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.ContaCliente.ApplicationUser.Nome)

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.ContaCliente.NumeroConta)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.ContaCliente.NumeroConta)

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.Saldo.SaldoAtualRS)

</dt>

@if (User.IsInRole("Admin"))

{

<dd class="col-sm-10">

R$ @totalAdm

</dd>

}

else

{

<dd class="col-sm-10">

R$ @total

</dd>

}

<div style="white-space:pre-line; font-weight: bold; font-size:larger" class="col-sm-12">

@ViewBag.ContaVinculada

</div>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.ContaCliente.Conta.Banco)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.ContaCliente.Conta.Banco)

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.ContaCliente.Conta.Agencia)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.ContaCliente.Conta.Agencia)

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.ContaCliente.Conta.TipoConta)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.ContaCliente.Conta.TipoConta)

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.ContaCliente.Conta.NumeroConta)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.ContaCliente.Conta.NumeroConta)

</dd>

</dl>

</div>

<div>

<a asp-action="Index">Voltar</a>

</div>

**Views – ContaClientes – Aplicacoes**

@model IEnumerable<BlockTechMVC.Models.CriptoSaldo>

@{

ViewData["Title"] = "Aplicações";

}

<br />

<h1>@ViewData["Title"]</h1>

<br />

@if (User.IsInRole("Admin"))

{

<form asp-controller="ContasClientes" asp-action="Aplicacoes" method="get">

<p>

@Html.DropDownList("Busca", null, new

{

onchange = "document.location.href='/ContasClientes/Aplicacoes?Busca='+this.value;"

})

<input type="text" name="SearchString">

<input type="submit" value="Filtrar" />

</p>

</form>

}

else

{

<form asp-controller="ContasClientes" asp-action="Aplicacoes" method="get">

<p>

Criptomoeda:

<input type="text" name="SearchString">

<input type="submit" value="Filtrar" />

</p>

</form>

}

<div class="table-responsive">

<table class="table">

<caption>Lista de Aplicações</caption>

<thead>

<tr>

@if (User.IsInRole("Admin"))

{

<th>

@Html.ActionLink("Nome", "Aplicacoes", new { sortOrder = ViewBag.Nome }, new { @class = "imgOrder" })

</th>

}

<th>

@Html.ActionLink("Criptomoeda", "Aplicacoes", new { sortOrder = ViewBag.Criptomoeda }, new { @class = "imgOrder" })

</th>

<th>

@Html.ActionLink("Quantidade", "Aplicacoes", new { sortOrder = ViewBag.Quantidade }, new { @class = "imgOrder" })

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model)

{

<tr>

@if (User.IsInRole("Admin"))

{

<td>

@Html.DisplayFor(modelItem => item.ContaCliente.ApplicationUser.Nome)

</td>

}

<td>

@Html.DisplayFor(modelItem => item.Criptomoeda)

</td>

<td>

@Html.DisplayFor(modelItem => item.Quantidade)

</td>

</tr>

}

</tbody>

</table>

</div>

**Views – Criptomoedas – Index**

@model IEnumerable<BlockTechMVC.Models.Criptomoeda>

@{

ViewData["Title"] = "Criptomoedas";

}

<br />

<h1>@ViewData["Title"]</h1>

<br />

@if (User.IsInRole("Admin"))

{

<p>

<a asp-action="Create" class="btn btn-light">Adicionar</a>

</p>

}

<form asp-controller="Criptomoedas" asp-action="Index" method="get">

<p>

Nome: <input type="text" name="SearchString">

<input type="submit" value="Filtrar" />

</p>

</form>

<div class="table-responsive">

<table class="table table-hover">

<caption>Lista de Criptomoedas</caption> @\*acessibilidade\*@

<thead>

<tr>

<th>

@Html.ActionLink("Nome", "Index", new { sortOrder = ViewBag.NameSortParm }, new { @class = "imgOrder" })

</th>

<th></th>

<th>

@Html.DisplayNameFor(model => model.Simbolo)

</th>

<th></th>

<th>

@Html.DisplayNameFor(model => model.Cadastro)

</th>

<th></th>

<th style="white-space: nowrap"></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model)

{

<tr>

<td>

@Html.DisplayFor(modelItem => item.Nome)

</td>

<td></td>

<td>

@Html.DisplayFor(modelItem => item.Simbolo)

</td>

<td></td>

<td>

@Html.DisplayFor(modelItem => item.Cadastro)

</td>

<td></td>

<td style="white-space: nowrap">

@if (User.IsInRole("Admin"))

{

<a asp-action="Edit" asp-route-id="@item.Id" class="btn btn-outline-dark">Editar</a>

<a asp-action="Delete" asp-route-id="@item.Id" class="btn btn-outline-dark">Deletar</a>

}

</td>

</tr>

}

</tbody>

</table>

</div>

**Views – Criptomoedas – Edit**

@model BlockTechMVC.Models.Criptomoeda

@{

ViewData["Title"] = "Editar";

}

<br />

<h1>@ViewData["Title"]</h1>

<br />

<hr />

<div class="row">

<div class="col-md-4">

<form asp-action="Edit">

<div asp-validation-summary="ModelOnly" class="text-danger"></div>

<input type="hidden" asp-for="Id" />

<div class="form-group">

<label asp-for="Nome" class="control-label"></label>

<input asp-for="Nome" class="form-control" />

<span asp-validation-for="Nome" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="Simbolo" class="control-label"></label>

<input asp-for="Simbolo" class="form-control" />

<span asp-validation-for="Simbolo" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="Cadastro" class="control-label"></label>

<input asp-for="Cadastro" class="form-control" />

<span asp-validation-for="Cadastro" class="text-danger"></span>

</div>

<div class="form-group">

<input type="submit" value="Salvar" class="btn btn-primary" />

</div>

</form>

</div>

</div>

<div>

<a asp-action="Index">Voltar</a>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

**Views – Criptomoedas – Details**

@model BlockTechMVC.Models.Criptomoeda

@{

ViewData["Title"] = "Detalhes";

}

<br />

<h1>@ViewData["Title"]</h1>

<br />

<div>

<hr />

<dl class="row">

<dt class = "col-sm-4">

@Html.DisplayNameFor(model => model.Nome)

</dt>

<dd class = "col-sm-8">

@Html.DisplayFor(model => model.Nome)

</dd>

<dt class = "col-sm-4">

@Html.DisplayNameFor(model => model.Simbolo)

</dt>

<dd class = "col-sm-8">

@Html.DisplayFor(model => model.Simbolo)

</dd>

<dt class = "col-sm-4">

@Html.DisplayNameFor(model => model.Cadastro)

</dt>

<dd class = "col-sm-8">

@Html.DisplayFor(model => model.Cadastro)

</dd>

</dl>

</div>

@if (User.IsInRole("Admin"))

{

<div>

<a asp-action="Edit" asp-route-id="@Model.Id" class="btn btn-outline-dark">Editar</a>

</div>

}

<div>

<a asp-action="Index">Voltar</a>

</div>

**Views – Criptomoedas – Delete**

@model BlockTechMVC.Models.Criptomoeda

@{

ViewData["Title"] = "Deletar";

}

<br />

<h1>@ViewData["Title"]</h1>

<br />

<h3>Tem certeza que deseja deletar essa criptomoeda?</h3>

<div>

<hr />

<dl class="row">

<dt class = "col-sm-4">

@Html.DisplayNameFor(model => model.Nome)

</dt>

<dd class = "col-sm-8">

@Html.DisplayFor(model => model.Nome)

</dd>

<dt class = "col-sm-4">

@Html.DisplayNameFor(model => model.Simbolo)

</dt>

<dd class = "col-sm-8">

@Html.DisplayFor(model => model.Simbolo)

</dd>

<dt class = "col-sm-4">

@Html.DisplayNameFor(model => model.Cadastro)

</dt>

<dd class = "col-sm-8">

@Html.DisplayFor(model => model.Cadastro)

</dd>

</dl>

<form asp-action="Delete">

<input type="hidden" asp-for="Id" />

<input type="submit" value="Deletar" class="btn btn-danger" />

</form>

<div>

<a asp-action="Index">Voltar</a>

</div>

</div>

**Views – Criptomoedas – Create**

@model BlockTechMVC.Models.Criptomoeda

@{

ViewData["Title"] = "Adicionar";

}

<br />

<h1>@ViewData["Title"]</h1>

<br />

<hr />

<div class="row">

<div class="col-md-4">

<form asp-action="Create">

<div asp-validation-summary="ModelOnly" class="text-danger"></div>

<div class="form-group">

<label asp-for="Nome" class="control-label"></label>

<input asp-for="Nome" class="form-control" />

<span asp-validation-for="Nome" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="Simbolo" class="control-label"></label>

<input asp-for="Simbolo" class="form-control" />

<span asp-validation-for="Simbolo" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="Cadastro" class="control-label"></label>

<input asp-for="Cadastro" class="form-control"/>

<span asp-validation-for="Cadastro" class="text-danger"></span>

</div>

<div class="form-group">

<input type="submit" value="Adicionar" class="btn btn-primary" />

</div>

</form>

</div>

</div>

<div>

<a asp-action="Index">Voltar</a>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

**Views – CriptomoedasHoje – Create**

@model BlockTechMVC.Models.CriptomoedaHoje

@{

ViewData["Title"] = "Adicionar";

}

<br />

<h1>@ViewData["Title"]</h1>

<br />

<hr />

<div class="row">

<div class="col-md-4">

<form asp-action="Create">

<div asp-validation-summary="ModelOnly" class="text-danger"></div>

<div class="form-group">

<label asp-for="Data" class="control-label"></label>

<input asp-for="Data" class="form-control" />

<span asp-validation-for="Data" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="CriptomoedaId" class="control-label"></label>

<select asp-for="CriptomoedaId" class="form-control" asp-items="ViewBag.CriptomoedaId"></select>

</div>

<div class="form-group">

<label asp-for="Valor" class="control-label"></label>

<input asp-for="Valor" class="form-control" />

<span asp-validation-for="Valor" class="text-danger"></span>

</div>

<div class="form-group">

<input type="submit" value="Adicionar" class="btn btn-primary" />

</div>

</form>

</div>

</div>

<div>

<a asp-action="Index">Voltar</a>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

**Views – CriptomoedasHoje – Delete**

@model BlockTechMVC.Models.CriptomoedaHoje

@{

ViewData["Title"] = "Deletar";

}

<br />

<h1>@ViewData["Title"]</h1>

<br />

<h3>Tem certeza que deseja deletar essa criptomoeda?</h3>

<div>

<hr />

<dl class="row">

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.Data)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.Data)

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.Criptomoeda.Simbolo)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.Criptomoeda.Simbolo)

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.Criptomoeda.Nome)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.Criptomoeda.Nome)

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.Valor)

</dt>

<dd class="col-sm-10">

R$ @Html.DisplayFor(model => model.Valor)

</dd>

</dl>

<form asp-action="Delete">

<input type="hidden" asp-for="Id" />

<input type="submit" value="Deletar" class="btn btn-danger" /> |

<a asp-action="Index">Voltar</a>

</form>

</div>

**Views – CriptomoedasHoje – Details**

@model BlockTechMVC.Models.CriptomoedaHoje

@{

ViewData["Title"] = "Detalhes";

}

<br />

<h1>@ViewData["Title"]</h1>

<div>

<hr />

<dl class="row">

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.Data)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.Data)

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.Criptomoeda.Simbolo)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.Criptomoeda.Simbolo)

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.Criptomoeda.Nome)

</dt>

<dd class="col-sm-10">

@Html.DisplayFor(model => model.Criptomoeda.Nome)

</dd>

<dt class="col-sm-2">

@Html.DisplayNameFor(model => model.Valor)

</dt>

<dd class="col-sm-10">

R$ @Html.DisplayFor(model => model.Valor)

</dd>

</dl>

</div>

<div>

@if (User.IsInRole("Admin"))

{

<a asp-action="Edit" asp-route-id="@Model.Id">Editar</a>

}

<a asp-action="Index">Voltar</a>

</div>

**Views – CriptomoedasHoje – Edit**

@model BlockTechMVC.Models.CriptomoedaHoje

@{

ViewData["Title"] = "Editar";

}

<br />

<h1>@ViewData["Title"]</h1>

<br />

<hr />

<div class="row">

<div class="col-md-4">

<form asp-action="Edit">

<div asp-validation-summary="ModelOnly" class="text-danger"></div>

<input type="hidden" asp-for="Id" />

<div class="form-group">

<label asp-for="Data" class="control-label"></label>

<input asp-for="Data" class="form-control" />

<span asp-validation-for="Data" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="CriptomoedaId" class="control-label"></label>

<select asp-for="CriptomoedaId" class="form-control" asp-items="ViewBag.CriptomoedaId"></select>

<span asp-validation-for="CriptomoedaId" class="text-danger"></span>

</div>

<div class="form-group">

<label asp-for="Valor" class="control-label"></label>

<input asp-for="Valor" class="form-control" />

<span asp-validation-for="Valor" class="text-danger"></span>

</div>

<div class="form-group">

<input type="submit" value="Salvar" class="btn btn-primary" />

</div>

</form>

</div>

</div>

<div>

<a asp-action="Index">Voltar</a>

</div>

@section Scripts {

@{await Html.RenderPartialAsync("\_ValidationScriptsPartial");}

}

**Views – CriptomoedasHoje – Index**

@model IEnumerable<BlockTechMVC.Models.CriptomoedaHoje>

@{

ViewData["Title"] = "Criptomoedas";

}

<br />

<h1>@ViewData["Title"]</h1>

<br />

@if (User.IsInRole("Admin"))

{

<p>

<a asp-action="Create" class="btn btn-light">Adicionar</a>

</p>

}

<form asp-controller="CriptomoedasHoje" asp-action="Index">

<p>

Data: <input type="date" name="SearchDate">

<input type="submit" value="Filtrar" />

</p>

</form>

<div class="table-responsive">

<table class="table table-hover">

<caption>Lista de Criptomoedas e seus valores</caption>

<thead>

<tr>

<th>

@Html.DisplayNameFor(model => model.Data)

</th>

<th>

@Html.ActionLink("Nome", "Index", new { sortOrder = ViewBag.NameSortParm }, new { @class = "imgOrder" })

</th>

<th>

@Html.ActionLink("Valor", "Index", new { sortOrder = ViewBag.ValueSortParm }, new { @class = "imgOrder" })

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model)

{

<tr>

<td>

@Html.DisplayFor(modelItem => item.Data)

</td>

<td>

@Html.DisplayFor(modelItem => item.Criptomoeda.Nome)

</td>

<td>

R$ @Html.DisplayFor(modelItem => item.Valor)

</td>

<td style="white-space: nowrap">

@if (User.IsInRole("Admin"))

{

<a asp-action="Edit" asp-route-id="@item.Id" class="btn btn-outline-dark">Editar</a>

}

<a asp-action="Details" asp-route-id="@item.Id" class="btn btn-outline-dark">Detalhes</a>

@if (User.IsInRole("Admin"))

{

<a asp-action="Delete" asp-route-id="@item.Id" class="btn btn-outline-dark">Deletar</a>

}

</td>

</tr>

}

</tbody>

</table>

</div>

**Views – CriptomoedasHoje – Simulacao**

@{

ViewData["Title"] = "Simulação";

var total = ViewBag.Total;

}

<br />

<h1 class="text-center">@ViewData["Title"]</h1>

<br />

<div class="text-center bg-light-d">

<form asp-controller="CriptomoedasHoje" asp-action="Simulacao" method="get">

<br />

<p>

Criptomoeda

@Html.DropDownList("Busca", null, new

{

onchange = "document.location.href='/CriptomoedasHoje/Simulacao?Busca='+this.value;"

})

</p>

<p>

Valor

<input type="text" name="SearchString">

</p>

<p>

<input type="submit" value="Simular" class="btn btn-dark" />

</p>

<br />

<h4>Total de criptomoeda </h4><br />

<h4>@total</h4>

<br />

</form>

</div>

**Views – Home – Index**

@{

ViewData["Title"] = "Início";

}

<div class="text-center">

<img src="~/images/logo.png" alt="logo BLockTech" />

<br />

<br />

<div class="container">

<div class="col-12 col-md-6 col-lg-6 ">

<h3 class="text-left home">Um novo conceito para gerenciar seus Criptoativos</h3>

<hr />

<br />

<p class="text-left line">

Acompanhe todas as operações realizadas, visualize

relatórios executivos, tomada de decisões e diversas

operações. Com o sistema, você terá uma ampla visão

de seus investimentos com diversas ferramentas para

análise de investimentos ao longo prazo.

</p>

</div>

<div class="col-6">

</div>

</div>

</div>

**Views – MeusInvestimentos – Bitcoin**

@{

ViewData["Title"] = "Bitcoin";

var bitcoinQuantidade = ViewBag.QuantidadaTotalBitcoin;

var bitcoinValorReal = ViewBag.QuantidadaEmRealBitcoin;

var bitcoinInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoBitcoin);

var lucroOuPerda = Convert.ToDouble(ViewBag.LucroOuPerda);

var diasList = ViewBag.Ultimos7Dias;

var dias = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(diasList));

var valores7Dias = ViewBag.ValorBitcoin7Dias;

var valores = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(valores7Dias));

var dias30List = ViewBag.UltimoMes;

var dias30 = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(dias30List));

var valores30Dias = ViewBag.ValorBitcoinMes;

var valores30 = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(valores30Dias));

var ethereumInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoEthereum);

var bitcoinCashInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoBitcoinCash);

var xrpInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoXrp);

var paxGoldInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoPaxGold);

var litecoinInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoLitecoin);

//adm

var bitcoinTotal = ViewBag.Ultimos7DiasAdm;

var bitcoinTotalAdm = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(bitcoinTotal));

var admBitcoinTotal = ViewBag.Bitcoin;

var BitcoinValorRS = ViewBag.BitcoinValorRS;

var BitcoinInvestido = ViewBag.BitcoinInvestido;

var bitcoinTotalAdmMes = ViewBag.ValorBitcoinMesAdm;

var bitcoinTotalAdm30 = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(bitcoinTotalAdmMes));

var dias30ListAdm = ViewBag.UltimoMesAdm;

var diasAdm30 = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(dias30ListAdm));

var lucroOuPerdaAdm = Convert.ToDouble(ViewBag.LucroOuPerdaAdm);

}

@if (!User.IsInRole("Admin"))

{

<div class="flex-row">

<a asp-action="Index" class="btn btn-outline-dark">Investimentos</a>

@if (bitcoinInvestimento > 0)

{

<a asp-action="Bitcoin" class="btn btn-outline-dark">Bitcoin</a>

}

@if (ethereumInvestimento > 0)

{

<a asp-action="Ethereum" class="btn btn-outline-dark">Ethereum</a>

}

@if (bitcoinCashInvestimento > 0)

{

<a asp-action="BitcoinCash" class="btn btn-outline-dark">Bitcoin Cash</a>

}

@if (xrpInvestimento > 0)

{

<a asp-action="Xrp" class="btn btn-outline-dark">XRP</a>

}

@if (paxGoldInvestimento > 0)

{

<a asp-action="PaxGold" class="btn btn-outline-dark">PAX Gold</a>

}

@if (litecoinInvestimento > 0)

{

<a asp-action="Litecoin" class="btn btn-outline-dark">Litecoin</a>

}

</div>

<br />

<h1 class="text-center">@ViewData["Title"]</h1>

<br />

<hr />

<h5>Quantidade: @bitcoinQuantidade</h5>

<h5>Valor Investido: R$ @bitcoinInvestimento</h5>

<h5>Valor Atual: R$ @bitcoinValorReal</h5>

@if (lucroOuPerda > 0)

{

<h5>Lucro: R$ @lucroOuPerda</h5>

}

else

{

if (lucroOuPerda < 0)

{

lucroOuPerda = lucroOuPerda \* (-1);

}

<h5>Perda: R$ @lucroOuPerda</h5>

}

<div id="botaoMes">

<button type="button" class="btn btn-light" onclick="TrocarGrafico()">Mensal</button>

</div>

<div id="botaoSemana">

<button type="button" class="btn btn-light" onclick="TrocarGraficoSemana()">Semanal</button>

</div>

<br />

<div id="graficoSemana">

<canvas id="graficoBitcoin" width="400" height="150">

</canvas>

</div>

<div id="graficoMes">

<canvas id="graficoBitcoin30" width="400" height="150">

</canvas>

</div>

@section Scripts {

<script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>

<script>

$(document)

.ready(function () {

var ctx = document.getElementById('graficoBitcoin').getContext('2d');

var myChart = new Chart(ctx, {

type: 'line',

data: {

labels: @dias,

datasets: [{

data: @valores,

backgroundColor: 'rgba(250, 150, 26, 0.0)',

borderColor: 'rgba(250, 150, 26, 1)',

borderWidth: 1

}]

},

options: {

legend: {

display: false

},

scales: {

yAxes: [{

ticks: {

beginAtZero: false

}

}]

}

}

});

});

$(document)

.ready(function () {

var ctx = document.getElementById('graficoBitcoin30').getContext('2d');

var myChart = new Chart(ctx, {

type: 'line',

data: {

labels: @dias30,

datasets: [{

data: @valores30,

backgroundColor: 'rgba(250, 150, 26, 0.0)',

borderColor: 'rgba(250, 150, 26, 1)',

borderWidth: 1

}]

},

options: {

legend: {

display: false

},

scales: {

yAxes: [{

ticks: {

beginAtZero: false

}

}]

}

}

});

});

</script>

}

}

else

{

<div class="flex-row">

<a asp-action="Index" class="btn btn-outline-dark">Investimentos</a>

<a asp-action="Bitcoin" class="btn btn-outline-dark">Bitcoin</a>

<a asp-action="Ethereum" class="btn btn-outline-dark">Ethereum</a>

<a asp-action="BitcoinCash" class="btn btn-outline-dark">Bitcoin Cash</a>

<a asp-action="Xrp" class="btn btn-outline-dark">XRP</a>

<a asp-action="PaxGold" class="btn btn-outline-dark">PAX Gold</a>

<a asp-action="Litecoin" class="btn btn-outline-dark">Litecoin</a>

</div>

<br />

<h1 class="text-center">@ViewData["Title"]</h1>

<br />

<h5>Quantidade: @admBitcoinTotal</h5>

<h5>Valor Investido: R$ @BitcoinInvestido</h5>

<h5>Valor Atual: R$ @BitcoinValorRS</h5>

@if (lucroOuPerdaAdm > 0)

{

<h5>Lucro: R$ @lucroOuPerdaAdm</h5>

}

else

{

if (lucroOuPerdaAdm < 0)

{

lucroOuPerdaAdm = lucroOuPerdaAdm \* (-1);

}

<h5>Perda: R$ @lucroOuPerdaAdm</h5>

}

<div id="botaoMes">

<button type="button" class="btn btn-light" onclick="TrocarGrafico()">Mensal</button>

</div>

<div id="botaoSemana">

<button type="button" class="btn btn-light" onclick="TrocarGraficoSemana()">Semanal</button>

</div>

<br />

<div id="graficoSemana">

<canvas id="graficoBitcoin" width="400" height="150">

</canvas>

</div>

<div id="graficoMes">

<canvas id="graficoBitcoin30" width="400" height="150">

</canvas>

</div>

@section Scripts {

<script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>

<script>

$(document)

.ready(function () {

var ctx = document.getElementById('graficoBitcoin').getContext('2d');

var myChart = new Chart(ctx, {

type: 'line',

data: {

labels: @dias,

datasets: [{

data: @bitcoinTotalAdm,

backgroundColor: 'rgba(250, 150, 26, 0.0)',

borderColor: 'rgba(250, 150, 26, 1)',

borderWidth: 1

}]

},

options: {

legend: {

display: false

},

scales: {

yAxes: [{

ticks: {

beginAtZero: false

}

}]

}

}

});

});

$(document)

.ready(function () {

var ctx = document.getElementById('graficoBitcoin30').getContext('2d');

var myChart = new Chart(ctx, {

type: 'line',

data: {

labels: @diasAdm30,

datasets: [{

data: @bitcoinTotalAdm30,

backgroundColor: 'rgba(250, 150, 26, 0.0)',

borderColor: 'rgba(250, 150, 26, 1)',

borderWidth: 1

}]

},

options: {

legend: {

display: false

},

scales: {

yAxes: [{

ticks: {

beginAtZero: false

}

}]

}

}

});

});

</script>

}

}

**Views – MeusInvestimentos – BitcoinCash**

@{

ViewData["Title"] = "Bitcoin Cash";

var bitcoinCashQuantidade = ViewBag.QuantidadaTotalBitcoinCash;

var bitcoinCashValorReal = ViewBag.QuantidadaEmRealBitcoinCash;

var bitcoinCashInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoBitcoinCash);

var lucroOuPerda = Convert.ToDouble(ViewBag.LucroOuPerda);

var diasList = ViewBag.Ultimos7Dias;

var dias = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(diasList));

var valores7Dias = ViewBag.ValorBitcoinCash7Dias;

var valores = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(valores7Dias));

var dias30List = ViewBag.UltimoMes;

var dias30 = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(dias30List));

var valores30Dias = ViewBag.ValorMes;

var valores30 = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(valores30Dias));

var bitcoinInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoBitcoin);

var ethereumInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoEthereum);

var xrpInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoXrp);

var paxGoldInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoPaxGold);

var litecoinInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoLitecoin);

var bitcoinCashTotal = ViewBag.Ultimos7DiasAdm;

var bitcoinCashTotalAdm = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(bitcoinCashTotal));

var admBitcoinCashTotal = ViewBag.BitcoinCash;

var BitcoinCashValorRS = ViewBag.BitcoinCashValorRS;

var BitcoinCashInvestido = ViewBag.BitcoinCashInvestido;

var TotalAdmMes = ViewBag.ValorMesAdm;

var TotalAdm30 = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(TotalAdmMes));

var dias30ListAdm = ViewBag.UltimoMesAdm;

var diasAdm30 = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(dias30ListAdm));

var lucroOuPerdaAdm = Convert.ToDouble(ViewBag.LucroOuPerdaAdm);

}

@if (!User.IsInRole("Admin"))

{

<div class="flex-row">

<a asp-action="Index" class="btn btn-outline-dark">Investimentos</a>

@if (bitcoinInvestimento > 0)

{

<a asp-action="Bitcoin" class="btn btn-outline-dark">Bitcoin</a>

}

@if (ethereumInvestimento > 0)

{

<a asp-action="Ethereum" class="btn btn-outline-dark">Ethereum</a>

}

@if (bitcoinCashInvestimento > 0)

{

<a asp-action="BitcoinCash" class="btn btn-outline-dark">Bitcoin Cash</a>

}

@if (xrpInvestimento > 0)

{

<a asp-action="Xrp" class="btn btn-outline-dark">XRP</a>

}

@if (paxGoldInvestimento > 0)

{

<a asp-action="PaxGold" class="btn btn-outline-dark">PAX Gold</a>

}

@if (litecoinInvestimento > 0)

{

<a asp-action="Litecoin" class="btn btn-outline-dark">Litecoin</a>

}

</div>

<br />

<h1 class="text-center">@ViewData["Title"]</h1>

<br>

<hr />

<h5>Quantidade: @bitcoinCashQuantidade</h5>

<h5>Valor Investido: R$ @bitcoinCashInvestimento</h5>

<h5>Valor Atual: R$ @bitcoinCashValorReal</h5>

@if (lucroOuPerda > 0)

{

<h5>Lucro: R$ @lucroOuPerda</h5>

}

else

{

if (lucroOuPerda < 0)

{

lucroOuPerda = lucroOuPerda \* (-1);

}

<h5>Perda: R$ @lucroOuPerda</h5>

}

<div id="botaoMes">

<button type="button" class="btn btn-light" onclick="TrocarGrafico()">Mensal</button>

</div>

<div id="botaoSemana">

<button type="button" class="btn btn-light" onclick="TrocarGraficoSemana()">Semanal</button>

</div>

<br />

<div id="graficoSemana">

<canvas id="graficoBitcoinCash" width="400" height="150">

</canvas>

</div>

<div id="graficoMes">

<canvas id="graficoBitcoinCash30" width="400" height="150">

</canvas>

</div>

@section Scripts {

<script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>

<script>

$(document)

.ready(function () {

var ctx = document.getElementById('graficoBitcoinCash').getContext('2d');

var myChart = new Chart(ctx, {

type: 'line',

data: {

labels: @dias,

datasets: [{

data: @valores,

backgroundColor: 'rgba(141, 195, 81, 0.0)',

borderColor: 'rgba(141, 195, 81, 1)',

borderWidth: 1

}]

},

options: {

legend: {

display: false

},

scales: {

yAxes: [{

ticks: {

beginAtZero: false

}

}]

}

}

});

});

$(document)

.ready(function () {

var ctx = document.getElementById('graficoBitcoinCash30').getContext('2d');

var myChart = new Chart(ctx, {

type: 'line',

data: {

labels: @dias30,

datasets: [{

data: @valores30,

backgroundColor: 'rgba(141, 195, 81, 0.0)',

borderColor: 'rgba(141, 195, 81, 1)',

borderWidth: 1

}]

},

options: {

legend: {

display: false

},

scales: {

yAxes: [{

ticks: {

beginAtZero: false

}

}]

}

}

});

});

</script>

}

}

else

{

<div class="flex-row">

<a asp-action="Index" class="btn btn-outline-dark">Investimentos</a>

<a asp-action="Bitcoin" class="btn btn-outline-dark">Bitcoin</a>

<a asp-action="Ethereum" class="btn btn-outline-dark">Ethereum</a>

<a asp-action="BitcoinCash" class="btn btn-outline-dark">Bitcoin Cash</a>

<a asp-action="Xrp" class="btn btn-outline-dark">XRP</a>

<a asp-action="PaxGold" class="btn btn-outline-dark">PAX Gold</a>

<a asp-action="Litecoin" class="btn btn-outline-dark">Litecoin</a>

</div>

<br />

<h1 class="text-center">@ViewData["Title"]</h1>

<br />

<h5>Quantidade: @admBitcoinCashTotal</h5>

<h5>Valor Investido: R$ @BitcoinCashInvestido</h5>

<h5>Valor Atual: R$ @BitcoinCashValorRS</h5>

@if (lucroOuPerdaAdm > 0)

{

<h5>Lucro: R$ @lucroOuPerdaAdm</h5>

}

else

{

if (lucroOuPerdaAdm < 0)

{

lucroOuPerdaAdm = lucroOuPerdaAdm \* (-1);

}

<h5>Perda: R$ @lucroOuPerdaAdm</h5>

}

<div id="botaoMes">

<button type="button" class="btn btn-light" onclick="TrocarGrafico()">Mensal</button>

</div>

<div id="botaoSemana">

<button type="button" class="btn btn-light" onclick="TrocarGraficoSemana()">Semanal</button>

</div>

<br />

<div id="graficoSemana">

<canvas id="graficoBitcoinCash" width="400" height="150">

</canvas>

</div>

<div id="graficoMes">

<canvas id="graficoBitcoinCash30" width="400" height="150">

</canvas>

</div>

<br />

@section Scripts {

<script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>

<script>

$(document)

.ready(function () {

var ctx = document.getElementById('graficoBitcoinCash').getContext('2d');

var myChart = new Chart(ctx, {

type: 'line',

data: {

labels: @dias,

datasets: [{

data: @bitcoinCashTotalAdm,

backgroundColor: 'rgba(141, 195, 81, 0.0)',

borderColor: 'rgba(141, 195, 81, 1)',

borderWidth: 1

}]

},

options: {

legend: {

display: false

},

scales: {

yAxes: [{

ticks: {

beginAtZero: false

}

}]

}

}

});

});

$(document)

.ready(function () {

var ctx = document.getElementById('graficoBitcoinCash30').getContext('2d');

var myChart = new Chart(ctx, {

type: 'line',

data: {

labels: @diasAdm30,

datasets: [{

data: @TotalAdm30,

backgroundColor: 'rgba(141, 195, 81, 0.0)',

borderColor: 'rgba(141, 195, 81, 1)',

borderWidth: 1

}]

},

options: {

legend: {

display: false

},

scales: {

yAxes: [{

ticks: {

beginAtZero: false

}

}]

}

}

});

});

</script>

}

}

**Views – MeusInvestimentos – Ethereum**

@{

ViewData["Title"] = "Ethereum";

var ethereumQuantidade = ViewBag.QuantidadaTotalEthereum;

var ethereumValorReal = ViewBag.QuantidadaEmRealEthereum;

var ethereumInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoEthereum);

var lucroOuPerda = Convert.ToDouble(ViewBag.LucroOuPerda);

var diasList = ViewBag.Ultimos7Dias;

var dias = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(diasList));

var valores7Dias = ViewBag.ValorEthereum7Dias;

var valores = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(valores7Dias));

var dias30List = ViewBag.UltimoMes;

var dias30 = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(dias30List));

var valores30Dias = ViewBag.ValorMes;

var valores30 = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(valores30Dias));

var bitcoinInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoBitcoin);

var bitcoinCashInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoBitcoinCash);

var xrpInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoXrp);

var paxGoldInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoPaxGold);

var litecoinInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoLitecoin);

var ethereumTotal = ViewBag.Ultimos7DiasAdm;

var ethereumTotalAdm = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(ethereumTotal));

var admEthereumTotal = ViewBag.Ethereum;

var EthereumValorRS = ViewBag.EthereumValorRS;

var EthereumInvestido = ViewBag.EthereumInvestido;

var TotalAdmMes = ViewBag.ValorMesAdm;

var TotalAdm30 = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(TotalAdmMes));

var dias30ListAdm = ViewBag.UltimoMesAdm;

var diasAdm30 = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(dias30ListAdm));

var lucroOuPerdaAdm = Convert.ToDouble(ViewBag.LucroOuPerdaAdm);

}

@if (!User.IsInRole("Admin"))

{

<div class="flex-row">

<a asp-action="Index" class="btn btn-outline-dark">Investimentos</a>

@if (bitcoinInvestimento > 0)

{

<a asp-action="Bitcoin" class="btn btn-outline-dark">Bitcoin</a>

}

@if (ethereumInvestimento > 0)

{

<a asp-action="Ethereum" class="btn btn-outline-dark">Ethereum</a>

}

@if (bitcoinCashInvestimento > 0)

{

<a asp-action="BitcoinCash" class="btn btn-outline-dark">Bitcoin Cash</a>

}

@if (xrpInvestimento > 0)

{

<a asp-action="Xrp" class="btn btn-outline-dark">XRP</a>

}

@if (paxGoldInvestimento > 0)

{

<a asp-action="PaxGold" class="btn btn-outline-dark">PAX Gold</a>

}

@if (litecoinInvestimento > 0)

{

<a asp-action="Litecoin" class="btn btn-outline-dark">Litecoin</a>

}

</div>

<br />

<h1 class="text-center">@ViewData["Title"]</h1>

<br />

<hr />

<h5>Quantidade: @ethereumQuantidade</h5>

<h5>Valor Investido: R$ @ethereumInvestimento</h5>

<h5>Valor Atual: R$ @ethereumValorReal</h5>

@if (lucroOuPerda > 0)

{

<h5>Lucro: R$ @lucroOuPerda</h5>

}

else

{

if (lucroOuPerda < 0)

{

lucroOuPerda = lucroOuPerda \* (-1);

}

<h5>Perda: R$ @lucroOuPerda</h5>

}

<div id="botaoMes">

<button type="button" class="btn btn-light" onclick="TrocarGrafico()">Mensal</button>

</div>

<div id="botaoSemana">

<button type="button" class="btn btn-light" onclick="TrocarGraficoSemana()">Semanal</button>

</div>

<br />

<div id="graficoSemana">

<canvas id="graficoEthereum" width="400" height="150">

</canvas>

</div>

<div id="graficoMes">

<canvas id="graficoEthereum30" width="400" height="150">

</canvas>

</div>

@section Scripts {

<script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>

<script>

$(document)

.ready(function () {

var ctx = document.getElementById('graficoEthereum').getContext('2d');

var myChart = new Chart(ctx, {

type: 'line',

data: {

labels: @dias30,

datasets: [{

data: @valores30,

backgroundColor: 'rgba(107, 108, 109, 0.0)',

borderColor: 'rgba(107, 108, 109, 1)',

borderWidth: 1

}]

},

options: {

legend: {

display: false

},

scales: {

yAxes: [{

ticks: {

beginAtZero: false

}

}]

}

}

});

});

$(document)

.ready(function () {

var ctx = document.getElementById('graficoEthereum30').getContext('2d');

var myChart = new Chart(ctx, {

type: 'line',

data: {

labels: @dias,

datasets: [{

data: @valores,

backgroundColor: 'rgba(107, 108, 109, 0.0)',

borderColor: 'rgba(107, 108, 109, 1)',

borderWidth: 1

}]

},

options: {

legend: {

display: false

},

scales: {

yAxes: [{

ticks: {

beginAtZero: false

}

}]

}

}

});

});

</script>

}

}

else

{

<div class="flex-row">

<a asp-action="Index" class="btn btn-outline-dark">Investimentos</a>

<a asp-action="Bitcoin" class="btn btn-outline-dark">Bitcoin</a>

<a asp-action="Ethereum" class="btn btn-outline-dark">Ethereum</a>

<a asp-action="BitcoinCash" class="btn btn-outline-dark">Bitcoin Cash</a>

<a asp-action="Xrp" class="btn btn-outline-dark">XRP</a>

<a asp-action="PaxGold" class="btn btn-outline-dark">PAX Gold</a>

<a asp-action="Litecoin" class="btn btn-outline-dark">Litecoin</a>

</div>

<br />

<h1 class="text-center">@ViewData["Title"]</h1>

<br />

<h5>Quantidade: @admEthereumTotal</h5>

<h5>Valor Investido: R$ @EthereumInvestido</h5>

<h5>Valor Atual: R$ @EthereumValorRS</h5>

@if (lucroOuPerdaAdm > 0)

{

<h5>Lucro: R$ @lucroOuPerdaAdm</h5>

}

else

{

if (lucroOuPerdaAdm < 0)

{

lucroOuPerdaAdm = lucroOuPerdaAdm \* (-1);

}

<h5>Perda: R$ @lucroOuPerdaAdm</h5>

}

<div id="botaoMes">

<button type="button" class="btn btn-light" onclick="TrocarGrafico()">Mensal</button>

</div>

<div id="botaoSemana">

<button type="button" class="btn btn-light" onclick="TrocarGraficoSemana()">Semanal</button>

</div>

<br />

<div id="graficoSemana">

<canvas id="graficoEhtereum" width="400" height="150">

</canvas>

</div>

<div id="graficoMes">

<canvas id="graficoEhtereum30" width="400" height="150">

</canvas>

</div>

<br />

@section Scripts {

<script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>

<script>

$(document)

.ready(function () {

var ctx = document.getElementById('graficoEhtereum').getContext('2d');

var myChart = new Chart(ctx, {

type: 'line',

data: {

labels: @dias,

datasets: [{

data: @ethereumTotalAdm,

backgroundColor: 'rgba(107, 108, 109, 0.0)',

borderColor: 'rgba(107, 108, 109, 1)',

borderWidth: 1

}]

},

options: {

legend: {

display: false

},

scales: {

yAxes: [{

ticks: {

beginAtZero: false

}

}]

}

}

});

});

$(document)

.ready(function () {

var ctx = document.getElementById('graficoEhtereum30').getContext('2d');

var myChart = new Chart(ctx, {

type: 'line',

data: {

labels: @diasAdm30,

datasets: [{

data: @TotalAdm30,

backgroundColor: 'rgba(107, 108, 109, 0.0)',

borderColor: 'rgba(107, 108, 109, 1)',

borderWidth: 1

}]

},

options: {

legend: {

display: false

},

scales: {

yAxes: [{

ticks: {

beginAtZero: false

}

}]

}

}

});

});

</script>

}

}

**Views – MeusInvestimentos – Index**

@{

ViewData["Title"] = "Investimentos";

var bitcoinQuantidade = ViewBag.QuantidadaTotalBitcoin;

var bitcoinValorReal = ViewBag.QuantidadaEmRealBitcoin;

var bitcoinInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoBitcoin);

var ethereumQuantidade = ViewBag.QuantidadaTotalEthereum;

var ethereumValorReal = ViewBag.QuantidadaEmRealEthereum;

var ethereumInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoEthereum);

var bitcoinCashQuantidade = ViewBag.QuantidadaTotalBitcoinCash;

var bitcoinCashValorReal = ViewBag.QuantidadaEmRealBitcoinCash;

var bitcoinCashInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoBitcoinCash);

var xrpQuantidade = ViewBag.QuantidadaTotalXrp;

var xrpValorReal = ViewBag.QuantidadaEmRealXrp;

var xrpInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoXrp);

var paxGoldQuantidade = ViewBag.QuantidadaTotalPaxGold;

var paxGoldValorReal = ViewBag.QuantidadaEmRealPaxGold;

var paxGoldInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoPaxGold);

var litecoinQuantidade = ViewBag.QuantidadaTotalLitecoin;

var litecoinValorReal = ViewBag.QuantidadaEmRealLitecoin;

var litecoinInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoLitecoin);

var saldoSemInvestimento = Convert.ToDouble(ViewBag.SaldoSemInvestimento);

var dinheiroTotalContaRS = Convert.ToDouble(ViewBag.DinheiroTotalContaRS);

var investido = (dinheiroTotalContaRS - saldoSemInvestimento);

//Adm

var Bitcoin = Convert.ToDouble(ViewBag.Bitcoin);

var BitcoinRS = ViewBag.BitcoinValorRS;

var Ethereum = Convert.ToDouble(ViewBag.Ethereum);

var EthereumRS = ViewBag.EthereumValorRS;

var BitcoinCash = Convert.ToDouble(ViewBag.BitcoinCash);

var BitcoinCashRS = ViewBag.BitcoinCashValorRS;

var Litecoin = Convert.ToDouble(ViewBag.Litecoin);

var LitecoinRS = ViewBag.LitecoinValorRS;

var PaxGold = Convert.ToDouble(ViewBag.PaxGold);

var PaxGoldRS = ViewBag.PaxGoldValorRS;

var Xrp = Convert.ToDouble(ViewBag.Xrp);

var XrpRS = ViewBag.XrpValorRS;

}

@if (User.IsInRole("Admin"))

{

<a asp-action="Index" class="btn btn-outline-dark menu">Investimentos</a>

<a asp-action="Bitcoin" class="btn btn-outline-dark menu">Bitcoin</a>

<a asp-action="Ethereum" class="btn btn-outline-dark menu">Ethereum</a>

<a asp-action="BitcoinCash" class="btn btn-outline-dark menu">Bitcoin Cash</a>

<a asp-action="Xrp" class="btn btn-outline-dark menu">XRP</a>

<a asp-action="PaxGold" class="btn btn-outline-dark menu">PAX Gold</a>

<a asp-action="Litecoin" class="btn btn-outline-dark menu">Litecoin</a>

<br />

<br />

<h1 class="text-center">Investimento Total</h1>

<br />

<br />

<div class="container">

<div class="row">

<div class="col-12 col-md-6 criptomoedasImg">

<div class="simboloCripto">

<img src="~/images/Bitcoin.png" alt="Símbolo da criptomoeda Bitcoin" />

</div>

<div>

<h3>Bitcoin</h3>

<h5>Quantidade: @Bitcoin</h5>

<h5>Investido: R$ @BitcoinRS</h5>

</div>

</div>

<br />

<br />

<div class="col-12 col-md-6 criptomoedasImg">

<div class="simboloCripto">

<img src="~/images/Ethereum.png" alt="Símbolo da criptomoeda Ethereum" />

</div>

<div>

<h3>Ethereum</h3>

<h5>Quantidade: @Ethereum</h5>

<h5>Investido: R$ @EthereumRS</h5>

</div>

</div>

</div>

</div>

<div class="container">

<div class="row">

<div class="col-12 col-md-6 criptomoedasImg">

<div class="simboloCripto">

<img src="~/images/BitcoinCash.png" alt="Símbolo da criptomoeda Bitcoin Cash" />

</div>

<div>

<h3>Bitcoin Cash</h3>

<h5>Quantidade: @BitcoinCash</h5>

<h5>Investido: R$ @BitcoinCashRS</h5>

</div>

</div>

<br />

<div class="col-12 col-md-6 criptomoedasImg">

<div class="simboloCripto">

<img src="~/images/Litecoin.png" alt="Símbolo da criptomoeda Litecoin" />

</div>

<div>

<h3>Litecoin</h3>

<h5>Quantidade: @Litecoin</h5>

<h5>Investido: R$ @LitecoinRS</h5>

</div>

</div>

</div>

</div>

<div class="container">

<div class="row">

<div class="col-12 col-md-6 criptomoedasImg">

<div class="simboloCripto">

<img src="~/images/PaxGold.jpg" alt="Símbolo da criptomoeda PAX Gold" />

</div>

<div>

<h3>PAX Gold</h3>

<h5>Quantidade: @PaxGold</h5>

<h5>Investido: R$ @PaxGoldRS</h5>

</div>

</div>

<br />

<br />

<div class="col-12 col-md-6 criptomoedasImg">

<div class="simboloCripto">

<img src="~/images/Xrp.png" alt="Símbolo da criptomoeda XRP" />

</div>

<div>

<h3>XRP</h3>

<h5>Quantidade: @Xrp</h5>

<h5>Investido: R$ @XrpRS</h5>

</div>

</div>

</div>

</div>

}

@if (!User.IsInRole("Admin"))

{

<div class="flex-row">

<a asp-action="Index" class="btn btn-outline-dark">Investimentos</a>

@if (bitcoinInvestimento > 0)

{

<a asp-action="Bitcoin" class="btn btn-outline-dark">Bitcoin</a>

}

@if (ethereumInvestimento > 0)

{

<a asp-action="Ethereum" class="btn btn-outline-dark">Ethereum</a>

}

@if (bitcoinCashInvestimento > 0)

{

<a asp-action="BitcoinCash" class="btn btn-outline-dark">Bitcoin Cash</a>

}

@if (xrpInvestimento > 0)

{

<a asp-action="Xrp" class="btn btn-outline-dark">XRP</a>

}

@if (paxGoldInvestimento > 0)

{

<a asp-action="PaxGold" class="btn btn-outline-dark">PAX Gold</a>

}

@if (litecoinInvestimento > 0)

{

<a asp-action="Litecoin" class="btn btn-outline-dark">Litecoin</a>

}

</div>

<br />

<h2 class="text-center">Investimento Atual</h2>

<br />

<h5>Saldo BlockTech: R$ @saldoSemInvestimento</h5>

<h5>Meu investimento: R$ @investido</h5>

<h5>Total na Conta: R$ @dinheiroTotalContaRS</h5>

<br />

<div class="container">

<div class="row">

@if (bitcoinInvestimento > 0)

{

<div class="col-12 col-md-6 criptomoedasImg">

<div class="simboloCripto">

<img src="~/images/Bitcoin.png" alt="Símbolo da criptomoeda Bitcoin" />

</div>

<div>

<h5>Quantidade de Bitcoin: @bitcoinQuantidade</h5>

<h5>Investido: R$ @bitcoinInvestimento</h5>

<h5>Valor Atual: R$ @bitcoinValorReal</h5>

</div>

</div>

}

@if (ethereumInvestimento > 0)

{

<div class="col-12 col-md-6 criptomoedasImg">

<div class="simboloCripto">

<img src="~/images/Ethereum.png" alt="Símbolo da criptomoeda Ethereum" />

</div>

<div>

<h5>Quantidade de Ethereum: @ethereumQuantidade</h5>

<h5>Investido: R$ @ethereumInvestimento</h5>

<h5>Valor Atual: R$ @ethereumValorReal</h5>

</div>

</div>

}

@if (bitcoinCashInvestimento > 0)

{

<div class="col-12 col-md-6 criptomoedasImg">

<div class="simboloCripto">

<img src="~/images/BitcoinCash.png" alt="Símbolo da criptomoeda Bitcoin Cash" />

</div>

<div>

<h5>Quantidade de Bitcoin Cash: @bitcoinCashQuantidade</h5>

<h5>Investido: R$ @bitcoinCashInvestimento</h5>

<h5>Valor Atual: R$ @bitcoinCashValorReal</h5>

</div>

</div>

}

@if (xrpInvestimento > 0)

{

<div class="col-12 col-md-6 criptomoedasImg">

<div class="simboloCripto">

<img src="~/images/Xrp.png" alt="Símbolo da criptomoeda XRP" />

</div>

<div>

<h5>Quantidade de XRP: @xrpQuantidade</h5>

<h5>Investido: R$ @xrpInvestimento</h5>

<h5>Valor Atual: R$ @xrpValorReal</h5>

</div>

</div>

}

@if (paxGoldInvestimento > 0)

{

<div class="col-12 col-md-6 criptomoedasImg">

<div class="simboloCripto">

<img src="~/images/PaxGold.jpg" alt="Símbolo da criptomoeda PAX Gold" />

</div>

<div>

<h5>Quantidade de PAX Gold: @paxGoldQuantidade</h5>

<h5>Investido: R$ @paxGoldInvestimento</h5>

<h5>Valor Atual: R$ @paxGoldValorReal</h5>

</div>

</div>

}

@if (litecoinInvestimento > 0)

{

<div class="col-12 col-md-6 criptomoedasImg">

<div class="simboloCripto">

<img src="~/images/Litecoin.png" alt="Símbolo da criptomoeda Litecoin" />

</div>

<div>

<h5>Quantidade de Litecoin: @litecoinQuantidade</h5>

<h5>Investido: R$ @litecoinInvestimento</h5>

<h5>Valor Atual: R$ @litecoinValorReal</h5>

</div>

</div>

}

</div>

</div>

}

**Views – MeusInvestimentos – Litecoin**

@{

ViewData["Title"] = "Litecoin";

var litecoinQuantidade = ViewBag.QuantidadaTotalLitecoin;

var litecoinValorReal = ViewBag.QuantidadaEmRealLitecoin;

var litecoinInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoLitecoin);

var lucroOuPerda = Convert.ToDouble(ViewBag.LucroOuPerda);

var diasList = ViewBag.Ultimos7Dias;

var dias = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(diasList));

var valores7Dias = ViewBag.ValorLitecoin7Dias;

var valores = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(valores7Dias));

var dias30List = ViewBag.UltimoMes;

var dias30 = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(dias30List));

var valores30Dias = ViewBag.ValorMes;

var valores30 = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(valores30Dias));

var bitcoinInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoBitcoin);

var ethereumInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoEthereum);

var xrpInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoXrp);

var bitcoinCashInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoBitcoinCash);

var paxGoldInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoPaxGold);

var litecoinTotal = ViewBag.Ultimos7DiasAdm;

var litecoinTotalAdm = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(litecoinTotal));

var admLitecoinTotal = ViewBag.Litecoin;

var LitecoinValorRS = ViewBag.LitecoinValorRS;

var LitecoinInvestido = ViewBag.LitecoinInvestido;

var TotalAdmMes = ViewBag.ValorMesAdm;

var TotalAdm30 = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(TotalAdmMes));

var dias30ListAdm = ViewBag.UltimoMesAdm;

var diasAdm30 = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(dias30ListAdm));

var lucroOuPerdaAdm = Convert.ToDouble(ViewBag.LucroOuPerdaAdm);

}

@if (!User.IsInRole("Admin"))

{

<div class="flex-row">

<a asp-action="Index" class="btn btn-outline-dark">Investimentos</a>

@if (bitcoinInvestimento > 0)

{

<a asp-action="Bitcoin" class="btn btn-outline-dark">Bitcoin</a>

}

@if (ethereumInvestimento > 0)

{

<a asp-action="Ethereum" class="btn btn-outline-dark">Ethereum</a>

}

@if (bitcoinCashInvestimento > 0)

{

<a asp-action="BitcoinCash" class="btn btn-outline-dark">Bitcoin Cash</a>

}

@if (xrpInvestimento > 0)

{

<a asp-action="Xrp" class="btn btn-outline-dark">XRP</a>

}

@if (paxGoldInvestimento > 0)

{

<a asp-action="PaxGold" class="btn btn-outline-dark">PAX Gold</a>

}

@if (litecoinInvestimento > 0)

{

<a asp-action="Litecoin" class="btn btn-outline-dark">Litecoin</a>

}

</div>

<br />

<h1 class="text-center">@ViewData["Title"]</h1>

<br />

<hr />

<h5>Quantidade: @litecoinQuantidade</h5>

<h5>Valor Investido: R$ @litecoinInvestimento</h5>

<h5>Valor Atual: R$ @litecoinValorReal</h5>

@if (lucroOuPerda > 0)

{

<h5>Lucro: R$ @lucroOuPerda</h5>

}

else

{

if (lucroOuPerda < 0)

{

lucroOuPerda = lucroOuPerda \* (-1);

}

<h5>Perda: R$ @lucroOuPerda</h5>

}

<div id="botaoMes">

<button type="button" class="btn btn-light" onclick="TrocarGrafico()">Mensal</button>

</div>

<div id="botaoSemana">

<button type="button" class="btn btn-light" onclick="TrocarGraficoSemana()">Semanal</button>

</div>

<br />

<div id="graficoSemana">

<canvas id="graficoLitecoin" width="400" height="150">

</canvas>

</div>

<div id="graficoMes">

<canvas id="graficoLitecoin30" width="400" height="150">

</canvas>

</div>

@section Scripts {

<script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>

<script>

$(document)

.ready(function () {

var ctx = document.getElementById('graficoLitecoin').getContext('2d');

var myChart = new Chart(ctx, {

type: 'line',

data: {

labels: @dias,

datasets: [{

data: @valores,

backgroundColor: 'rgba(52, 92, 157, 0.0)',

borderColor: 'rgba(52, 92, 157, 1)',

borderWidth: 1

}]

},

options: {

legend: {

display: false

},

scales: {

yAxes: [{

ticks: {

beginAtZero: false

}

}]

}

}

});

});

$(document)

.ready(function () {

var ctx = document.getElementById('graficoLitecoin30').getContext('2d');

var myChart = new Chart(ctx, {

type: 'line',

data: {

labels: @dias30,

datasets: [{

data: @valores30,

backgroundColor: 'rgba(52, 92, 157, 0.0)',

borderColor: 'rgba(52, 92, 157, 1)',

borderWidth: 1

}]

},

options: {

legend: {

display: false

},

scales: {

yAxes: [{

ticks: {

beginAtZero: false

}

}]

}

}

});

});

</script>

}

}

else

{

<div class="flex-row">

<a asp-action="Index" class="btn btn-outline-dark">Investimentos</a>

<a asp-action="Bitcoin" class="btn btn-outline-dark">Bitcoin</a>

<a asp-action="Ethereum" class="btn btn-outline-dark">Ethereum</a>

<a asp-action="BitcoinCash" class="btn btn-outline-dark">Bitcoin Cash</a>

<a asp-action="Xrp" class="btn btn-outline-dark">XRP</a>

<a asp-action="PaxGold" class="btn btn-outline-dark">PAX Gold</a>

<a asp-action="Litecoin" class="btn btn-outline-dark">Litecoin</a>

</div>

<br />

<h1 class="text-center">@ViewData["Title"]</h1>

<br />

<h5>Quantidade: @admLitecoinTotal</h5>

<h5>Valor Investido: R$ @LitecoinInvestido</h5>

<h5>Valor Atual: R$ @LitecoinValorRS</h5>

@if (lucroOuPerdaAdm > 0)

{

<h5>Lucro: R$ @lucroOuPerdaAdm</h5>

}

else

{

if (lucroOuPerdaAdm < 0)

{

lucroOuPerdaAdm = lucroOuPerdaAdm \* (-1);

}

<h5>Perda: R$ @lucroOuPerdaAdm</h5>

}

<br />

<div id="botaoMes">

<button type="button" class="btn btn-light" onclick="TrocarGrafico()">Mensal</button>

</div>

<div id="botaoSemana">

<button type="button" class="btn btn-light" onclick="TrocarGraficoSemana()">Semanal</button>

</div>

<br />

<div id="graficoSemana">

<canvas id="graficoLitecoin" width="400" height="150">

</canvas>

</div>

<div id="graficoMes">

<canvas id="graficoLitecoin30" width="400" height="150">

</canvas>

</div>

@section Scripts {

<script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>

<script>

$(document)

.ready(function () {

var ctx = document.getElementById('graficoLitecoin').getContext('2d');

var myChart = new Chart(ctx, {

type: 'line',

data: {

labels: @dias,

datasets: [{

data: @litecoinTotalAdm,

backgroundColor: 'rgba(52, 92, 157, 0.0)',

borderColor: 'rgba(52, 92, 157, 1)',

borderWidth: 1

}]

},

options: {

legend: {

display: false

},

scales: {

yAxes: [{

ticks: {

beginAtZero: false

}

}]

}

}

});

});

$(document)

.ready(function () {

var ctx = document.getElementById('graficoLitecoin30').getContext('2d');

var myChart = new Chart(ctx, {

type: 'line',

data: {

labels: @diasAdm30,

datasets: [{

data: @TotalAdm30,

backgroundColor: 'rgba(52, 92, 157, 0.0)',

borderColor: 'rgba(52, 92, 157, 1)',

borderWidth: 1

}]

},

options: {

legend: {

display: false

},

scales: {

yAxes: [{

ticks: {

beginAtZero: false

}

}]

}

}

});

});

</script>

}

}

**Views – MeusInvestimentos – PaxGold**

@{

ViewData["Title"] = "PAX Gold";

var paxGoldQuantidade = ViewBag.QuantidadaTotalPaxGold;

var paxGoldValorReal = ViewBag.QuantidadaEmRealPaxGold;

var paxGoldInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoPaxGold);

var lucroOuPerda = Convert.ToDouble(ViewBag.LucroOuPerda);

var diasList = ViewBag.Ultimos7Dias;

var dias = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(diasList));

var valores7Dias = ViewBag.ValorPaxGold7Dias;

var valores = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(valores7Dias));

var dias30List = ViewBag.UltimoMes;

var dias30 = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(dias30List));

var valores30Dias = ViewBag.ValorMes;

var valores30 = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(valores30Dias));

var bitcoinInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoBitcoin);

var ethereumInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoEthereum);

var xrpInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoXrp);

var bitcoinCashInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoBitcoinCash);

var litecoinInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoLitecoin);

var paxGoldTotal = ViewBag.Ultimos7DiasAdm;

var paxGoldTotalAdm = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(paxGoldTotal));

var admPaxGoldTotal = ViewBag.PaxGold;

var PaxGoldValorRS = ViewBag.PaxGoldValorRS;

var PaxGoldInvestido = ViewBag.PaxGoldInvestido;

var TotalAdmMes = ViewBag.ValorMesAdm;

var TotalAdm30 = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(TotalAdmMes));

var dias30ListAdm = ViewBag.UltimoMesAdm;

var diasAdm30 = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(dias30ListAdm));

var lucroOuPerdaAdm = Convert.ToDouble(ViewBag.LucroOuPerdaAdm);

}

@if (!User.IsInRole("Admin"))

{

<div class="flex-row">

<a asp-action="Index" class="btn btn-outline-dark">Investimentos</a>

@if (bitcoinInvestimento > 0)

{

<a asp-action="Bitcoin" class="btn btn-outline-dark">Bitcoin</a>

}

@if (ethereumInvestimento > 0)

{

<a asp-action="Ethereum" class="btn btn-outline-dark">Ethereum</a>

}

@if (bitcoinCashInvestimento > 0)

{

<a asp-action="BitcoinCash" class="btn btn-outline-dark">Bitcoin Cash</a>

}

@if (xrpInvestimento > 0)

{

<a asp-action="Xrp" class="btn btn-outline-dark">XRP</a>

}

@if (paxGoldInvestimento > 0)

{

<a asp-action="PaxGold" class="btn btn-outline-dark">PAX Gold</a>

}

@if (litecoinInvestimento > 0)

{

<a asp-action="Litecoin" class="btn btn-outline-dark">Litecoin</a>

}

</div>

<br />

<h1 class="text-center">@ViewData["Title"]</h1>

<br />

<hr />

<h5>Quantidade: @paxGoldQuantidade</h5>

<h5>Valor Investido: R$ @paxGoldInvestimento</h5>

<h5>Valor Atual: R$ @paxGoldValorReal</h5>

@if (lucroOuPerda > 0)

{

<h5>Lucro: R$ @lucroOuPerda</h5>

}

else

{

if (lucroOuPerda < 0)

{

lucroOuPerda = lucroOuPerda \* (-1);

}

<h5>Perda: R$ @lucroOuPerda</h5>

}

<div id="botaoMes">

<button type="button" class="btn btn-light" onclick="TrocarGrafico()">Mensal</button>

</div>

<div id="botaoSemana">

<button type="button" class="btn btn-light" onclick="TrocarGraficoSemana()">Semanal</button>

</div>

<br />

<div id="graficoSemana">

<canvas id="graficoPaxGold" width="400" height="150">

</canvas>

</div>

<div id="graficoMes">

<canvas id="graficoPaxGold30" width="400" height="150">

</canvas>

</div>

@section Scripts {

<script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>

<script>

$(document)

.ready(function () {

var ctx = document.getElementById('graficoPaxGold').getContext('2d');

var myChart = new Chart(ctx, {

type: 'line',

data: {

labels: @dias,

datasets: [{

data: @valores,

backgroundColor: 'rgba(250, 244, 15, 0.0)',

borderColor: 'rgba(250, 244, 15, 1)',

borderWidth: 1

}]

},

options: {

legend: {

display: false

},

scales: {

yAxes: [{

ticks: {

beginAtZero: false

}

}]

}

}

});

});

$(document)

.ready(function () {

var ctx = document.getElementById('graficoPaxGold30').getContext('2d');

var myChart = new Chart(ctx, {

type: 'line',

data: {

labels: @dias30,

datasets: [{

data: @valores30,

backgroundColor: 'rgba(250, 244, 15, 0.0)',

borderColor: 'rgba(250, 244, 15, 1)',

borderWidth: 1

}]

},

options: {

legend: {

display: false

},

scales: {

yAxes: [{

ticks: {

beginAtZero: false

}

}]

}

}

});

});

</script>

}

}

else

{

<div class="flex-row">

<a asp-action="Index" class="btn btn-outline-dark">Investimentos</a>

<a asp-action="Bitcoin" class="btn btn-outline-dark">Bitcoin</a>

<a asp-action="Ethereum" class="btn btn-outline-dark">Ethereum</a>

<a asp-action="BitcoinCash" class="btn btn-outline-dark">Bitcoin Cash</a>

<a asp-action="Xrp" class="btn btn-outline-dark">XRP</a>

<a asp-action="PaxGold" class="btn btn-outline-dark">PAX Gold</a>

<a asp-action="Litecoin" class="btn btn-outline-dark">Litecoin</a>

</div>

<br />

<h1 class="text-center">@ViewData["Title"]</h1>

<br />

<h5>Quantidade: @admPaxGoldTotal</h5>

<h5>Valor Investido: R$ @PaxGoldInvestido</h5>

<h5>Valor Atual: R$ @PaxGoldValorRS</h5>

@if (lucroOuPerdaAdm > 0)

{

<h5>Lucro: R$ @lucroOuPerdaAdm</h5>

}

else

{

if (lucroOuPerdaAdm < 0)

{

lucroOuPerdaAdm = lucroOuPerdaAdm \* (-1);

}

<h5>Perda: R$ @lucroOuPerdaAdm</h5>

}

<div id="botaoMes">

<button type="button" class="btn btn-light" onclick="TrocarGrafico()">Mensal</button>

</div>

<div id="botaoSemana">

<button type="button" class="btn btn-light" onclick="TrocarGraficoSemana()">Semanal</button>

</div>

<br />

<div id="graficoSemana">

<canvas id="graficoPaxGold" width="400" height="150">

</canvas>

</div>

<div id="graficoMes">

<canvas id="graficoPaxGold30" width="400" height="150">

</canvas>

</div>

@section Scripts {

<script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>

<script>

$(document)

.ready(function () {

var ctx = document.getElementById('graficoPaxGold').getContext('2d');

var myChart = new Chart(ctx, {

type: 'line',

data: {

labels: @dias,

datasets: [{

data: @paxGoldTotalAdm,

backgroundColor: 'rgba(250, 244, 15, 0.0)',

borderColor: 'rgba(250, 244, 15, 1)',

borderWidth: 1

}]

},

options: {

legend: {

display: false

},

scales: {

yAxes: [{

ticks: {

beginAtZero: false

}

}]

}

}

});

});

$(document)

.ready(function () {

var ctx = document.getElementById('graficoPaxGold30').getContext('2d');

var myChart = new Chart(ctx, {

type: 'line',

data: {

labels: @diasAdm30,

datasets: [{

data: @TotalAdm30,

backgroundColor: 'rgba(250, 244, 15, 0.0)',

borderColor: 'rgba(250, 244, 15, 1)',

borderWidth: 1

}]

},

options: {

legend: {

display: false

},

scales: {

yAxes: [{

ticks: {

beginAtZero: false

}

}]

}

}

});

});

</script>

}

}

**Views – MeusInvestimentos – Xrp**

@{

ViewData["Title"] = "XRP";

var xrpQuantidade = ViewBag.QuantidadaTotalXrp;

var xrpValorReal = ViewBag.QuantidadaEmRealXrp;

var xrpInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoXrp);

var lucroOuPerda = Convert.ToDouble(ViewBag.LucroOuPerda);

var diasList = ViewBag.Ultimos7Dias;

var dias = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(diasList));

var valores7Dias = ViewBag.ValorXrp7Dias;

var valores = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(valores7Dias));

var dias30List = ViewBag.UltimoMes;

var dias30 = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(dias30List));

var valores30Dias = ViewBag.ValorMes;

var valores30 = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(valores30Dias));

var bitcoinInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoBitcoin);

var ethereumInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoEthereum);

var paxGoldInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoPaxGold);

var bitcoinCashInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoBitcoinCash);

var litecoinInvestimento = Convert.ToDouble(ViewBag.ValorInvestidoLitecoin);

var xrpTotal = ViewBag.Ultimos7DiasAdm;

var xrpTotalAdm = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(xrpTotal));

var admXrpTotal = ViewBag.Xrp;

var XrpValorRS = ViewBag.XrpValorRS;

var XrpInvestido = ViewBag.XrpInvestido;

var TotalAdmMes = ViewBag.ValorMesAdm;

var TotalAdm30 = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(TotalAdmMes));

var dias30ListAdm = ViewBag.UltimoMesAdm;

var diasAdm30 = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(dias30ListAdm));

var lucroOuPerdaAdm = Convert.ToDouble(ViewBag.LucroOuPerdaAdm);

}

@if (!User.IsInRole("Admin"))

{

<div class="flex-row">

<a asp-action="Index" class="btn btn-outline-dark">Investimentos</a>

@if (bitcoinInvestimento > 0)

{

<a asp-action="Bitcoin" class="btn btn-outline-dark">Bitcoin</a>

}

@if (ethereumInvestimento > 0)

{

<a asp-action="Ethereum" class="btn btn-outline-dark">Ethereum</a>

}

@if (bitcoinCashInvestimento > 0)

{

<a asp-action="BitcoinCash" class="btn btn-outline-dark">Bitcoin Cash</a>

}

@if (xrpInvestimento > 0)

{

<a asp-action="Xrp" class="btn btn-outline-dark">XRP</a>

}

@if (paxGoldInvestimento > 0)

{

<a asp-action="PaxGold" class="btn btn-outline-dark">PAX Gold</a>

}

@if (litecoinInvestimento > 0)

{

<a asp-action="Litecoin" class="btn btn-outline-dark">Litecoin</a>

}

</div>

<br />

<h1 class="text-center">@ViewData["Title"]</h1>

<br />

<hr />

<h5>Quantidade: @xrpQuantidade</h5>

<h5>Valor Investido: R$ @xrpInvestimento</h5>

<h5>Valor Atual: R$ @xrpValorReal</h5>

@if (lucroOuPerda > 0)

{

<h5>Lucro: R$ @lucroOuPerda</h5>

}

else

{

if (lucroOuPerda < 0)

{

lucroOuPerda = lucroOuPerda \* (-1);

}

<h5>Perda: R$ @lucroOuPerda</h5>

}

<div id="botaoMes">

<button type="button" class="btn btn-light" onclick="TrocarGrafico()">Mensal</button>

</div>

<div id="botaoSemana">

<button type="button" class="btn btn-light" onclick="TrocarGraficoSemana()">Semanal</button>

</div>

<br />

<div id="graficoSemana">

<canvas id="graficoXrp" width="400" height="150">

</canvas>

</div>

<div id="graficoMes">

<canvas id="graficoXrp30" width="400" height="150">

</canvas>

</div>

@section Scripts {

<script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>

<script>

$(document)

.ready(function () {

var ctx = document.getElementById('graficoXrp').getContext('2d');

var myChart = new Chart(ctx, {

type: 'line',

data: {

labels: @dias,

datasets: [{

data: @valores,

backgroundColor: 'rgba(35, 41, 47, 0.0)',

borderColor: 'rgba(35, 41, 47, 1)',

borderWidth: 1

}]

},

options: {

legend: {

display: false

},

scales: {

yAxes: [{

ticks: {

beginAtZero: false

}

}]

}

}

});

});

$(document)

.ready(function () {

var ctx = document.getElementById('graficoXrp30').getContext('2d');

var myChart = new Chart(ctx, {

type: 'line',

data: {

labels: @dias30,

datasets: [{

data: @valores30,

backgroundColor: 'rgba(35, 41, 47, 0.0)',

borderColor: 'rgba(35, 41, 47, 1)',

borderWidth: 1

}]

},

options: {

legend: {

display: false

},

scales: {

yAxes: [{

ticks: {

beginAtZero: false

}

}]

}

}

});

});

</script>

}

}

else

{

<div class="flex-row">

<a asp-action="Index" class="btn btn-outline-dark">Investimentos</a>

<a asp-action="Bitcoin" class="btn btn-outline-dark">Bitcoin</a>

<a asp-action="Ethereum" class="btn btn-outline-dark">Ethereum</a>

<a asp-action="BitcoinCash" class="btn btn-outline-dark">Bitcoin Cash</a>

<a asp-action="Xrp" class="btn btn-outline-dark">XRP</a>

<a asp-action="PaxGold" class="btn btn-outline-dark">PAX Gold</a>

<a asp-action="Litecoin" class="btn btn-outline-dark">Litecoin</a>

</div>

<br />

<h1 class="text-center">@ViewData["Title"]</h1>

<br />

<h5>Quantidade: @admXrpTotal</h5>

<h5>Valor Investido: R$ @XrpInvestido</h5>

<h5>Valor Atual: R$ @XrpValorRS</h5>

@if (lucroOuPerdaAdm > 0)

{

<h5>Lucro: R$ @lucroOuPerdaAdm</h5>

}

else

{

if (lucroOuPerdaAdm < 0)

{

lucroOuPerdaAdm = lucroOuPerdaAdm \* (-1);

}

<h5>Perda: R$ @lucroOuPerdaAdm</h5>

}

<div id="botaoMes">

<button type="button" class="btn btn-light" onclick="TrocarGrafico()">Mensal</button>

</div>

<div id="botaoSemana">

<button type="button" class="btn btn-light" onclick="TrocarGraficoSemana()">Semanal</button>

</div>

<br />

<div id="graficoSemana">

<canvas id="graficoXrp" width="400" height="150">

</canvas>

</div>

<div id="graficoMes">

<canvas id="graficoXrp30" width="400" height="150">

</canvas>

</div>

@section Scripts {

<script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>

<script>

$(document)

.ready(function () {

var ctx = document.getElementById('graficoXrp').getContext('2d');

var myChart = new Chart(ctx, {

type: 'line',

data: {

labels: @dias,

datasets: [{

data: @xrpTotalAdm,

backgroundColor: 'rgba(35, 41, 47, 0.0)',

borderColor: 'rgba(35, 41, 47, 1)',

borderWidth: 1

}]

},

options: {

legend: {

display: false

},

scales: {

yAxes: [{

ticks: {

beginAtZero: false

}

}]

}

}

});

});

$(document)

.ready(function () {

var ctx = document.getElementById('graficoXrp30').getContext('2d');

var myChart = new Chart(ctx, {

type: 'line',

data: {

labels: @diasAdm30,

datasets: [{

data: @TotalAdm30,

backgroundColor: 'rgba(35, 41, 47, 0.0)',

borderColor: 'rgba(35, 41, 47, 1)',

borderWidth: 1

}]

},

options: {

legend: {

display: false

},

scales: {

yAxes: [{

ticks: {

beginAtZero: false

}

}]

}

}

});

});

</script>

}

}

**Views – Relatorios - Bitcoin**

@{

ViewData["Title"] = "Bitcoin";

var diasList = ViewBag.Dias;

var valoresList = ViewBag.Valores;

var dias = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(diasList));

var valores = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(valoresList));

}

<br />

<div class="flex-row">

<a asp-action="Index" class="btn btn-outline-dark">Hoje</a>

<a asp-action="Mensal" class="btn btn-outline-dark">Últimos 30 dias</a>

<a asp-action="Semanal" class="btn btn-outline-dark">Todas as Criptomoedas</a>

<a asp-action="Bitcoin" class="btn btn-dark text-white">Bitcoin</a>

<a asp-action="Ethereum" class="btn btn-outline-dark">Ethereum</a>

<a asp-action="BitcoinCash" class="btn btn-outline-dark">Bitcoin Cash</a>

<a asp-action="Xrp" class="btn btn-outline-dark">XRP</a>

<a asp-action="PaxGold" class="btn btn-outline-dark">PAX Gold</a>

<a asp-action="Litecoin" class="btn btn-outline-dark">Litecoin</a>

</div>

<br />

<h1>@ViewData["Title"]</h1>

<br />

<h4 class="text-center">Últimos 7 Dias</h4>

<canvas id="graficoBitcoin" width="400" height="150">

</canvas>

@section Scripts {

<script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>

<script>

$(document)

.ready(function () {

var ctx = document.getElementById('graficoBitcoin').getContext('2d');

var myChart = new Chart(ctx, {

type: 'line',

data: {

labels: @dias,

datasets: [{

data: @valores,

backgroundColor: 'rgba(250, 150, 26, 0.0)',

borderColor: 'rgba(250, 150, 26, 1)',

borderWidth: 1

}]

},

options: {

legend: {

display: false

},

scales: {

yAxes: [{

ticks: {

beginAtZero: false

}

}]

}

}

});

});

</script>

}

**Views – Relatorios – Bitcoin30**

@{

ViewData["Title"] = "Bitcoin";

var diasList = ViewBag.Dias;

var valoresList = ViewBag.Valores;

var dias = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(diasList));

var valores = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(valoresList));

}

<br />

<div class="flex-row">

<a asp-action="Index" class="btn btn-outline-dark menu">Hoje</a>

<a asp-action="Semanal" class="btn btn-outline-dark menu">Últimos 7 dias</a>

<a asp-action="Mensal" class="btn btn-outline-dark menu">Todas as Criptomoedas</a>

<a asp-action="Bitcoin30" class="btn btn-dark text-white">Bitcoin</a>

<a asp-action="Ethereum30" class="btn btn-outline-dark menu">Ethereum</a>

<a asp-action="BitcoinCash30" class="btn btn-outline-dark menu">Bitcoin Cash</a>

<a asp-action="Xrp30" class="btn btn-outline-dark menu">XRP</a>

<a asp-action="PaxGold30" class="btn btn-outline-dark menu">PAX Gold</a>

<a asp-action="Litecoin30" class="btn btn-outline-dark menu">Litecoin</a>

</div>

<br />

<h1>@ViewData["Title"]</h1>

<br />

<h4 class="text-center">Últimos 30 Dias</h4>

<canvas id="graficoBitcoin" width="400" height="150">

</canvas>

@section Scripts {

<script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>

<script>

$(document)

.ready(function () {

var ctx = document.getElementById('graficoBitcoin').getContext('2d');

var myChart = new Chart(ctx, {

type: 'line',

data: {

labels: @dias,

datasets: [{

data: @valores,

backgroundColor: 'rgba(250, 150, 26, 0.0)',

borderColor: 'rgba(250, 150, 26, 1)',

borderWidth: 1

}]

},

options: {

legend: {

display: false

},

scales: {

yAxes: [{

ticks: {

beginAtZero: false

}

}]

}

}

});

});

</script>

}

**Views – Relatorios – BitcoinCash**

@{

ViewData["Title"] = "Bitcoin Cash";

var diasList = ViewBag.Dias;

var valoresList = ViewBag.Valores;

var dias = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(diasList));

var valores = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(valoresList));

}

<br />

<div class="flex-row">

<a asp-action="Index" class="btn btn-outline-dark">Hoje</a>

<a asp-action="Mensal" class="btn btn-outline-dark">Últimos 30 dias</a>

<a asp-action="Semanal" class="btn btn-outline-dark">Todas as Criptomoedas</a>

<a asp-action="Bitcoin" class="btn btn-outline-dark">Bitcoin</a>

<a asp-action="Ethereum" class="btn btn-outline-dark">Ethereum</a>

<a asp-action="BitcoinCash" class="btn btn-dark text-white">Bitcoin Cash</a>

<a asp-action="Xrp" class="btn btn-outline-dark">XRP</a>

<a asp-action="PaxGold" class="btn btn-outline-dark">PAX Gold</a>

<a asp-action="Litecoin" class="btn btn-outline-dark">Litecoin</a>

</div>

<br />

<h1>@ViewData["Title"]</h1>

<br />

<h4 class="text-center">Últimos 7 Dias</h4>

<canvas id="graficoBitcoinCash" width="400" height="150">

</canvas>

@section Scripts {

<script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>

<script>

$(document)

.ready(function () {

var ctx = document.getElementById('graficoBitcoinCash').getContext('2d');

var myChart = new Chart(ctx, {

type: 'line',

data: {

labels: @dias,

datasets: [{

data: @valores,

backgroundColor: 'rgba(141, 195, 81, 0.0)',

borderColor: 'rgba(141, 195, 81, 1)',

borderWidth: 1

}]

},

options: {

legend: {

display: false

},

scales: {

yAxes: [{

ticks: {

beginAtZero: false

}

}]

}

}

});

});

</script>

}

**Views – Relatorios – BitcoinCash30**

@{

ViewData["Title"] = "Bitcoin Cash";

var diasList = ViewBag.Dias;

var valoresList = ViewBag.Valores;

var dias = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(diasList));

var valores = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(valoresList));

}

<br />

<div class="flex-row">

<a asp-action="Index" class="btn btn-outline-dark menu">Hoje</a>

<a asp-action="Semanal" class="btn btn-outline-dark menu">Últimos 7 dias</a>

<a asp-action="Mensal" class="btn btn-outline-dark menu">Todas as Criptomoedas</a>

<a asp-action="Bitcoin30" class="btn btn-outline-dark menu">Bitcoin</a>

<a asp-action="Ethereum30" class="btn btn-outline-dark menu">Ethereum</a>

<a asp-action="BitcoinCash30" class="btn btn-dark text-white menu">Bitcoin Cash</a>

<a asp-action="Xrp30" class="btn btn-outline-dark menu">XRP</a>

<a asp-action="PaxGold30" class="btn btn-outline-dark menu">PAX Gold</a>

<a asp-action="Litecoin30" class="btn btn-outline-dark menu">Litecoin</a>

</div>

<br />

<h1>@ViewData["Title"]</h1>

<br />

<h4 class="text-center">Últimos 30 Dias</h4>

<canvas id="graficoBitcoinCash" width="400" height="150">

</canvas>

@section Scripts {

<script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>

<script>

$(document)

.ready(function () {

var ctx = document.getElementById('graficoBitcoinCash').getContext('2d');

var myChart = new Chart(ctx, {

type: 'line',

data: {

labels: @dias,

datasets: [{

data: @valores,

backgroundColor: 'rgba(141, 195, 81, 0.0)',

borderColor: 'rgba(141, 195, 81, 1)',

borderWidth: 1

}]

},

options: {

legend: {

display: false

},

scales: {

yAxes: [{

ticks: {

beginAtZero: false

}

}]

}

}

});

});

</script>

}

**Views – Relatorios – Ethereum**

@{

ViewData["Title"] = "Ethereum";

var diasList = ViewBag.Dias;

var valoresList = ViewBag.Valores;

var dias = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(diasList));

var valores = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(valoresList));

}

<br />

<div class="flex-row">

<a asp-action="Index" class="btn btn-outline-dark">Hoje</a>

<a asp-action="Mensal" class="btn btn-outline-dark">Últimos 30 dias</a>

<a asp-action="Semanal" class="btn btn-outline-dark">Todas as Criptomoedas</a>

<a asp-action="Bitcoin" class="btn btn-outline-dark">Bitcoin</a>

<a asp-action="Ethereum" class="btn btn-dark text-white">Ethereum</a>

<a asp-action="BitcoinCash" class="btn btn-outline-dark">Bitcoin Cash</a>

<a asp-action="Xrp" class="btn btn-outline-dark">XRP</a>

<a asp-action="PaxGold" class="btn btn-outline-dark">PAX Gold</a>

<a asp-action="Litecoin" class="btn btn-outline-dark">Litecoin</a>

</div>

<br />

<h1>@ViewData["Title"]</h1>

<br />

<h4 class="text-center">Últimos 7 Dias</h4>

<canvas id="graficoEthereum" width="400" height="150">

</canvas>

@section Scripts {

<script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>

<script>

$(document)

.ready(function () {

var ctx = document.getElementById('graficoEthereum').getContext('2d');

var myChart = new Chart(ctx, {

type: 'line',

data: {

labels: @dias,

datasets: [{

data: @valores,

backgroundColor: 'rgba(107, 108, 109, 0.0)',

borderColor: 'rgba(107, 108, 109, 1)',

borderWidth: 1

}]

},

options: {

legend: {

display: false

},

scales: {

yAxes: [{

ticks: {

beginAtZero: false

}

}]

}

}

});

});

</script>

}

**Views – Relatorios – Ethereum30**

@{

ViewData["Title"] = "Ethereum";

var diasList = ViewBag.Dias;

var valoresList = ViewBag.Valores;

var dias = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(diasList));

var valores = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(valoresList));

}

<br />

<div class="flex-row">

<a asp-action="Index" class="btn btn-outline-dark menu">Hoje</a>

<a asp-action="Semanal" class="btn btn-outline-dark menu">Últimos 7 dias</a>

<a asp-action="Mensal" class="btn btn-outline-dark menu">Todas as Criptomoedas</a>

<a asp-action="Bitcoin30" class="btn btn-outline-dark menu">Bitcoin</a>

<a asp-action="Ethereum30" class="btn btn-dark text-white">Ethereum</a>

<a asp-action="BitcoinCash30" class="btn btn-outline-dark menu">Bitcoin Cash</a>

<a asp-action="Xrp30" class="btn btn-outline-dark menu">XRP</a>

<a asp-action="PaxGold30" class="btn btn-outline-dark menu">PAX Gold</a>

<a asp-action="Litecoin30" class="btn btn-outline-dark menu">Litecoin</a>

</div>

<br />

<h1>@ViewData["Title"]</h1>

<br />

<h4 class="text-center">Últimos 30 Dias</h4>

<canvas id="graficoEthereum" width="400" height="150">

</canvas>

@section Scripts {

<script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>

<script>

$(document)

.ready(function () {

var ctx = document.getElementById('graficoEthereum').getContext('2d');

var myChart = new Chart(ctx, {

type: 'line',

data: {

labels: @dias,

datasets: [{

data: @valores,

backgroundColor: 'rgba(107, 108, 109, 0.0)',

borderColor: 'rgba(107, 108, 109, 1)',

borderWidth: 1

}]

},

options: {

legend: {

display: false

},

scales: {

yAxes: [{

ticks: {

beginAtZero: false

}

}]

}

}

});

});

</script>

}

**Views – Relatorios – Index**

@{

ViewData["Title"] = "Relatórios";

ViewData["Data"] = DateTime.Now.ToString("dd-MM-yyyy HH:mm");

var bitcoin = ViewBag.Bitcoin;

var ethereum = ViewBag.Ethereum;

var bitcoinCash = ViewBag.BitcoinCash;

var xrp = ViewBag.XRP;

var paxGold = ViewBag.PaxGold;

var litecoin = ViewBag.Litecoin;

}

<br />

<div class="flex-row">

<a asp-action="Index" class="btn btn-dark text-white">Hoje</a>

<a asp-action="Semanal" class="btn btn-outline-dark">Últimos 7 dias</a>

<a asp-action="Mensal" class="btn btn-outline-dark">Últimos 30 dias</a>

</div>

<br />

<h4 class="text-center">Valor Por Criptomoeda @ViewData["Data"]</h4>

<br />

<canvas id="graficoCriptomoedas" width="400" height="150"></canvas>

@section Scripts {

<script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>

<script>

$(document)

.ready(function () {

var ctx = document.getElementById('graficoCriptomoedas').getContext('2d');

var myChart = new Chart(ctx, {

type: 'bar',

data: {

labels: ['Bitcoin', 'Ethereum', 'Bitcoin Cash', 'XRP', 'PAX Gold', 'Litecoin'],

datasets: [{

data: [@bitcoin, @ethereum, @bitcoinCash, @xrp, @paxGold, @litecoin],

backgroundColor: [

'rgba(250, 150, 26, 0.2)',

'rgba(107, 108, 109, 0.2)',

'rgba(141, 195, 81, 0.2)',

'rgba(35, 41, 47, 0.2)',

'rgba(250, 244, 15, 0.2)',

'rgba(52, 92, 157, 0.2)'

],

borderColor: [

'rgba(250, 150, 26, 1)',

'rgba(107, 108, 109, 1)',

'rgba(141, 195, 81, 1)',

'rgba(35, 41, 47, 1)',

'rgba(250, 244, 15, 1)',

'rgba(52, 92, 157, 1)'

],

borderWidth: 1

}]

},

options: {

legend: {

display: false

},

scales: {

yAxes: [{

ticks: {

beginAtZero: false

}

}]

}

}

});

});

</script>

}

**Views – Relatorios – Litecoin**

@{

ViewData["Title"] = "Litecoin";

var diasList = ViewBag.Dias;

var valoresList = ViewBag.Valores;

var dias = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(diasList));

var valores = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(valoresList));

}

<br />

<div class="flex-row">

<a asp-action="Index" class="btn btn-outline-dark">Hoje</a>

<a asp-action="Mensal" class="btn btn-outline-dark">Últimos 30 dias</a>

<a asp-action="Semanal" class="btn btn-outline-dark">Todas as Criptomoedas</a>

<a asp-action="Bitcoin" class="btn btn-outline-dark">Bitcoin</a>

<a asp-action="Ethereum" class="btn btn-outline-dark">Ethereum</a>

<a asp-action="BitcoinCash" class="btn btn-outline-dark">Bitcoin Cash</a>

<a asp-action="Xrp" class="btn btn-outline-dark">XRP</a>

<a asp-action="PaxGold" class="btn btn-outline-dark">PAX Gold</a>

<a asp-action="Litecoin" class="btn btn-dark text-white">Litecoin</a>

</div>

<br />

<h1>@ViewData["Title"]</h1>

<br />

<h4 class="text-center">Últimos 7 Dias</h4>

<canvas id="graficoBitcoin" width="400" height="150">

</canvas>

@section Scripts {

<script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>

<script>

$(document)

.ready(function () {

var ctx = document.getElementById('graficoBitcoin').getContext('2d');

var myChart = new Chart(ctx, {

type: 'line',

data: {

labels: @dias,

datasets: [{

data: @valores,

backgroundColor: 'rgba(52, 92, 157, 0.0)',

borderColor: 'rgba(52, 92, 157, 1)',

borderWidth: 1

}]

},

options: {

legend: {

display: false

},

scales: {

yAxes: [{

ticks: {

beginAtZero: false

}

}]

}

}

});

});

</script>

}

**Views – Relatorios – Litecoin30**

@{

ViewData["Title"] = "Litecoin";

var diasList = ViewBag.Dias;

var valoresList = ViewBag.Valores;

var dias = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(diasList));

var valores = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(valoresList));

}

<br />

<div class="flex-row">

<a asp-action="Index" class="btn btn-outline-dark menu">Hoje</a>

<a asp-action="Semanal" class="btn btn-outline-dark menu">Últimos 7 dias</a>

<a asp-action="Mensal" class="btn btn-outline-dark menu">Todas as Criptomoedas</a>

<a asp-action="Bitcoin30" class="btn btn-outline-dark menu">Bitcoin</a>

<a asp-action="Ethereum30" class="btn btn-outline-dark menu">Ethereum</a>

<a asp-action="BitcoinCash30" class="btn btn-outline-dark menu">Bitcoin Cash</a>

<a asp-action="Xrp30" class="btn btn-outline-dark menu">XRP</a>

<a asp-action="PaxGold30" class="btn btn-outline-dark menu">PAX Gold</a>

<a asp-action="Litecoin30" class="btn btn-dark text-white">Litecoin</a>

</div>

<br />

<h1>@ViewData["Title"]</h1>

<br />

<h4 class="text-center">Últimos 30 Dias</h4>

<canvas id="graficoLitecoin" width="400" height="150">

</canvas>

@section Scripts {

<script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>

<script>

$(document)

.ready(function () {

var ctx = document.getElementById('graficoLitecoin').getContext('2d');

var myChart = new Chart(ctx, {

type: 'line',

data: {

labels: @dias,

datasets: [{

data: @valores,

backgroundColor: 'rgba(52, 92, 157, 0.0)',

borderColor: 'rgba(52, 92, 157, 1)',

borderWidth: 1

}]

},

options: {

legend: {

display: false

},

scales: {

yAxes: [{

ticks: {

beginAtZero: false

}

}]

}

}

});

});

</script>

}

**Views – Relatorios - Mensal**

@{

ViewData["Title"] = "Relatórios";

var diasList = ViewBag.Dias;

var bitcoinList = ViewBag.Bitcoin;

var ethereumList = ViewBag.Ethereum;

var bitcoinCashList = ViewBag.BitcoinCash;

var xrpList = ViewBag.XRP;

var paxGoldList = ViewBag.PaxGold;

var litecoinList = ViewBag.Litecoin;

var dias = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(diasList));

var bitcoin = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(bitcoinList));

var ethereum = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(ethereumList));

var bitcoinCash = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(bitcoinCashList));

var xrp = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(xrpList));

var paxGold = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(paxGoldList));

var litecoin = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(litecoinList));

}

<br />

<div>

<a asp-action="Index" class="btn btn-outline-dark">Hoje</a>

<a asp-action="Semanal" class="btn btn-outline-dark">Últimos 7 dias</a>

<a asp-action="Mensal" class="btn btn-dark text-white">Todas as Criptomoedas</a>

<a asp-action="Bitcoin30" class="btn btn-outline-dark menu">Bitcoin</a>

<a asp-action="Ethereum30" class="btn btn-outline-dark menu">Ethereum</a>

<a asp-action="BitcoinCash30" class="btn btn-outline-dark menu">Bitcoin Cash</a>

<a asp-action="Xrp30" class="btn btn-outline-dark menu">XRP</a>

<a asp-action="PaxGold30" class="btn btn-outline-dark menu">PAX Gold</a>

<a asp-action="Litecoin30" class="btn btn-outline-dark menu">Litecoin</a>

</div>

<br />

<h4 class="text-center">Últimos 30 Dias - Taxa de Crescimento</h4>

<br />

<div class="container align-items-center">

<div class="row">

<div class="col-md-2 col-6 legendas">

<div id="bitcoinLegenda"></div>

<div class="textoLegenda">

Bitcoin

</div>

</div>

<div class="col-md-2 col-6 legendas">

<div id="ethereumLegenda"></div>

<div class="textoLegenda">

Ethereum

</div>

</div>

<div class="col-md-2 col-6 legendas">

<div id="bitcoinCashLegenda"></div>

<div class="textoLegenda">

Bitcoin Cash

</div>

</div>

<div class="col-md-2 col-6 legendas">

<div id="xrpLegenda"></div>

<div class="textoLegenda">

XRP

</div>

</div>

<div class="col-md-2 col-6 legendas">

<div id="paxGoldLegenda"></div>

<div class="textoLegenda">

PAX Gold

</div>

</div>

<div class="col-md-2 col-6 legendas">

<div id="litecoinLegenda"></div>

<div class="textoLegenda">

Litecoin

</div>

</div>

</div>

<br />

</div>

<canvas id="graficoCriptomoedas" width="400" height="150" style="min-height: 30vh;"></canvas>

@section Scripts {

<script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>

<script>

$(document)

.ready(function () {

var ctx = document.getElementById('graficoCriptomoedas').getContext('2d');

var myChart = new Chart(ctx, {

type: 'line',

data: {

labels: @dias,

datasets: [{

label: 'Bitcoin',

data: @bitcoin,

backgroundColor: 'rgba(250, 150, 26, 0)',

borderColor: 'rgba(250, 150, 26, 1)',

borderWidth: 1

},

{

label: 'Ethereum',

data: @ethereum,

backgroundColor: 'rgba(107, 108, 109, 0)',

borderColor: 'rgba(107, 108, 109, 1)',

borderWidth: 1

},

{

label: 'Bitcoin Cash',

data: @bitcoinCash,

backgroundColor: 'rgba(141, 195, 81, 0)',

borderColor: 'rgba(141, 195, 81, 1)',

borderWidth: 1

},

{

label: 'XRP',

data: @xrp,

backgroundColor: 'rgba(35, 41, 47, 0)',

borderColor: 'rgba(35, 41, 47, 1)',

borderWidth: 1

},

{

label: 'PAX Gold',

data: @paxGold,

backgroundColor: 'rgba(250, 244, 15, 0)',

borderColor: 'rgba(250, 244, 15, 1)',

borderWidth: 1

},

{

label: 'Litecoin',

data: @litecoin,

backgroundColor: 'rgba(52, 92, 157, 0)',

borderColor: 'rgba(52, 92, 157, 1)',

borderWidth: 1

}]

},

options: {

legend: {

display: false

},

scales: {

yAxes: [{

ticks: {

beginAtZero: false

}

}]

}

}

});

});

</script>

}

**Views – Relatorios – PaxGold**

@{

ViewData["Title"] = "PAX Gold";

var diasList = ViewBag.Dias;

var valoresList = ViewBag.Valores;

var dias = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(diasList));

var valores = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(valoresList));

}

<br />

<div class="flex-row">

<a asp-action="Index" class="btn btn-outline-dark">Hoje</a>

<a asp-action="Mensal" class="btn btn-outline-dark">Últimos 30 dias</a>

<a asp-action="Semanal" class="btn btn-outline-dark">Todas as Criptomoedas</a>

<a asp-action="Bitcoin" class="btn btn-outline-dark">Bitcoin</a>

<a asp-action="Ethereum" class="btn btn-outline-dark">Ethereum</a>

<a asp-action="BitcoinCash" class="btn btn-outline-dark">Bitcoin Cash</a>

<a asp-action="Xrp" class="btn btn-outline-dark">XRP</a>

<a asp-action="PaxGold" class="btn btn-dark text-white">PAX Gold</a>

<a asp-action="Litecoin" class="btn btn-outline-dark">Litecoin</a>

</div>

<br />

<h1>@ViewData["Title"]</h1>

<br />

<h4 class="text-center">Últimos 7 Dias</h4>

<canvas id="graficoBPaxGold" width="400" height="150">

</canvas>

@section Scripts {

<script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>

<script>

$(document)

.ready(function () {

var ctx = document.getElementById('graficoBPaxGold').getContext('2d');

var myChart = new Chart(ctx, {

type: 'line',

data: {

labels: @dias,

datasets: [{

data: @valores,

backgroundColor: 'rgba(250, 244, 15, 0.0)',

borderColor: 'rgba(250, 244, 15, 1)',

borderWidth: 1

}]

},

options: {

legend: {

display: false

},

scales: {

yAxes: [{

ticks: {

beginAtZero: false

}

}]

}

}

});

});

</script>

}

**Views – Relatorios – PaxGold30**

@{

ViewData["Title"] = "PAX Gold";

var diasList = ViewBag.Dias;

var valoresList = ViewBag.Valores;

var dias = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(diasList));

var valores = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(valoresList));

}

<br />

<div class="flex-row">

<a asp-action="Index" class="btn btn-outline-dark menu">Hoje</a>

<a asp-action="Semanal" class="btn btn-outline-dark menu">Últimos 7 dias</a>

<a asp-action="Mensal" class="btn btn-outline-dark menu">Todas as Criptomoedas</a>

<a asp-action="Bitcoin30" class="btn btn-outline-dark menu">Bitcoin</a>

<a asp-action="Ethereum30" class="btn btn-outline-dark menu">Ethereum</a>

<a asp-action="BitcoinCash30" class="btn btn-outline-dark menu">Bitcoin Cash</a>

<a asp-action="Xrp30" class="btn btn-outline-dark menu">XRP</a>

<a asp-action="PaxGold30" class="btn btn-dark text-white">PAX Gold</a>

<a asp-action="Litecoin30" class="btn btn-outline-dark menu">Litecoin</a>

</div>

<br />

<h1>@ViewData["Title"]</h1>

<br />

<h4 class="text-center">Últimos 30 Dias</h4>

<canvas id="graficoBPaxGold" width="400" height="150">

</canvas>

@section Scripts {

<script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>

<script>

$(document)

.ready(function () {

var ctx = document.getElementById('graficoBPaxGold').getContext('2d');

var myChart = new Chart(ctx, {

type: 'line',

data: {

labels: @dias,

datasets: [{

data: @valores,

backgroundColor: 'rgba(250, 244, 15, 0.0)',

borderColor: 'rgba(250, 244, 15, 1)',

borderWidth: 1

}]

},

options: {

legend: {

display: false

},

scales: {

yAxes: [{

ticks: {

beginAtZero: false

}

}]

}

}

});

});

</script>

}

**Views – Relatorios – Semanal**

@{

ViewData["Title"] = "Relatórios";

var diasList = ViewBag.Dias;

var bitcoinList = ViewBag.Bitcoin;

var ethereumList = ViewBag.Ethereum;

var bitcoinCashList = ViewBag.BitcoinCash;

var xrpList = ViewBag.XRP;

var paxGoldList = ViewBag.PaxGold;

var litecoinList = ViewBag.Litecoin;

var dias = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(diasList));

var bitcoin = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(bitcoinList));

var ethereum = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(ethereumList));

var bitcoinCash = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(bitcoinCashList));

var xrp = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(xrpList));

var paxGold = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(paxGoldList));

var litecoin = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(litecoinList));

}

<br />

<div>

<a asp-action="Index" class="btn btn-outline-dark menu">Hoje</a>

<a asp-action="Mensal" class="btn btn-outline-dark menu">Últimos 30 dias</a>

<a asp-action="Semanal" class="btn btn-dark text-white menu">Todas as Criptomoedas</a>

<a asp-action="Bitcoin" class="btn btn-outline-dark menu">Bitcoin</a>

<a asp-action="Ethereum" class="btn btn-outline-dark menu">Ethereum</a>

<a asp-action="BitcoinCash" class="btn btn-outline-dark menu">Bitcoin Cash</a>

<a asp-action="Xrp" class="btn btn-outline-dark menu">XRP</a>

<a asp-action="PaxGold" class="btn btn-outline-dark menu">PAX Gold</a>

<a asp-action="Litecoin" class="btn btn-outline-dark menu">Litecoin</a>

</div>

<br />

<h4 class="text-center">Últimos 7 Dias - Taxa de Crescimento</h4>

<br />

<div class="container align-items-center">

<div class="row">

<div class="col-md-2 col-6 legendas">

<div id="bitcoinLegenda"></div>

<div class="textoLegenda">

Bitcoin

</div>

</div>

<div class="col-md-2 col-6 legendas">

<div id="ethereumLegenda"></div>

<div class="textoLegenda">

Ethereum

</div>

</div>

<div class="col-md-2 col-6 legendas">

<div id="bitcoinCashLegenda"></div>

<div class="textoLegenda">

Bitcoin Cash

</div>

</div>

<div class="col-md-2 col-6 legendas">

<div id="xrpLegenda"></div>

<div class="textoLegenda">

XRP

</div>

</div>

<div class="col-md-2 col-6 legendas">

<div id="paxGoldLegenda"></div>

<div class="textoLegenda">

PAX Gold

</div>

</div>

<div class="col-md-2 col-6 legendas">

<div id="litecoinLegenda"></div>

<div class="textoLegenda">

Litecoin

</div>

</div>

</div>

<br />

</div>

<canvas id="graficoCriptomoedas" width="400" height="150" style="min-height: 30vh;"></canvas>

@section Scripts {

<script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>

<script>

$(document)

.ready(function () {

var ctx = document.getElementById('graficoCriptomoedas').getContext('2d');

var myChart = new Chart(ctx, {

type: 'line',

data: {

labels: @dias,

datasets: [{

label: 'Bitcoin',

data: @bitcoin,

backgroundColor: 'rgba(250, 150, 26, 0)',

borderColor: 'rgba(250, 150, 26, 1)',

borderWidth: 1

},

{

label: 'Ethereum',

data: @ethereum,

backgroundColor: 'rgba(107, 108, 109, 0)',

borderColor: 'rgba(107, 108, 109, 1)',

borderWidth: 1

},

{

label: 'Bitcoin Cash',

data: @bitcoinCash,

backgroundColor: 'rgba(141, 195, 81, 0)',

borderColor: 'rgba(141, 195, 81, 1)',

borderWidth: 1

},

{

label: 'XRP',

data: @xrp,

backgroundColor: 'rgba(35, 41, 47, 0)',

borderColor: 'rgba(35, 41, 47, 1)',

borderWidth: 1

},

{

label: 'PAX Gold',

data: @paxGold,

backgroundColor: 'rgba(250, 244, 15, 0)',

borderColor: 'rgba(250, 244, 15, 1)',

borderWidth: 1

},

{

label: 'Litecoin',

data: @litecoin,

backgroundColor: 'rgba(52, 92, 157, 0)',

borderColor: 'rgba(52, 92, 157, 1)',

borderWidth: 1

}]

},

options: {

legend: {

display: false

},

scales: {

yAxes: [{

ticks: {

beginAtZero: false

}

}]

}

}

});

});

</script>

}

**Views – Relatorios – Xrp**

@{

ViewData["Title"] = "XRP";

var diasList = ViewBag.Dias;

var valoresList = ViewBag.Valores;

var dias = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(diasList));

var valores = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(valoresList));

}

<br />

<div class="flex-row">

<a asp-action="Index" class="btn btn-outline-dark">Hoje</a>

<a asp-action="Mensal" class="btn btn-outline-dark">Últimos 30 dias</a>

<a asp-action="Semanal" class="btn btn-outline-dark">Todas as Criptomoedas</a>

<a asp-action="Bitcoin" class="btn btn-outline-dark">Bitcoin</a>

<a asp-action="Ethereum" class="btn btn-outline-dark">Ethereum</a>

<a asp-action="BitcoinCash" class="btn btn-outline-dark">Bitcoin Cash</a>

<a asp-action="Xrp" class="btn btn-dark text-white">XRP</a>

<a asp-action="PaxGold" class="btn btn-outline-dark">PAX Gold</a>

<a asp-action="Litecoin" class="btn btn-outline-dark">Litecoin</a>

</div>

<br />

<h1>@ViewData["Title"]</h1>

<br />

<h4 class="text-center">Últimos 7 Dias</h4>

<canvas id="graficoXrp" width="400" height="150">

</canvas>

@section Scripts {

<script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>

<script>

$(document)

.ready(function () {

var ctx = document.getElementById('graficoXrp').getContext('2d');

var myChart = new Chart(ctx, {

type: 'line',

data: {

labels: @dias,

datasets: [{

data: @valores,

backgroundColor: 'rgba(35, 41, 47, 0.0)',

borderColor: 'rgba(35, 41, 47, 1)',

borderWidth: 1

}]

},

options: {

legend: {

display: false

},

scales: {

yAxes: [{

ticks: {

beginAtZero: false

}

}]

}

}

});

});

</script>

}

**Views – Relatorios – Xrp30**

@{

ViewData["Title"] = "XRP";

var diasList = ViewBag.Dias;

var valoresList = ViewBag.Valores;

var dias = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(diasList));

var valores = Html.Raw(Newtonsoft.Json.JsonConvert.SerializeObject(valoresList));

}

<br />

<div class="flex-row">

<a asp-action="Index" class="btn btn-outline-dark menu">Hoje</a>

<a asp-action="Semanal" class="btn btn-outline-dark menu">Últimos 7 dias</a>

<a asp-action="Mensal" class="btn btn-outline-dark menu">Todas as Criptomoedas</a>

<a asp-action="Bitcoin30" class="btn btn-outline-dark menu">Bitcoin</a>

<a asp-action="Ethereum30" class="btn btn-outline-dark menu">Ethereum</a>

<a asp-action="BitcoinCash30" class="btn btn-outline-dark menu">Bitcoin Cash</a>

<a asp-action="Xrp30" class="btn btn-dark text-white">XRP</a>

<a asp-action="PaxGold30" class="btn btn-outline-dark menu">PAX Gold</a>

<a asp-action="Litecoin30" class="btn btn-outline-dark menu">Litecoin</a>

</div>

<br />

<h1>@ViewData["Title"]</h1>

<br />

<h4 class="text-center">Últimos 30 Dias</h4>

<canvas id="graficoXrp" width="400" height="150">

</canvas>

@section Scripts {

<script src="https://cdn.jsdelivr.net/npm/chart.js@2.8.0"></script>

<script>

$(document)

.ready(function () {

var ctx = document.getElementById('graficoXrp').getContext('2d');

var myChart = new Chart(ctx, {

type: 'line',

data: {

labels: @dias,

datasets: [{

data: @valores,

backgroundColor: 'rgba(35, 41, 47, 0.0)',

borderColor: 'rgba(35, 41, 47, 1)',

borderWidth: 1

}]

},

options: {

legend: {

display: false

},

scales: {

yAxes: [{

ticks: {

beginAtZero: false

}

}]

}

}

});

});

</script>

}

**Views – Shared - \_Layout**

<!DOCTYPE html>

<html lang="pt-br">

<head>

<meta charset="utf-8" />

<meta name="viewport" content="width=device-width, initial-scale=1.0" />

<title>@ViewData["Title"] - BlockTech</title>

<link rel="stylesheet" href="~/lib/bootstrap/dist/css/bootstrap.min.css" />

<link rel="stylesheet" href="~/css/site.css" />

<link rel="shortcut icon" href="~/Icon/logo.ico" type="image/x-icon" />

</head>

<body>

<header>

<nav class="navbar navbar-expand-sm navbar-toggleable-sm navbar-dark bg-dark border-bottom box-shadow mb-3">

<div class="container">

<a class="navbar-brand" asp-area="" asp-controller="Home" asp-action="Index">

<img src="~/images/logoMin.png" width="50"/>

</a>

<button class="navbar-toggler" type="button" data-toggle="collapse" data-target=".navbar-collapse" aria-controls="navbarSupportedContent"

aria-expanded="false" aria-label="Toggle navigation">

<span class="navbar-toggler-icon"></span>

</button>

<div class="navbar-collapse collapse d-sm-inline-flex flex-sm-row-reverse">

<partial name="\_LoginPartial" />

<ul class="navbar-nav flex-grow-1">

<li class="nav-item">

<a class="nav-link text-white" asp-area="" asp-controller="Home" asp-action="Index">Início</a>

</li>

@if (User?.Identity.IsAuthenticated == true)

{

@if (User.IsInRole("Admin"))

{

<li class="nav-item">

<a class="nav-link text-white" asp-area="" asp-controller="Criptomoedas" asp-action="Index">Criptomoedas</a>

</li>

}

@if (User.IsInRole("Admin"))

{

<li class="nav-item">

<a class="nav-link text-white" asp-area="" asp-controller="CriptomoedasHoje" asp-action="Index">Tempo Real</a>

</li>

}

@if (!User.IsInRole("Admin"))

{

<li class="nav-item">

<a class="nav-link text-white" asp-area="" asp-controller="CriptomoedasHoje" asp-action="Index">Criptomoedas</a>

</li>

}

<li class="nav-item">

<a class="nav-link text-white" asp-area="" asp-controller="Transacoes" asp-action="Index">Transações</a>

</li>

<li class="nav-item">

<a class="nav-link text-white" asp-area="" asp-controller="ContasClientes" asp-action="Index">Conta</a>

</li>

<li class="nav-item">

<a class="nav-link text-white" asp-area="" asp-controller="ContasClientes" asp-action="Aplicacoes">Aplicações</a>

</li>

<li class="nav-item">

<a class="nav-link text-white" asp-area="" asp-controller="Relatorios" asp-action="Index">Relatórios</a>

</li>

<li class="nav-item">

<a class="nav-link text-white" asp-area="" asp-controller="MeusInvestimentos" asp-action="Index">Investimentos</a>

</li>

<li class="nav-item">

<a class="nav-link text-white" asp-area="" asp-controller="CriptomoedasHoje" asp-action="Simulacao">Simulação</a>

</li>

}

</ul>

</div>

</div>

</nav>

</header>

<div class="container">

<main role="main" class="pb-3">

@RenderBody()

</main>

</div>

<footer class="border-top footer text-muted">

<div class="container">

&copy; 2020 - BlockTech

</div>

</footer>

<script src="~/lib/jquery/dist/jquery.min.js"></script>

<script src="~/lib/bootstrap/dist/js/bootstrap.bundle.min.js"></script>

<script src="~/js/site.js" asp-append-version="true"></script>

@RenderSection("Scripts", required: false)

</body>

</html>

**Views – Shared - \_LoginPartial**

@using Microsoft.AspNetCore.Identity

@inject SignInManager<ApplicationUser> SignInManager

@inject UserManager<ApplicationUser> UserManager

<ul class="navbar-nav">

@if (SignInManager.IsSignedIn(User))

{

<li class="nav-item">

<a class="nav-link text-white" asp-area="Identity" asp-page="/Account/Manage/Index" title="Manage">Olá @User.Identity.Name!</a>

</li>

<li class="nav-item">

<form class="form-inline" asp-area="Identity" asp-page="/Account/Logout" asp-route-returnUrl="@Url.Action("Index", "Home", new { area = "" })">

<button type="submit" class="nav-link btn btn-link text-white">Sair</button>

</form>

</li>

}

else

{

<li class="nav-item">

<a class="nav-link text-white" asp-area="Identity" asp-page="/Account/Register">Cadastrar</a>

</li>

<li class="nav-item">

<a class="nav-link text-white" asp-area="Identity" asp-page="/Account/Login">Login</a>

</li>

}

</ul>

**Views – Shared - \_VAlidationScriptsPartial**

<script src="~/lib/jquery-validation/dist/jquery.validate.min.js"></script>

<script src="~/lib/jquery-validation-unobtrusive/jquery.validate.unobtrusive.min.js"></script>

**Views – Shared - Error**

@model ErrorViewModel

@{

ViewData["Title"] = "Erro";

}

<br />

<h1 class="text-danger">@ViewData["Title"]</h1>

<br />

<h2>@Model.Message</h2>

<br />

@if (Model.ShowRequestId)

{

<p>

<strong>Requisição:</strong> <code>@Model.RequestId</code>

</p>

}

**Views – Transacoes – Index**

@model IEnumerable<BlockTechMVC.Models.Transacao>

@{

ViewData["Title"] = "Transações";

}

<br />

<h1>@ViewData["Title"]</h1>

<br />

@if (User.IsInRole("Admin"))

{

<form asp-controller="Transacoes" asp-action="Index" method="get">

<p>

@Html.DropDownList("Busca", null, new

{

onchange = "document.location.href='/Transacoes/Index?Busca='+this.value;"

})

<input type="text" name="SearchString">

<input type="submit" value="Filtrar" />

</p>

</form>

}

else

{

<form asp-controller="Transacoes" asp-action="Index" method="get">

<p>

Criptomoeda:

<input type="text" name="SearchString">

<input type="submit" value="Filtrar" />

</p>

</form>

}

<div class="table-responsive">

<table class="table">

<caption>Lista de Transações</caption>

<thead>

<tr>

<th>

@Html.ActionLink("Data", "Index", new { sortOrder = ViewBag.Data }, new { @class = "imgOrder" })

</th>

@if (User.IsInRole("Admin"))

{

<th>

@Html.ActionLink("Nome", "Index", new { sortOrder = ViewBag.Nome }, new { @class = "imgOrder" })

</th>

}

<th>

@Html.DisplayNameFor(model => model.Tipo)

</th>

<th>

@Html.ActionLink("Criptomoeda", "Index", new { sortOrder = ViewBag.Criptomoeda }, new { @class = "imgOrder" })

</th>

<th>

@Html.ActionLink("Valor", "Index", new { sortOrder = ViewBag.Valor }, new { @class = "imgOrder" })

</th>

<th>

@Html.ActionLink("Quantidade", "Index", new { sortOrder = ViewBag.Quantidade }, new { @class = "imgOrder" })

</th>

<th></th>

</tr>

</thead>

<tbody>

@foreach (var item in Model)

{

<tr>

<td>

@Html.DisplayFor(modelItem => item.Data)

</td>

@if (User.IsInRole("Admin"))

{

<td>

@Html.DisplayFor(modelItem => item.ContaCliente.ApplicationUser.Nome)

</td>

}

<td>

@Html.DisplayFor(modelItem => item.Tipo)

</td>

<td>

@Html.DisplayFor(modelItem => item.CriptomoedaHoje.Criptomoeda.Nome)

</td>

<td>

R$ @Html.DisplayFor(modelItem => item.Valor)

</td>

@if (User.IsInRole("Admin") || (!User.IsInRole("Adimin")))

{

var valor = item.Valor;

var valorCriptomoedaData = item.CriptomoedaHoje.Valor;

var total = (valor / valorCriptomoedaData).ToString("F6");

<td> @total </td>

}

</tr>

}

</tbody>

</table>

</div>

**Views - \_ViewImports**

@using BlockTechMVC

@using BlockTechMVC.Models

@addTagHelper \*, Microsoft.AspNetCore.Mvc.TagHelpers

**Views - \_ViewStart**

@{

Layout = "\_Layout";

}

**Appsettings**

{

"ConnectionStrings": {

"DefaultConnection": "Server=LAPTOP-LBQF810B;Database=BlockTechMVC;Trusted\_Connection=True;MultipleActiveResultSets=true"

},

"Logging": {

"LogLevel": {

"Default": "Information",

"Microsoft": "Warning",

"Microsoft.Hosting.Lifetime": "Information"

}

},

"AllowedHosts": "\*"

}

**Startup**

using System.Collections.Generic;

using Microsoft.AspNetCore.Builder;

using Microsoft.AspNetCore.Identity;

using Microsoft.AspNetCore.Hosting;

using Microsoft.EntityFrameworkCore;

using BlockTechMVC.Data;

using Microsoft.Extensions.Configuration;

using Microsoft.Extensions.DependencyInjection;

using Microsoft.Extensions.Hosting;

using BlockTechMVC.Models;

using System.Globalization;

namespace BlockTechMVC

{

public class Startup

{

public Startup(IConfiguration configuration)

{

Configuration = configuration;

}

public IConfiguration Configuration { get; }

// This method gets called by the runtime. Use this method to add services to the container.

public void ConfigureServices(IServiceCollection services)

{

services.AddDbContext<ApplicationDbContext>(options =>

options.UseSqlServer(

Configuration.GetConnectionString("DefaultConnection")));

services.AddDefaultIdentity<ApplicationUser>(options => options.SignIn.RequireConfirmedAccount = true)

.AddRoles<IdentityRole>()

.AddEntityFrameworkStores<ApplicationDbContext>();

services.AddControllersWithViews();

services.AddRazorPages();

services.AddScoped<SeedingService>();

}

// This method gets called by the runtime. Use this method to configure the HTTP request pipeline.

public void Configure(IApplicationBuilder app, IWebHostEnvironment env, SeedingService seedingService)

{

var enUs = new CultureInfo("en-US");

var localizationOption = new RequestLocalizationOptions

{

DefaultRequestCulture = new Microsoft.AspNetCore.Localization.RequestCulture(enUs),

SupportedCultures = new List<CultureInfo> { enUs },

SupportedUICultures = new List<CultureInfo> { enUs }

};

app.UseRequestLocalization(localizationOption);

if (env.IsDevelopment())

{

app.UseDeveloperExceptionPage();

app.UseDatabaseErrorPage();

seedingService.Seed();

}

else

{

app.UseExceptionHandler("/Home/Error");

// The default HSTS value is 30 days. You may want to change this for production scenarios, see https://aka.ms/aspnetcore-hsts.

app.UseHsts();

}

app.UseHttpsRedirection();

app.UseStaticFiles();

app.UseRouting();

app.UseAuthentication();

app.UseAuthorization();

app.UseEndpoints(endpoints =>

{

endpoints.MapControllerRoute(

name: "default",

pattern: "{controller=Home}/{action=Index}/{id?}");

endpoints.MapRazorPages();

});

}

}

}

**Controller – ContasClientesController**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Rendering;

using Microsoft.EntityFrameworkCore;

using BlockTechMVC.Data;

using BlockTechMVC.Models;

using Microsoft.AspNetCore.Authorization;

using System.Diagnostics;

namespace BlockTechMVC.Controllers

{

[Authorize]

public class ContasClientesController : Controller

{

private readonly ApplicationDbContext \_context;

public ContasClientesController(ApplicationDbContext context)

{

\_context = context;

}

// GET: ContasClientes

public async Task<IActionResult> Index(string searchString, string sortOrder)

{

var user = User.Identity.Name;

ViewBag.Total = SaldoTotal(user);

if (user == "Administrador")

{

double[] usuarios = new double[20];

int i = 0;

foreach (var item in \_context.Saldo)

{

var usuario = (from c in \_context.Saldo

join conta in \_context.ContaCliente

on c.ContaClienteId equals conta.Id

join cliente in \_context.ApplicationUser

on conta.ApplicationUserID equals cliente.Id

where item.ContaClienteId == conta.Id

select cliente.UserName).FirstOrDefault();

var saldo = SaldoTotal(usuario);

usuarios[i] = saldo;

i++;

}

ViewBag.TotalAdm = usuarios.ToList();

}

if (user == "Administrador")

{

ViewBag.NameSortParm = sortOrder == "Nome" ? "Nome\_desc" : "Nome";

ViewBag.ValueSortParm = sortOrder == "Valor" ? "Valor\_desc" : "Valor";

var application = \_context.Saldo

.Include(t => t.ContaCliente)

.Include(t => t.ContaCliente.ApplicationUser);

if (sortOrder != null)

{

var usuario = application.OrderBy(c => c.ContaCliente.ApplicationUser.Nome);

switch (sortOrder)

{

case "Nome":

usuario = application.OrderBy(c => c.ContaCliente.ApplicationUser.Nome);

break;

case "Nome\_desc":

usuario = application.OrderByDescending(c => c.ContaCliente.ApplicationUser.Nome);

break;

case "Valor\_desc":

usuario = application.OrderByDescending(c => c.SaldoAtualRS);

break;

case "Valor":

usuario = application.OrderBy(c => c.SaldoAtualRS);

break;

default:

usuario = application.OrderByDescending(c => c.ContaCliente.ApplicationUser.Nome);

break;

}

return View(await usuario.ToListAsync());

}

if (!String.IsNullOrEmpty(searchString))

{

var usuarioSelecionado = \_context.Saldo

.Include(t => t.ContaCliente)

.Include(t => t.ContaCliente.ApplicationUser)

.Where(t => t.ContaCliente.ApplicationUser.Nome.Contains(searchString));

return View(usuarioSelecionado.ToList());

}

return View(await application.ToListAsync());

}

else

{

var usuario = \_context.Saldo

.Include(t => t.ContaCliente)

.Include(t => t.ContaCliente.ApplicationUser)

.Where(t => t.ContaCliente.ApplicationUser.UserName == user);

return View(await usuario.ToListAsync());

}

}

// GET: ContasClientes/Details/5

public async Task<IActionResult> Details(int? id)

{

var user = User.Identity.Name;

var usuario = (from c in \_context.Saldo

join conta in \_context.ContaCliente

on c.ContaClienteId equals conta.Id

join cliente in \_context.ApplicationUser

on conta.ApplicationUserID equals cliente.Id

where c.Id == id

select cliente.UserName).FirstOrDefault();

ViewBag.TotalAdm = SaldoTotal(usuario);

if (id == null)

{

return RedirectToAction(nameof(Error), new { message = "Transação não encontrada!" });

}

var transacao = await \_context.Transacao

.Include(t => t.ContaCliente)

.Include(t => t.CriptoSaldo)

.Include(t => t.CriptomoedaHoje)

.Include(t => t.ContaCliente.Conta)

.Include(t => t.ContaCliente.ApplicationUser)

.Include(t => t.Saldo)

.FirstOrDefaultAsync(m => m.Saldo.Id == id);

if (transacao == null)

{

return RedirectToAction(nameof(Error), new { message = "Transação não encontrada!" });

}

ViewData["ContaVinculada"] = "Conta Vinculada";

ViewBag.Total = SaldoTotal(user);

return View(transacao);

}

private bool TransacaoExists(int id)

{

return \_context.Transacao.Any(e => e.Id == id);

}

public async Task<IActionResult> Aplicacoes(int? id, string searchString, int? Busca, string sortOrder)

{

var user = User.Identity.Name;

ViewBag.Quantidade = sortOrder == "Quantidade" ? "Quantidade\_desc" : "Quantidade";

ViewBag.Criptomoeda = sortOrder == "Criptomoeda" ? "Criptomoeda\_desc" : "Criptomoeda";

if (user == "Administrador")

{

ViewBag.Nome = sortOrder == "Nome" ? "Nome\_desc" : "Nome";

List<SelectListItem> itens = new List<SelectListItem>();

SelectListItem item1 = new SelectListItem() { Text = "Nome/Razão Social", Value = "1", Selected = true };

SelectListItem item2 = new SelectListItem() { Text = "Criptomoeda", Value = "2", Selected = false };

itens.Add(item1);

itens.Add(item2);

ViewBag.Busca = itens;

var applicationDbContext = \_context.CriptoSaldo

.Include(t => t.ContaCliente)

.Include(t => t.ContaCliente.ApplicationUser);

if (sortOrder != null)

{

var orderName = applicationDbContext.OrderBy(t => t.ContaCliente.ApplicationUser.Nome);

switch (sortOrder)

{

case "Nome\_desc":

orderName = orderName.OrderByDescending(s => s.ContaCliente.ApplicationUser.Nome);

break;

case "Nome":

orderName = applicationDbContext.OrderBy(s => s.ContaCliente.ApplicationUser.Nome);

break;

case "Quantidade":

orderName = applicationDbContext.OrderBy(s => s.Quantidade);

break;

case "Quantidade\_desc":

orderName = applicationDbContext.OrderByDescending(s => s.Quantidade);

break;

case "Criptomoeda":

orderName = applicationDbContext.OrderBy(s => s.Criptomoeda);

break;

case "Criptomoeda\_desc":

orderName = applicationDbContext.OrderByDescending(s => s.Criptomoeda);

break;

default:

orderName = applicationDbContext.OrderBy(s => s.ContaCliente.ApplicationUser.Nome);

break;

};

return View(orderName.ToList());

}

if (Busca != null)

{

itens.Where(i => i.Value == Busca.ToString()).First().Selected = true;

}

if (Busca == 1)

{

if (!String.IsNullOrEmpty(searchString))

{

var usuarioSelecionado = \_context.CriptoSaldo

.Include(t => t.ContaCliente)

.Include(t => t.ContaCliente.ApplicationUser)

.Where(t => t.ContaCliente.ApplicationUser.Nome.Contains(searchString));

return View(usuarioSelecionado.ToList());

}

}

if (Busca == 2)

{

if (!String.IsNullOrEmpty(searchString))

{

var usuarioSelecionado = \_context.CriptoSaldo

.Include(t => t.ContaCliente)

.Include(t => t.ContaCliente.ApplicationUser)

.Where(t => t.Criptomoeda.Contains(searchString));

return View(usuarioSelecionado.ToList());

}

}

return View(await applicationDbContext.ToListAsync());

}

else

{

var usuario = \_context.CriptoSaldo

.Where(t => t.ContaCliente.ApplicationUser.UserName == user)

.Include(t => t.ContaCliente)

.Where(t => t.ContaCliente.ApplicationUser.UserName == user);

if (sortOrder != null)

{

var orderName = usuario.OrderBy(t => t.Criptomoeda);

switch (sortOrder)

{

case "Quantidade":

orderName = usuario.OrderBy(s => s.Quantidade);

break;

case "Quantidade\_desc":

orderName = usuario.OrderByDescending(s => s.Quantidade);

break;

case "Criptomoeda":

orderName = usuario.OrderBy(s => s.Criptomoeda);

break;

case "Criptomoeda\_desc":

orderName = usuario.OrderByDescending(s => s.Criptomoeda);

break;

default:

orderName = usuario.OrderBy(s => s.Criptomoeda);

break;

};

return View(orderName.ToList());

}

if (!String.IsNullOrEmpty(searchString))

{

var criptoSelecionada = \_context.CriptoSaldo

.Include(t => t.ContaCliente)

.Include(t => t.ContaCliente.ApplicationUser)

.Where(t => t.Criptomoeda.Contains(searchString) && (t.ContaCliente.ApplicationUser.UserName == user));

return View(criptoSelecionada.ToList());

}

return View(await usuario.ToListAsync());

}

}

public IActionResult Error(string message)

{

var viewModel = new ErrorViewModel

{

Message = message,

RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier

};

return View(viewModel);

}

public double SaldoTotal(string user)

{

MeusInvestimentosController investimentos = new MeusInvestimentosController(\_context);

var saldoSemInvestimento = investimentos.SaldoSemInvestimento(user);

double bitcoin = investimentos.QuantidadeTotalCriptomoeda("Bitcoin", user);

double saldoTotalBitcoin = investimentos.CalcularSaldoAtual(bitcoin, "Bitcoin");

double ethereum = investimentos.QuantidadeTotalCriptomoeda("Ethereum", user);

double saldoTotalEthereum = investimentos.CalcularSaldoAtual(ethereum, "Ethereum");

double bitcoinCash = investimentos.QuantidadeTotalCriptomoeda("Bitcoin Cash", user);

double saldoTotalBitcoinCash = investimentos.CalcularSaldoAtual(bitcoinCash, "Bitcoin Cash");

double xrp = investimentos.QuantidadeTotalCriptomoeda("XRP", user);

double saldoTotalXrp = investimentos.CalcularSaldoAtual(xrp, "XRP");

double paxGold = investimentos.QuantidadeTotalCriptomoeda("PAX Gold", user);

double saldoTotalPaxGold = investimentos.CalcularSaldoAtual(paxGold, "PAX Gold");

double litecoin = investimentos.QuantidadeTotalCriptomoeda("Litecoin", user);

double saldoTotalLitecoin = investimentos.CalcularSaldoAtual(litecoin, "Litecoin");

var total = (saldoSemInvestimento + saldoTotalBitcoin + saldoTotalEthereum + saldoTotalBitcoinCash + saldoTotalXrp + saldoTotalPaxGold + saldoTotalLitecoin).ToString("F2");

ViewBag.Total = total;

return Convert.ToDouble(total);

}

}

}

**Controller – CriptomoedasController**

using System;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using BlockTechMVC.Data;

using BlockTechMVC.Models;

using Microsoft.AspNetCore.Authorization;

using System.Diagnostics;

namespace BlockTechMVC.Controllers

{

//Controller apenas para ADM adicionar, remover e alterar Criptomoedas (nome e símbolo)

[Authorize(Roles = "Admin")]

public class CriptomoedasController : Controller

{

private readonly ApplicationDbContext \_context;

public CriptomoedasController(ApplicationDbContext context)

{

\_context = context;

}

// GET: Criptomoedas

public async Task<IActionResult> Index(string searchString, string sortOrder)

{

ViewBag.NameSortParm = sortOrder == "Nome" ? "Nome\_desc" : "Nome";

var criptomoedas = from c in \_context.Criptomoeda

select c;

switch (sortOrder)

{

case "Nome\_desc":

criptomoedas = criptomoedas.OrderByDescending(s => s.Nome);

break;

default:

criptomoedas = criptomoedas.OrderBy(s => s.Nome);

break;

}

if (!String.IsNullOrEmpty(searchString))

{

criptomoedas = criptomoedas.Where(c => c.Nome.Contains(searchString));

}

return View(await criptomoedas.ToListAsync());

}

// GET: Criptomoedas/Details/5

public async Task<IActionResult> Details(int? id)

{

if (id == null)

{

return RedirectToAction(nameof(Error), new { message = "Id não encontrado!" });

}

var criptomoeda = await \_context.Criptomoeda

.FirstOrDefaultAsync(m => m.Id == id);

if (criptomoeda == null)

{

return RedirectToAction(nameof(Error), new { message = "Id não encontrado!" });

}

return View(criptomoeda);

}

// GET: Criptomoedas/Create

public IActionResult Create()

{

return View();

}

// POST: Criptomoedas/Create

// To protect from overposting attacks, enable the specific properties you want to bind to, for

// more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create([Bind("Id,Nome,Simbolo,Cadastro")] Criptomoeda criptomoeda)

{

if (\_context.Criptomoeda.Any(c => c.Nome == criptomoeda.Nome)) return RedirectToAction(nameof(Error), new { message = "Criptomoeda já cadastrada!" });

if (\_context.Criptomoeda.Any(c => c.Simbolo == criptomoeda.Simbolo)) return RedirectToAction(nameof(Error), new { message = "Criptomoeda já cadastrada!" });

if (ModelState.IsValid)

{

\_context.Add(criptomoeda);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

return View(criptomoeda);

}

// GET: Criptomoedas/Edit/5

public async Task<IActionResult> Edit(int? id)

{

if (id == null)

{

return RedirectToAction(nameof(Error), new { message = "Id não encontrado!" });

}

var criptomoeda = await \_context.Criptomoeda.FindAsync(id);

if (criptomoeda == null)

{

return RedirectToAction(nameof(Error), new { message = "Id não encontrado!" });

}

return View(criptomoeda);

}

// POST: Criptomoedas/Edit/5

// To protect from overposting attacks, enable the specific properties you want to bind to, for

// more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(int id, [Bind("Id,Nome,Simbolo,Cadastro")] Criptomoeda criptomoeda)

{

if (id != criptomoeda.Id)

{

return RedirectToAction(nameof(Error), new { message = "Id não encontrado!" });

}

if (ModelState.IsValid)

{

try

{

\_context.Update(criptomoeda);

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!CriptomoedaExists(criptomoeda.Id))

{

return RedirectToAction(nameof(Error), new { message = "Criptomoeda não encontrada!" });

}

else

{

return RedirectToAction(nameof(Error), new { message = "" });

}

}

return RedirectToAction(nameof(Index));

}

return View(criptomoeda);

}

// GET: Criptomoedas/Delete/5

public async Task<IActionResult> Delete(int? id)

{

if (id == null)

{

return RedirectToAction(nameof(Error), new { message = "Id não encontrado!" });

}

var criptomoeda = await \_context.Criptomoeda

.FirstOrDefaultAsync(m => m.Id == id);

if (criptomoeda == null)

{

return RedirectToAction(nameof(Error), new { message = "Criptomoeda não encontrada!" });

}

return View(criptomoeda);

}

// POST: Criptomoedas/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{

var criptomoeda = await \_context.Criptomoeda.FindAsync(id);

\_context.Criptomoeda.Remove(criptomoeda);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

private bool CriptomoedaExists(int id)

{

return \_context.Criptomoeda.Any(e => e.Id == id);

}

public IActionResult Error(string message)

{

var viewModel = new ErrorViewModel

{

Message = message,

RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier

};

return View(viewModel);

}

}

}

**Controller – CriptomoedasHojeController**

using System;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.EntityFrameworkCore;

using BlockTechMVC.Data;

using BlockTechMVC.Models;

using Microsoft.AspNetCore.Authorization;

using System.Diagnostics;

namespace BlockTechMVC.Controllers

{

//Controller apenas para ADM adicionar, remover e alterar Criptomoedas (nome e símbolo)

[Authorize(Roles = "Admin")]

public class CriptomoedasController : Controller

{

private readonly ApplicationDbContext \_context;

public CriptomoedasController(ApplicationDbContext context)

{

\_context = context;

}

// GET: Criptomoedas

public async Task<IActionResult> Index(string searchString, string sortOrder)

{

ViewBag.NameSortParm = sortOrder == "Nome" ? "Nome\_desc" : "Nome";

var criptomoedas = from c in \_context.Criptomoeda

select c;

switch (sortOrder)

{

case "Nome\_desc":

criptomoedas = criptomoedas.OrderByDescending(s => s.Nome);

break;

default:

criptomoedas = criptomoedas.OrderBy(s => s.Nome);

break;

}

if (!String.IsNullOrEmpty(searchString))

{

criptomoedas = criptomoedas.Where(c => c.Nome.Contains(searchString));

}

return View(await criptomoedas.ToListAsync());

}

// GET: Criptomoedas/Details/5

public async Task<IActionResult> Details(int? id)

{

if (id == null)

{

return RedirectToAction(nameof(Error), new { message = "Id não encontrado!" });

}

var criptomoeda = await \_context.Criptomoeda

.FirstOrDefaultAsync(m => m.Id == id);

if (criptomoeda == null)

{

return RedirectToAction(nameof(Error), new { message = "Id não encontrado!" });

}

return View(criptomoeda);

}

// GET: Criptomoedas/Create

public IActionResult Create()

{

return View();

}

// POST: Criptomoedas/Create

// To protect from overposting attacks, enable the specific properties you want to bind to, for

// more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Create([Bind("Id,Nome,Simbolo,Cadastro")] Criptomoeda criptomoeda)

{

if (\_context.Criptomoeda.Any(c => c.Nome == criptomoeda.Nome)) return RedirectToAction(nameof(Error), new { message = "Criptomoeda já cadastrada!" });

if (\_context.Criptomoeda.Any(c => c.Simbolo == criptomoeda.Simbolo)) return RedirectToAction(nameof(Error), new { message = "Criptomoeda já cadastrada!" });

if (ModelState.IsValid)

{

\_context.Add(criptomoeda);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

return View(criptomoeda);

}

// GET: Criptomoedas/Edit/5

public async Task<IActionResult> Edit(int? id)

{

if (id == null)

{

return RedirectToAction(nameof(Error), new { message = "Id não encontrado!" });

}

var criptomoeda = await \_context.Criptomoeda.FindAsync(id);

if (criptomoeda == null)

{

return RedirectToAction(nameof(Error), new { message = "Id não encontrado!" });

}

return View(criptomoeda);

}

// POST: Criptomoedas/Edit/5

// To protect from overposting attacks, enable the specific properties you want to bind to, for

// more details, see http://go.microsoft.com/fwlink/?LinkId=317598.

[HttpPost]

[ValidateAntiForgeryToken]

public async Task<IActionResult> Edit(int id, [Bind("Id,Nome,Simbolo,Cadastro")] Criptomoeda criptomoeda)

{

if (id != criptomoeda.Id)

{

return RedirectToAction(nameof(Error), new { message = "Id não encontrado!" });

}

if (ModelState.IsValid)

{

try

{

\_context.Update(criptomoeda);

await \_context.SaveChangesAsync();

}

catch (DbUpdateConcurrencyException)

{

if (!CriptomoedaExists(criptomoeda.Id))

{

return RedirectToAction(nameof(Error), new { message = "Criptomoeda não encontrada!" });

}

else

{

return RedirectToAction(nameof(Error), new { message = "" });

}

}

return RedirectToAction(nameof(Index));

}

return View(criptomoeda);

}

// GET: Criptomoedas/Delete/5

public async Task<IActionResult> Delete(int? id)

{

if (id == null)

{

return RedirectToAction(nameof(Error), new { message = "Id não encontrado!" });

}

var criptomoeda = await \_context.Criptomoeda

.FirstOrDefaultAsync(m => m.Id == id);

if (criptomoeda == null)

{

return RedirectToAction(nameof(Error), new { message = "Criptomoeda não encontrada!" });

}

return View(criptomoeda);

}

// POST: Criptomoedas/Delete/5

[HttpPost, ActionName("Delete")]

[ValidateAntiForgeryToken]

public async Task<IActionResult> DeleteConfirmed(int id)

{

var criptomoeda = await \_context.Criptomoeda.FindAsync(id);

\_context.Criptomoeda.Remove(criptomoeda);

await \_context.SaveChangesAsync();

return RedirectToAction(nameof(Index));

}

private bool CriptomoedaExists(int id)

{

return \_context.Criptomoeda.Any(e => e.Id == id);

}

public IActionResult Error(string message)

{

var viewModel = new ErrorViewModel

{

Message = message,

RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier

};

return View(viewModel);

}

}

}

**Controller – HomeController**

using System;

using System.Collections.Generic;

using System.Diagnostics;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.Extensions.Logging;

using BlockTechMVC.Models;

using Microsoft.AspNetCore.Authorization;

namespace BlockTechMVC.Controllers

{

public class HomeController : Controller

{

private readonly ILogger<HomeController> \_logger;

public HomeController(ILogger<HomeController> logger)

{

\_logger = logger;

}

public IActionResult Index()

{

return View();

}

[ResponseCache(Duration = 0, Location = ResponseCacheLocation.None, NoStore = true)]

public IActionResult Error()

{

return View(new ErrorViewModel { RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier });

}

}

}

**Controller – MeusInvestimentosController**

using System;

using System.Collections.Generic;

using System.Linq;

using Microsoft.AspNetCore.Mvc;

using BlockTechMVC.Data;

using BlockTechMVC.Models.Enums;

using Microsoft.AspNetCore.Authorization;

namespace BlockTechMVC.Controllers

{

[Authorize]

public class MeusInvestimentosController : Controller

{

private readonly ApplicationDbContext \_context;

public MeusInvestimentosController(ApplicationDbContext context)

{

\_context = context;

}

// GET: MeusInvestimentos

public IActionResult Index()

{

var user = User.Identity.Name;

if (user == "Administrador")

{

var Bitcoin = QuantidadeTotalCriptomoedaAdm("Bitcoin");

ViewBag.Bitcoin = Bitcoin;

ViewBag.BitcoinValorRS = ValorInvestidoAdm("Bitcoin").ToString("F2"); ;

var Ethereum = QuantidadeTotalCriptomoedaAdm("Ethereum");

ViewBag.Ethereum = Ethereum;

ViewBag.EthereumValorRS = ValorInvestidoAdm("Ethereum").ToString("F2");

var BitcoinCash = QuantidadeTotalCriptomoedaAdm("Bitcoin Cash");

ViewBag.BitcoinCash = BitcoinCash;

ViewBag.BitcoinCashValorRS = ValorInvestidoAdm("Bitcoin Cash").ToString("F2");

var Litecoin = QuantidadeTotalCriptomoedaAdm("Litecoin");

ViewBag.Litecoin = Litecoin;

ViewBag.LitecoinValorRS = ValorInvestidoAdm("Litecoin").ToString("F2");

var PaxGold = QuantidadeTotalCriptomoedaAdm("PAX Gold");

ViewBag.PaxGold = PaxGold;

ViewBag.PaxGoldValorRS = ValorInvestidoAdm("PAX Gold").ToString("F2");

var Xrp = QuantidadeTotalCriptomoedaAdm("XRP");

ViewBag.Xrp = Xrp;

ViewBag.XrpValorRS = ValorInvestidoAdm("XRP").ToString("F2");

return View();

}

else

{

double bitcoin = QuantidadeTotalCriptomoeda("Bitcoin", user);

ViewBag.QuantidadaTotalBitcoin = bitcoin.ToString("F6");

double saldoTotalBitcoin = CalcularSaldoAtual(bitcoin, "Bitcoin");

ViewBag.QuantidadaEmRealBitcoin = saldoTotalBitcoin.ToString("F2");

double valorInvestidoBitcoin = ValorInvestido("Bitcoin", user);

ViewBag.ValorInvestidoBitcoin = valorInvestidoBitcoin.ToString("F2");

double ethereum = QuantidadeTotalCriptomoeda("Ethereum", user);

ViewBag.QuantidadaTotalEthereum = ethereum.ToString("F6");

double saldoTotalEthereum = CalcularSaldoAtual(ethereum, "Ethereum");

ViewBag.QuantidadaEmRealEthereum = saldoTotalEthereum.ToString("F2");

double valorInvestidoEthereum = ValorInvestido("Ethereum", user);

ViewBag.ValorInvestidoEthereum = valorInvestidoEthereum.ToString("F2");

double bitcoinCash = QuantidadeTotalCriptomoeda("Bitcoin Cash", user);

ViewBag.QuantidadaTotalBitcoinCash = bitcoinCash.ToString("F6");

double saldoTotalBitcoinCash = CalcularSaldoAtual(bitcoinCash, "Bitcoin Cash");

ViewBag.QuantidadaEmRealBitcoinCash = saldoTotalBitcoinCash.ToString("F2");

double valorInvestidoBitcoinCash = ValorInvestido("Bitcoin Cash", user);

ViewBag.ValorInvestidoBitcoinCash = valorInvestidoBitcoinCash.ToString("F2");

double xrp = QuantidadeTotalCriptomoeda("XRP", user);

ViewBag.QuantidadaTotalXrp = xrp.ToString("F6");

double saldoTotalXrp = CalcularSaldoAtual(xrp, "XRP");

ViewBag.QuantidadaEmRealXrp = saldoTotalXrp.ToString("F2");

double valorInvestidoXrp = ValorInvestido("XRP", user);

ViewBag.ValorInvestidoXrp = valorInvestidoXrp.ToString("F2");

double paxGold = QuantidadeTotalCriptomoeda("PAX Gold", user);

ViewBag.QuantidadaTotalPaxGold = paxGold.ToString("F6");

double saldoTotalPaxGold = CalcularSaldoAtual(paxGold, "PAX Gold");

ViewBag.QuantidadaEmRealPaxGold = saldoTotalPaxGold.ToString("F2");

double valorInvestidoPaxGold = ValorInvestido("PAX Gold", user);

ViewBag.ValorInvestidoPaxGold = valorInvestidoPaxGold.ToString("F2");

double litecoin = QuantidadeTotalCriptomoeda("Litecoin", user);

ViewBag.QuantidadaTotalLitecoin = litecoin.ToString("F6");

double saldoTotalLitecoin = CalcularSaldoAtual(litecoin, "Litecoin");

ViewBag.QuantidadaEmRealLitecoin = saldoTotalLitecoin.ToString("F2");

double valorInvestidoLitecoin = ValorInvestido("Litecoin", user);

ViewBag.ValorInvestidoLitecoin = valorInvestidoLitecoin.ToString("F2");

double saldoSemInvestimento = SaldoSemInvestimento(user);

ViewBag.SaldoSemInvestimento = saldoSemInvestimento.ToString("F2");

double dinheiroTotalConta = saldoSemInvestimento + saldoTotalBitcoin + saldoTotalEthereum + saldoTotalBitcoinCash + saldoTotalXrp + saldoTotalPaxGold + saldoTotalLitecoin;

ViewBag.DinheiroTotalContaRS = dinheiroTotalConta.ToString("F2");

return View();

}

}

public double QuantidadeTotalCriptomoeda(string criptomoeda, string user)

{

var investimentos = (from transacoes in \_context.Transacao

join conta in \_context.ContaCliente

on transacoes.ContaClienteId equals conta.Id

join usuario in \_context.ApplicationUser

on conta.ApplicationUserID equals usuario.Id

join criptomoedahoje in \_context.CriptomoedaHoje

on transacoes.CriptomoedaHojeId equals criptomoedahoje.Id

join criptomoedas in \_context.Criptomoeda

on criptomoedahoje.CriptomoedaId equals criptomoedas.Id

join criptosaldo in \_context.CriptoSaldo

on transacoes.CriptoSaldoId equals criptosaldo.Id

where criptomoedas.Nome == criptomoeda &&

usuario.UserName == user

select criptosaldo.Quantidade).FirstOrDefault();

return investimentos;

}

public double CalcularSaldoAtual(double quantidadeCriptomoeda, string criptomoeda)

{

var valorAtual = (from criptohoje in \_context.CriptomoedaHoje

join cripto in \_context.Criptomoeda

on criptohoje.CriptomoedaId equals cripto.Id

where cripto.Nome == criptomoeda &&

criptohoje.Data == DateTime.Today

select criptohoje.Valor).Single();

return quantidadeCriptomoeda \* valorAtual;

}

public double ValorInvestido(string criptomoeda, string user)

{

var investimentos = (from transacoes in \_context.Transacao

join conta in \_context.ContaCliente

on transacoes.ContaClienteId equals conta.Id

join usuario in \_context.ApplicationUser

on conta.ApplicationUserID equals usuario.Id

join criptomoedahoje in \_context.CriptomoedaHoje

on transacoes.CriptomoedaHojeId equals criptomoedahoje.Id

join criptomoedas in \_context.Criptomoeda

on criptomoedahoje.CriptomoedaId equals criptomoedas.Id

join criptosaldo in \_context.CriptoSaldo

on transacoes.CriptoSaldoId equals criptosaldo.Id

where criptomoedas.Nome == criptomoeda &&

usuario.UserName == user &&

transacoes.Tipo.Equals(TipoTransacao.Compra)

select transacoes.Valor).Sum();

var investimentosVenda = (from transacoes in \_context.Transacao

join conta in \_context.ContaCliente

on transacoes.ContaClienteId equals conta.Id

join usuario in \_context.ApplicationUser

on conta.ApplicationUserID equals usuario.Id

join criptomoedahoje in \_context.CriptomoedaHoje

on transacoes.CriptomoedaHojeId equals criptomoedahoje.Id

join criptomoedas in \_context.Criptomoeda

on criptomoedahoje.CriptomoedaId equals criptomoedas.Id

join criptosaldo in \_context.CriptoSaldo

on transacoes.CriptoSaldoId equals criptosaldo.Id

where criptomoedas.Nome == criptomoeda &&

usuario.UserName == user &&

transacoes.Tipo.Equals(TipoTransacao.Venda)

select transacoes.Valor).Sum();

return investimentos - investimentosVenda;

}

public double ValorInvestidoAdm(string criptomoeda)

{

var investimentos = (from transacoes in \_context.Transacao

join conta in \_context.ContaCliente

on transacoes.ContaClienteId equals conta.Id

join criptomoedahoje in \_context.CriptomoedaHoje

on transacoes.CriptomoedaHojeId equals criptomoedahoje.Id

join criptomoedas in \_context.Criptomoeda

on criptomoedahoje.CriptomoedaId equals criptomoedas.Id

join criptosaldo in \_context.CriptoSaldo

on transacoes.CriptoSaldoId equals criptosaldo.Id

where criptomoedas.Nome == criptomoeda &&

transacoes.Tipo.Equals(TipoTransacao.Compra)

select transacoes.Valor).Sum();

var investimentosVenda = (from transacoes in \_context.Transacao

join conta in \_context.ContaCliente

on transacoes.ContaClienteId equals conta.Id

join criptomoedahoje in \_context.CriptomoedaHoje

on transacoes.CriptomoedaHojeId equals criptomoedahoje.Id

join criptomoedas in \_context.Criptomoeda

on criptomoedahoje.CriptomoedaId equals criptomoedas.Id

join criptosaldo in \_context.CriptoSaldo

on transacoes.CriptoSaldoId equals criptosaldo.Id

where criptomoedas.Nome == criptomoeda &&

transacoes.Tipo.Equals(TipoTransacao.Venda)

select transacoes.Valor).Sum();

return investimentos - investimentosVenda;

}

public double SaldoSemInvestimento(string user)

{

var saldoNaoInvestido = (from transacoes in \_context.Transacao

join conta in \_context.ContaCliente

on transacoes.ContaClienteId equals conta.Id

join usuario in \_context.ApplicationUser

on conta.ApplicationUserID equals usuario.Id

join criptomoedahoje in \_context.CriptomoedaHoje

on transacoes.CriptomoedaHojeId equals criptomoedahoje.Id

join criptomoedas in \_context.Criptomoeda

on criptomoedahoje.CriptomoedaId equals criptomoedas.Id

join criptosaldo in \_context.CriptoSaldo

on transacoes.CriptoSaldoId equals criptosaldo.Id

join saldo in \_context.Saldo

on transacoes.SaldoId equals saldo.Id

where usuario.UserName == user

select saldo.SaldoAtualRS).FirstOrDefault();

return saldoNaoInvestido;

}

public ActionResult Bitcoin()

{

var user = User.Identity.Name;

double bitcoin = QuantidadeTotalCriptomoeda("Bitcoin", user);

ViewBag.QuantidadaTotalBitcoin = bitcoin.ToString("F6");

double saldoTotalBitcoin = CalcularSaldoAtual(bitcoin, "Bitcoin");

ViewBag.QuantidadaEmRealBitcoin = saldoTotalBitcoin.ToString("F2");

double valorInvestidoBitcoin = ValorInvestido("Bitcoin", user);

ViewBag.ValorInvestidoBitcoin = valorInvestidoBitcoin.ToString("F2");

ViewBag.LucroOuPerda = (saldoTotalBitcoin - valorInvestidoBitcoin).ToString("F2");

ViewBag.ValorBitcoin7Dias = Valores7Dias("Bitcoin", bitcoin);

ViewBag.Ultimos7Dias = Ultimos7Dias();

double valorInvestidoEthereum = ValorInvestido("Ethereum", user);

ViewBag.ValorInvestidoEthereum = valorInvestidoEthereum.ToString("F2");

double valorInvestidoBitcoinCash = ValorInvestido("Bitcoin Cash", user);

ViewBag.ValorInvestidoBitcoinCash = valorInvestidoBitcoinCash.ToString("F2");

double valorInvestidoXrp = ValorInvestido("XRP", user);

ViewBag.ValorInvestidoXrp = valorInvestidoXrp.ToString("F2");

double valorInvestidoPaxGold = ValorInvestido("PAX Gold", user);

ViewBag.ValorInvestidoPaxGold = valorInvestidoPaxGold.ToString("F2");

double valorInvestidoLitecoin = ValorInvestido("Litecoin", user);

ViewBag.ValorInvestidoLitecoin = valorInvestidoLitecoin.ToString("F2");

var Bitcoin = QuantidadeTotalCriptomoedaAdm("Bitcoin");

ViewBag.Bitcoin = Bitcoin;

ViewBag.Ultimos7DiasAdm = Valores7DiasAdm("Bitcoin", Bitcoin);

var BitcoinValorRSString = ValorTotalCriptomoedaAdm("Bitcoin", Bitcoin).ToString("F2");

var BitcoinValorRS = Convert.ToDouble(BitcoinValorRSString);

ViewBag.BitcoinValorRS = BitcoinValorRS;

var BitcoinInvestidoString = ValorInvestidoAdm("Bitcoin").ToString("F2");

var BitcoinInvestido = Convert.ToDouble(BitcoinInvestidoString);

ViewBag.BitcoinInvestido = BitcoinInvestido;

ViewBag.LucroOuPerdaAdm = (BitcoinValorRS - BitcoinInvestido).ToString("F2");

int primeiroInvestimento = DataPrimeiroInvestimento("Bitcoin", user);

if (primeiroInvestimento <= 30)

{

ViewBag.UltimoMes = UltimosDias(primeiroInvestimento);

ViewBag.ValorBitcoinMes = ValoresDias("Bitcoin", bitcoin, primeiroInvestimento);

} else

{

ViewBag.UltimoMes = Ultimos30Dias();

ViewBag.ValorBitcoinMes = Valores30Dias("Bitcoin", bitcoin);

}

int primeiroInvestimentoGeralAdm = DataPrimeiroInvestimentoAdm("Bitcoin");

if (primeiroInvestimentoGeralAdm <= 30)

{

ViewBag.UltimoMesAdm = UltimosDias(primeiroInvestimentoGeralAdm);

ViewBag.ValorBitcoinMesAdm = ValoresDias("Bitcoin", Bitcoin, primeiroInvestimentoGeralAdm);

} else

{

ViewBag.UltimoMesAdm = Ultimos30Dias();

ViewBag.ValorBitcoinMesAdm = Valores30Dias("Bitcoin", Bitcoin);

}

return View();

}

public ActionResult Ethereum()

{

var user = User.Identity.Name;

double ethereum = QuantidadeTotalCriptomoeda("Ethereum", user);

ViewBag.QuantidadaTotalEthereum = ethereum.ToString("F6");

double saldoTotalEthereum = CalcularSaldoAtual(ethereum, "Ethereum");

ViewBag.QuantidadaEmRealEthereum = saldoTotalEthereum.ToString("F2");

double valorInvestidoEthereum = ValorInvestido("Ethereum", user);

ViewBag.ValorInvestidoEthereum = valorInvestidoEthereum.ToString("F2");

ViewBag.LucroOuPerda = (saldoTotalEthereum - valorInvestidoEthereum).ToString("F2");

ViewBag.ValorEthereum7Dias = Valores7Dias("Ethereum", ethereum);

ViewBag.Ultimos7Dias = Ultimos7Dias();

double valorInvestidoBitcoin = ValorInvestido("Bitcoin", user);

ViewBag.ValorInvestidoBitcoin = valorInvestidoBitcoin.ToString("F2");

double valorInvestidoBitcoinCash = ValorInvestido("Bitcoin Cash", user);

ViewBag.ValorInvestidoBitcoinCash = valorInvestidoBitcoinCash.ToString("F2");

double valorInvestidoXrp = ValorInvestido("XRP", user);

ViewBag.ValorInvestidoXrp = valorInvestidoXrp.ToString("F2");

double valorInvestidoPaxGold = ValorInvestido("PAX Gold", user);

ViewBag.ValorInvestidoPaxGold = valorInvestidoPaxGold.ToString("F2");

double valorInvestidoLitecoin = ValorInvestido("Litecoin", user);

ViewBag.ValorInvestidoLitecoin = valorInvestidoLitecoin.ToString("F2");

var Ethereum = QuantidadeTotalCriptomoedaAdm("Ethereum");

ViewBag.Ethereum = Ethereum;

ViewBag.Ultimos7DiasAdm = Valores7DiasAdm("Ethereum", Ethereum);

var EthereumValorRString = ValorTotalCriptomoedaAdm("Ethereum", Ethereum).ToString("F2");

double EthereumValorRS = Convert.ToDouble(EthereumValorRString);

ViewBag.EthereumValorRS = EthereumValorRS;

var EthereumInvestidoString = ValorInvestidoAdm("Ethereum").ToString("F2");

double EthereumInvestido = Convert.ToDouble(EthereumInvestidoString);

ViewBag.EthereumInvestido = EthereumInvestido;

ViewBag.LucroOuPerdaAdm = (EthereumValorRS - EthereumInvestido).ToString("F2");

int primeiroInvestimento = DataPrimeiroInvestimento("Ethereum", user);

if (primeiroInvestimento <= 30)

{

ViewBag.UltimoMes = UltimosDias(primeiroInvestimento);

ViewBag.ValorMes = ValoresDias("Ethereum", ethereum, primeiroInvestimento);

}

else

{

ViewBag.UltimoMes = Ultimos30Dias();

ViewBag.ValorMes = Valores30Dias("Ethereum", ethereum);

}

int primeiroInvestimentoGeralAdm = DataPrimeiroInvestimentoAdm("Ethereum");

if (primeiroInvestimentoGeralAdm <= 30)

{

ViewBag.UltimoMesAdm = UltimosDias(primeiroInvestimentoGeralAdm);

ViewBag.ValorMesAdm = ValoresDias("Ethereum", Ethereum, primeiroInvestimentoGeralAdm);

}

else

{

ViewBag.UltimoMesAdm = Ultimos30Dias();

ViewBag.ValorMesAdm = Valores30Dias("Ethereum", Ethereum);

}

return View();

}

public ActionResult BitcoinCash()

{

var user = User.Identity.Name;

double bitcoinCash = QuantidadeTotalCriptomoeda("Bitcoin Cash", user);

ViewBag.QuantidadaTotalBitcoinCash = bitcoinCash.ToString("F6");

double saldoTotalBitcoinCash = CalcularSaldoAtual(bitcoinCash, "Bitcoin Cash");

ViewBag.QuantidadaEmRealBitcoinCash = saldoTotalBitcoinCash.ToString("F2");

double valorInvestidoBitcoinCash = ValorInvestido("Bitcoin Cash", user);

ViewBag.ValorInvestidoBitcoinCash = valorInvestidoBitcoinCash.ToString("F2");

ViewBag.LucroOuPerda = (saldoTotalBitcoinCash - valorInvestidoBitcoinCash).ToString("F2");

ViewBag.ValorBitcoinCash7Dias = Valores7Dias("Bitcoin Cash", bitcoinCash);

ViewBag.Ultimos7Dias = Ultimos7Dias();

double valorInvestidoBitcoin = ValorInvestido("Bitcoin", user);

ViewBag.ValorInvestidoBitcoin = valorInvestidoBitcoin.ToString("F2");

double valorInvestidoEthereum = ValorInvestido("Ethereum", user);

ViewBag.ValorInvestidoEthereum = valorInvestidoEthereum.ToString("F2");

double valorInvestidoXrp = ValorInvestido("XRP", user);

ViewBag.ValorInvestidoXrp = valorInvestidoXrp.ToString("F2");

double valorInvestidoPaxGold = ValorInvestido("PAX Gold", user);

ViewBag.ValorInvestidoPaxGold = valorInvestidoPaxGold.ToString("F2");

double valorInvestidoLitecoin = ValorInvestido("Litecoin", user);

ViewBag.ValorInvestidoLitecoin = valorInvestidoLitecoin.ToString("F2");

var BitcoinCash = QuantidadeTotalCriptomoedaAdm("Bitcoin Cash");

ViewBag.BitcoinCash = BitcoinCash;

ViewBag.Ultimos7DiasAdm = Valores7DiasAdm("Bitcoin Cash", BitcoinCash);

var BitcoinCashValorRSString = ValorTotalCriptomoedaAdm("Bitcoin Cash", BitcoinCash).ToString("F2");

double BitcoinCashValorRS = Convert.ToDouble(BitcoinCashValorRSString);

ViewBag.BitcoinCashValorRS = BitcoinCashValorRS;

var BitcoinCashInvestidoString = ValorInvestidoAdm("Bitcoin Cash").ToString("F2");

double BitcoinCashInvestido = Convert.ToDouble(BitcoinCashInvestidoString);

ViewBag.BitcoinCashInvestido = BitcoinCashInvestido;

ViewBag.LucroOuPerdaAdm = (BitcoinCashValorRS - BitcoinCashInvestido).ToString("F2");

int primeiroInvestimento = DataPrimeiroInvestimento("Bitcoin Cash", user);

if (primeiroInvestimento <= 30)

{

ViewBag.UltimoMes = UltimosDias(primeiroInvestimento);

ViewBag.ValorMes = ValoresDias("Bitcoin Cash", bitcoinCash, primeiroInvestimento);

}

else

{

ViewBag.UltimoMes = Ultimos30Dias();

ViewBag.ValorMes = Valores30Dias("Bitcoin Cash", bitcoinCash);

}

int primeiroInvestimentoGeralAdm = DataPrimeiroInvestimentoAdm("Bitcoin Cash");

if (primeiroInvestimentoGeralAdm <= 30)

{

ViewBag.UltimoMesAdm = UltimosDias(primeiroInvestimentoGeralAdm);

ViewBag.ValorMesAdm = ValoresDias("Bitcoin Cash", BitcoinCash, primeiroInvestimentoGeralAdm);

}

else

{

ViewBag.UltimoMesAdm = Ultimos30Dias();

ViewBag.ValorMesAdm = Valores30Dias("Bitcoin Cash", BitcoinCash);

}

return View();

}

public ActionResult Xrp()

{

var user = User.Identity.Name;

double xrp = QuantidadeTotalCriptomoeda("XRP", user);

ViewBag.QuantidadaTotalXrp = xrp.ToString("F6");

double saldoTotalXrp = CalcularSaldoAtual(xrp, "XRP");

ViewBag.QuantidadaEmRealXrp = saldoTotalXrp.ToString("F2");

double valorInvestidoXrp = ValorInvestido("XRP", user);

ViewBag.ValorInvestidoXrp = valorInvestidoXrp.ToString("F2");

ViewBag.LucroOuPerda = (saldoTotalXrp - valorInvestidoXrp).ToString("F2");

ViewBag.ValorXrp7Dias = Valores7Dias("XRP", xrp);

ViewBag.Ultimos7Dias = Ultimos7Dias();

double valorInvestidoBitcoin = ValorInvestido("Bitcoin", user);

ViewBag.ValorInvestidoBitcoin = valorInvestidoBitcoin.ToString("F2");

double valorInvestidoEthereum = ValorInvestido("Ethereum", user);

ViewBag.ValorInvestidoEthereum = valorInvestidoEthereum.ToString("F2");

double valorInvestidoBitcoinCash = ValorInvestido("Bitcoin Cash", user);

ViewBag.ValorInvestidoBitcoinCash = valorInvestidoBitcoinCash.ToString("F2");

double valorInvestidoPaxGold = ValorInvestido("PAX Gold", user);

ViewBag.ValorInvestidoPaxGold = valorInvestidoPaxGold.ToString("F2");

double valorInvestidoLitecoin = ValorInvestido("Litecoin", user);

ViewBag.ValorInvestidoLitecoin = valorInvestidoLitecoin.ToString("F2");

var Xrp = QuantidadeTotalCriptomoedaAdm("XRP");

ViewBag.Xrp = Xrp;

ViewBag.Ultimos7DiasAdm = Valores7DiasAdm("XRP", Xrp);

var XrpValorRSString = ValorTotalCriptomoedaAdm("XRP", Xrp).ToString("F2");

var XrpValorRS = Convert.ToDouble(XrpValorRSString);

ViewBag.XrpValorRS = XrpValorRS;

var XrpInvestidoString = ValorInvestidoAdm("XRP").ToString("F2");

var XrpInvestido = Convert.ToDouble(XrpInvestidoString);

ViewBag.XrpInvestido = XrpInvestido;

ViewBag.LucroOuPerdaAdm = (XrpValorRS - XrpInvestido).ToString("F2");

int primeiroInvestimento = DataPrimeiroInvestimento("XRP", user);

if (primeiroInvestimento <= 30)

{

ViewBag.UltimoMes = UltimosDias(primeiroInvestimento);

ViewBag.ValorMes = ValoresDias("XRP", xrp, primeiroInvestimento);

}

else

{

ViewBag.UltimoMes = Ultimos30Dias();

ViewBag.ValorMes = Valores30Dias("XRP", xrp);

}

int primeiroInvestimentoGeralAdm = DataPrimeiroInvestimentoAdm("XRP");

if (primeiroInvestimentoGeralAdm <= 30)

{

ViewBag.UltimoMesAdm = UltimosDias(primeiroInvestimentoGeralAdm);

ViewBag.ValorMesAdm = ValoresDias("XRP", Xrp, primeiroInvestimentoGeralAdm);

}

else

{

ViewBag.UltimoMesAdm = Ultimos30Dias();

ViewBag.ValorMesAdm = Valores30Dias("XRP", Xrp);

}

return View();

}

public ActionResult PaxGold()

{

var user = User.Identity.Name;

double paxGold = QuantidadeTotalCriptomoeda("PAX Gold", user);

ViewBag.QuantidadaTotalPaxGold = paxGold.ToString("F6");

double saldoTotalPaxGold = CalcularSaldoAtual(paxGold, "PAX Gold");

ViewBag.QuantidadaEmRealPaxGold = saldoTotalPaxGold.ToString("F2");

double valorInvestidoPaxGold = ValorInvestido("PAX Gold", user);

ViewBag.ValorInvestidoPaxGold = valorInvestidoPaxGold.ToString("F2");

ViewBag.LucroOuPerda = (saldoTotalPaxGold - valorInvestidoPaxGold).ToString("F2");

ViewBag.ValorPaxGold7Dias = Valores7Dias("PAX Gold", paxGold);

ViewBag.Ultimos7Dias = Ultimos7Dias();

double valorInvestidoBitcoin = ValorInvestido("Bitcoin", user);

ViewBag.ValorInvestidoBitcoin = valorInvestidoBitcoin.ToString("F2");

double valorInvestidoEthereum = ValorInvestido("Ethereum", user);

ViewBag.ValorInvestidoEthereum = valorInvestidoEthereum.ToString("F2");

double valorInvestidoBitcoinCash = ValorInvestido("Bitcoin Cash", user);

ViewBag.ValorInvestidoBitcoinCash = valorInvestidoBitcoinCash.ToString("F2");

double valorInvestidoXrp = ValorInvestido("XRP", user);

ViewBag.ValorInvestidoXrp = valorInvestidoXrp.ToString("F2");

double valorInvestidoLitecoin = ValorInvestido("Litecoin", user);

ViewBag.ValorInvestidoLitecoin = valorInvestidoLitecoin.ToString("F2");

var PaxGold = QuantidadeTotalCriptomoedaAdm("PAX Gold");

ViewBag.PaxGold = PaxGold;

ViewBag.Ultimos7DiasAdm = Valores7DiasAdm("PAX Gold", PaxGold);

var PaxGoldValorRSString = ValorTotalCriptomoedaAdm("PAX Gold", PaxGold).ToString("F2");

double PaxGoldValorRS = Convert.ToDouble(PaxGoldValorRSString);

ViewBag.PaxGoldValorRS = PaxGoldValorRS;

var PaxGoldInvestidoString = ValorInvestidoAdm("PAX Gold").ToString("F2");

double PaxGoldInvestido = Convert.ToDouble(PaxGoldInvestidoString);

ViewBag.PaxGoldInvestido = PaxGoldInvestido;

ViewBag.LucroOuPerdaAdm = (PaxGoldValorRS - PaxGoldInvestido).ToString("F2");

int primeiroInvestimento = DataPrimeiroInvestimento("PAX Gold", user);

if (primeiroInvestimento <= 30)

{

ViewBag.UltimoMes = UltimosDias(primeiroInvestimento);

ViewBag.ValorMes = ValoresDias("PAX Gold", paxGold, primeiroInvestimento);

}

else

{

ViewBag.UltimoMes = Ultimos30Dias();

ViewBag.ValorMes = Valores30Dias("PAX Gold", paxGold);

}

int primeiroInvestimentoGeralAdm = DataPrimeiroInvestimentoAdm("PAX Gold");

if (primeiroInvestimentoGeralAdm <= 30)

{

ViewBag.UltimoMesAdm = UltimosDias(primeiroInvestimentoGeralAdm);

ViewBag.ValorMesAdm = ValoresDias("PAX Gold", PaxGold, primeiroInvestimentoGeralAdm);

}

else

{

ViewBag.UltimoMesAdm = Ultimos30Dias();

ViewBag.ValorMesAdm = Valores30Dias("PAX Gold", PaxGold);

}

return View();

}

public ActionResult Litecoin()

{

var user = User.Identity.Name;

double litecoin = QuantidadeTotalCriptomoeda("Litecoin", user);

ViewBag.QuantidadaTotalLitecoin = litecoin.ToString("F6");

double saldoTotalLitecoin = CalcularSaldoAtual(litecoin, "Litecoin");

ViewBag.QuantidadaEmRealLitecoin = saldoTotalLitecoin.ToString("F2");

double valorInvestidoLitecoin = ValorInvestido("Litecoin", user);

ViewBag.ValorInvestidoLitecoin = valorInvestidoLitecoin.ToString("F2");

ViewBag.LucroOuPerda = (saldoTotalLitecoin - valorInvestidoLitecoin).ToString("F2");

ViewBag.ValorLitecoin7Dias = Valores7Dias("Litecoin", litecoin);

ViewBag.Ultimos7Dias = Ultimos7Dias();

double valorInvestidoBitcoin = ValorInvestido("Bitcoin", user);

ViewBag.ValorInvestidoBitcoin = valorInvestidoBitcoin.ToString("F2");

double valorInvestidoEthereum = ValorInvestido("Ethereum", user);

ViewBag.ValorInvestidoEthereum = valorInvestidoEthereum.ToString("F2");

double valorInvestidoBitcoinCash = ValorInvestido("Bitcoin Cash", user);

ViewBag.ValorInvestidoBitcoinCash = valorInvestidoBitcoinCash.ToString("F2");

double valorInvestidoXrp = ValorInvestido("XRP", user);

ViewBag.ValorInvestidoXrp = valorInvestidoXrp.ToString("F2");

double valorInvestidoPaxGold = ValorInvestido("PAX Gold", user);

ViewBag.ValorInvestidoPaxGold = valorInvestidoPaxGold.ToString("F2");

var Litecoin = QuantidadeTotalCriptomoedaAdm("Litecoin");

ViewBag.Litecoin = Litecoin;

ViewBag.Ultimos7DiasAdm = Valores7DiasAdm("Litecoin", Litecoin);

var LitecoinValorRSString = ValorTotalCriptomoedaAdm("Litecoin", Litecoin).ToString("F2");

var LitecoinValorRS = Convert.ToDouble(LitecoinValorRSString);

ViewBag.LitecoinValorRS = LitecoinValorRS;

var LitecoinInvestidoString = ValorInvestidoAdm("Litecoin").ToString("F2");

var LitecoinInvestido = Convert.ToDouble(LitecoinInvestidoString);

ViewBag.LitecoinInvestido = LitecoinInvestido;

ViewBag.LucroOuPerdaAdm = (LitecoinValorRS - LitecoinInvestido).ToString("F2");

int primeiroInvestimento = DataPrimeiroInvestimento("Litecoin", user);

if (primeiroInvestimento <= 30)

{

ViewBag.UltimoMes = UltimosDias(primeiroInvestimento);

ViewBag.ValorMes = ValoresDias("Litecoin", litecoin, primeiroInvestimento);

}

else

{

ViewBag.UltimoMes = Ultimos30Dias();

ViewBag.ValorMes = Valores30Dias("Litecoin", litecoin);

}

int primeiroInvestimentoGeralAdm = DataPrimeiroInvestimentoAdm("Litecoin");

if (primeiroInvestimentoGeralAdm <= 30)

{

ViewBag.UltimoMesAdm = UltimosDias(primeiroInvestimentoGeralAdm);

ViewBag.ValorMesAdm = ValoresDias("Litecoin", Litecoin, primeiroInvestimentoGeralAdm);

}

else

{

ViewBag.UltimoMesAdm = Ultimos30Dias();

ViewBag.ValorMesAdm = Valores30Dias("Litecoin", Litecoin);

}

return View();

}

public List<int> Ultimos7Dias()

{

var diasList = new List<int>();

DateTime diasSete = DateTime.Today;

for (int i = 6; i >= 0; i--)

{

diasSete = DateTime.Today;

diasSete = diasSete.AddDays(-i);

diasList.Add(diasSete.Day);

}

return diasList;

}

public List<double> Valores7Dias(string nome, double quantidadeTotalCriptomoeda)

{

var valorList = new List<double>();

for (int i = 6; i >= 0; i--)

{

DateTime diasSete = DateTime.Today;

diasSete = diasSete.AddDays(-i);

DateTime data = diasSete;

var valor = (from coin in \_context.Criptomoeda

join criptohoje in \_context.CriptomoedaHoje

on coin.Id equals criptohoje.CriptomoedaId

where coin.Nome == nome && criptohoje.Data.Date.Equals(data.Date)

select criptohoje.Valor).Single();

var totalDia = (quantidadeTotalCriptomoeda \* valor).ToString("F2");

valorList.Add(Convert.ToDouble(totalDia));

}

return valorList;

}

public List<double> Valores7DiasAdm(string nome, double quantidadeTotalCriptomoeda)

{

var valorList = new List<double>();

for (int i = 6; i >= 0; i--)

{

DateTime diasSete = DateTime.Today;

diasSete = diasSete.AddDays(-i);

DateTime data = diasSete;

var valor = (from coin in \_context.Criptomoeda

join criptohoje in \_context.CriptomoedaHoje

on coin.Id equals criptohoje.CriptomoedaId

where coin.Nome == nome && criptohoje.Data.Date.Equals(data.Date)

select criptohoje.Valor).Sum();

var totalDia = (quantidadeTotalCriptomoeda \* valor).ToString("F2");

valorList.Add(Convert.ToDouble(totalDia));

}

return valorList;

}

public double QuantidadeTotalCriptomoedaAdm(string criptomoeda)

{

var cripto = (from transacoes in \_context.Transacao

join conta in \_context.ContaCliente

on transacoes.ContaClienteId equals conta.Id

join criptomoedahoje in \_context.CriptomoedaHoje

on transacoes.CriptomoedaHojeId equals criptomoedahoje.Id

join criptomoedas in \_context.Criptomoeda

on criptomoedahoje.CriptomoedaId equals criptomoedas.Id

join criptosaldo in \_context.CriptoSaldo

on transacoes.CriptoSaldoId equals criptosaldo.Id

where criptomoedas.Nome == criptomoeda

select criptosaldo.Quantidade).Sum();

return cripto;

}

public double ValorTotalCriptomoedaAdm(string criptomoeda, double quantidadeCripto)

{

var cripto = (from criptomoedahoje in \_context.CriptomoedaHoje

join criptomoedas in \_context.Criptomoeda

on criptomoedahoje.CriptomoedaId equals criptomoedas.Id

where criptomoedas.Nome == criptomoeda

orderby criptomoedahoje.Id descending

select criptomoedahoje.Valor).FirstOrDefault();

return cripto \* quantidadeCripto;

}

//

public int DataPrimeiroInvestimento(string criptomoeda, string user)

{

var primeiraData = (from transacao in \_context.Transacao

join contacliente in \_context.ContaCliente

on transacao.ContaClienteId equals contacliente.Id

join usuario in \_context.ApplicationUser

on contacliente.ApplicationUserID equals usuario.Id

where usuario.UserName == user

join criptohoje in \_context.CriptomoedaHoje

on transacao.CriptomoedaHojeId equals criptohoje.Id

join cripto in \_context.Criptomoeda

on criptohoje.CriptomoedaId equals cripto.Id

where cripto.Nome == criptomoeda

orderby transacao.Data ascending

select transacao.Data).FirstOrDefault();

DateTime hoje = DateTime.Today;

return hoje.Subtract(primeiraData).Days;

}

public int DataPrimeiroInvestimentoAdm(string criptomoeda)

{

var primeiraData = (from transacao in \_context.Transacao

join contacliente in \_context.ContaCliente

on transacao.ContaClienteId equals contacliente.Id

join usuario in \_context.ApplicationUser

on contacliente.ApplicationUserID equals usuario.Id

join criptohoje in \_context.CriptomoedaHoje

on transacao.CriptomoedaHojeId equals criptohoje.Id

join cripto in \_context.Criptomoeda

on criptohoje.CriptomoedaId equals cripto.Id

where cripto.Nome == criptomoeda

orderby transacao.Data ascending

select transacao.Data).FirstOrDefault();

DateTime hoje = DateTime.Today;

return hoje.Subtract(primeiraData).Days;

}

public List<int> UltimosDias(int dias) //retorno do DataPrimeiroInvestimento() //Mesmo ara ADM

{

var diasList = new List<int>();

DateTime diaAtual;

for (int i = dias; i >= 0; i--)

{

diaAtual = DateTime.Today;

diaAtual = diaAtual.AddDays(-i);

diasList.Add(diaAtual.Day);

}

return diasList;

}

public List<double> ValoresDias(string nome, double quantidadeTotalCriptomoeda, int dias) //dias = retorno do DataPrimeiroInvestimento()

{

var valorList = new List<double>();

for (int i = dias; i >= 0; i--)

{

DateTime diasTotal = DateTime.Today;

diasTotal = diasTotal.AddDays(-i);

DateTime data = diasTotal;

var valor = (from coin in \_context.Criptomoeda

join criptohoje in \_context.CriptomoedaHoje

on coin.Id equals criptohoje.CriptomoedaId

where coin.Nome == nome && criptohoje.Data.Date.Equals(data.Date)

select criptohoje.Valor).Single();

var totalDia = (quantidadeTotalCriptomoeda \* valor).ToString("F2");

valorList.Add(Convert.ToDouble(totalDia));

}

return valorList;

}

public List<int> Ultimos30Dias()

{

var diasList = new List<int>();

DateTime diaAtual;

for (int i = 31; i >= 0; i--)

{

diaAtual = DateTime.Today;

diaAtual = diaAtual.AddDays(-i);

diasList.Add(diaAtual.Day);

}

return diasList;

}

public List<double> Valores30Dias(string nome, double quantidadeTotalCriptomoeda)

{

var valorList = new List<double>();

for (int i = 31; i >= 0; i--)

{

DateTime diasTotal = DateTime.Today;

diasTotal = diasTotal.AddDays(-i);

DateTime data = diasTotal;

var valor = (from coin in \_context.Criptomoeda

join criptohoje in \_context.CriptomoedaHoje

on coin.Id equals criptohoje.CriptomoedaId

where coin.Nome == nome && criptohoje.Data.Date.Equals(data.Date)

select criptohoje.Valor).Single();

var totalDia = (quantidadeTotalCriptomoeda \* valor).ToString("F2");

valorList.Add(Convert.ToDouble(totalDia));

}

return valorList;

}

private bool TransacaoExists(int id)

{

return \_context.Transacao.Any(e => e.Id == id);

}

}

}

**Controller – RelatoriosController**

using System;

using System.Collections.Generic;

using System.Linq;

using Microsoft.AspNetCore.Mvc;

using BlockTechMVC.Data;

using Microsoft.AspNetCore.Authorization;

namespace BlockTechMVC.Controllers

{

[Authorize]

public class RelatoriosController : Controller

{

private readonly ApplicationDbContext \_context;

public RelatoriosController(ApplicationDbContext context)

{

\_context = context;

}

// GET: Relatorios

public ActionResult Index()

{

ViewBag.Bitcoin = CriptomoedaHoje("Bitcoin").ToString("F4");

ViewBag.Ethereum = CriptomoedaHoje("Ethereum").ToString("F4");

ViewBag.BitcoinCash = CriptomoedaHoje("Bitcoin Cash").ToString("F4");

ViewBag.XRP = CriptomoedaHoje("XRP").ToString("F4");

ViewBag.PaxGold = CriptomoedaHoje("PAX Gold").ToString("F4");

ViewBag.Litecoin = CriptomoedaHoje("Litecoin").ToString("F4");

return View();

}

public double CriptomoedaHoje(string nome)

{

var criptomoeda = (from coin in \_context.Criptomoeda

join criptohoje in \_context.CriptomoedaHoje

on coin.Id equals criptohoje.CriptomoedaId

where coin.Nome == nome && criptohoje.Data == DateTime.Today

select criptohoje.Valor).Single();

return criptomoeda;

}

public ActionResult Bitcoin()

{

ViewBag.Dias = Ultimos7Dias();

ViewBag.Valores = Valores7Dias("Bitcoin");

return View();

}

public ActionResult Bitcoin30()

{

ViewBag.Dias = Ultimos30Dias();

ViewBag.Valores = Valores30Dias("Bitcoin");

return View();

}

public ActionResult Ethereum()

{

ViewBag.Dias = Ultimos7Dias();

ViewBag.Valores = Valores7Dias("Ethereum");

return View();

}

public ActionResult Ethereum30()

{

ViewBag.Dias = Ultimos30Dias();

ViewBag.Valores = Valores30Dias("Ethereum");

return View();

}

public ActionResult BitcoinCash()

{

ViewBag.Dias = Ultimos7Dias();

ViewBag.Valores = Valores7Dias("Bitcoin Cash");

return View();

}

public ActionResult BitcoinCash30()

{

ViewBag.Dias = Ultimos30Dias();

ViewBag.Valores = Valores30Dias("Bitcoin Cash");

return View();

}

public ActionResult Xrp()

{

ViewBag.Dias = Ultimos7Dias();

ViewBag.Valores = Valores7Dias("XRP");

return View();

}

public ActionResult Xrp30()

{

ViewBag.Dias = Ultimos30Dias();

ViewBag.Valores = Valores30Dias("XRP");

return View();

}

public ActionResult PaxGold()

{

ViewBag.Dias = Ultimos7Dias();

ViewBag.Valores = Valores7Dias("PAX Gold");

return View();

}

public ActionResult PaxGold30()

{

ViewBag.Dias = Ultimos30Dias();

ViewBag.Valores = Valores30Dias("PAX Gold");

return View();

}

public ActionResult Litecoin()

{

ViewBag.Dias = Ultimos7Dias();

ViewBag.Valores = Valores7Dias("Litecoin");

return View();

}

public ActionResult Litecoin30()

{

ViewBag.Dias = Ultimos30Dias();

ViewBag.Valores = Valores30Dias("Litecoin");

return View();

}

public List<int> Ultimos7Dias() {

var diasList = new List<int>();

DateTime diasSete = DateTime.Today;

for (int i = 7; i >= 0; i--)

{

diasSete = DateTime.Today;

diasSete = diasSete.AddDays(-i);

diasList.Add(diasSete.Day);

}

return diasList;

}

public List<int> Ultimos30Dias() {

var diasList = new List<int>();

DateTime diasTrinta = DateTime.Today;

for (int i = 30; i >= 0; i--)

{

diasTrinta = DateTime.Today;

diasTrinta = diasTrinta.AddDays(-i);

diasList.Add(diasTrinta.Day);

}

return diasList;

}

public List<double> Valores7Dias(string nome)

{

var valorList = new List<double>();

for (int i = 7; i >= 0; i--)

{

DateTime diasSete = DateTime.Today;

diasSete = diasSete.AddDays(-i);

DateTime data = diasSete;

var valor = (from coin in \_context.Criptomoeda

join criptohoje in \_context.CriptomoedaHoje

on coin.Id equals criptohoje.CriptomoedaId

where coin.Nome == nome && criptohoje.Data.Date.Equals(data.Date)

select criptohoje.Valor).Single();

valorList.Add(valor);

}

return valorList;

}

public List<double> Valores30Dias(string nome)

{

var valorList = new List<double>();

for (int i = 30; i >= 0; i--)

{

DateTime diasSete = DateTime.Today;

diasSete = diasSete.AddDays(-i);

DateTime data = diasSete;

var valor = (from coin in \_context.Criptomoeda

join criptohoje in \_context.CriptomoedaHoje

on coin.Id equals criptohoje.CriptomoedaId

where coin.Nome == nome && criptohoje.Data.Date.Equals(data.Date)

select criptohoje.Valor).Single();

valorList.Add(valor);

}

return valorList;

}

public IActionResult Semanal()

{

ViewBag.Dias = Ultimos7Dias();

ViewBag.Bitcoin = Porcentagem("Bitcoin");

ViewBag.Ethereum = Porcentagem("Ethereum");

ViewBag.BitcoinCash = Porcentagem("Bitcoin Cash");

ViewBag.XRP = Porcentagem("XRP");

ViewBag.PaxGold = Porcentagem("PAX Gold");

ViewBag.Litecoin = Porcentagem("Litecoin");

return View();

}

public IActionResult Mensal()

{

ViewBag.Dias = Ultimos30Dias();

ViewBag.Bitcoin = PorcentagemTrinta("Bitcoin");

ViewBag.Ethereum = PorcentagemTrinta("Ethereum");

ViewBag.BitcoinCash = PorcentagemTrinta("Bitcoin Cash");

ViewBag.XRP = PorcentagemTrinta("XRP");

ViewBag.PaxGold = PorcentagemTrinta("PAX Gold");

ViewBag.Litecoin = PorcentagemTrinta("Litecoin");

return View();

}

public List<double> Porcentagem(string nome)

{

var date = DateTime.Today;

date = date.AddDays(-8);

var valor = (from coin in \_context.Criptomoeda

join criptohoje in \_context.CriptomoedaHoje

on coin.Id equals criptohoje.CriptomoedaId

where coin.Nome == nome && criptohoje.Data.Date.Equals(date.Date)

select criptohoje.Valor).Single();

var calculo = 100.0;

var porcentagem = new List<double>();

for (int i = 7; i >= 0; i--)

{

DateTime dia = DateTime.Today;

dia = dia.AddDays(-i);

DateTime data = dia;

var valorDia = (from coin in \_context.Criptomoeda

join criptohoje in \_context.CriptomoedaHoje

on coin.Id equals criptohoje.CriptomoedaId

where coin.Nome == nome && criptohoje.Data.Date.Equals(data.Date)

select criptohoje.Valor).Single();

var regra3 = ((valorDia \* calculo) / valor);

var resultado = regra3 - 100;

valor = valorDia;

var duasCasas = resultado.ToString("F2");

var duasCasasDouble = Convert.ToDouble(duasCasas);

porcentagem.Add(duasCasasDouble);

}

return porcentagem;

}

public List<double> PorcentagemTrinta(string nome)

{

var date = DateTime.Today;

date = date.AddDays(-31);

var valor = (from coin in \_context.Criptomoeda

join criptohoje in \_context.CriptomoedaHoje

on coin.Id equals criptohoje.CriptomoedaId

where coin.Nome == nome && criptohoje.Data.Date.Equals(date.Date)

select criptohoje.Valor).Single();

var calculo = 100.0;

var porcentagem = new List<double>();

for (int i = 30; i >= 0; i--)

{

DateTime dia = DateTime.Today;

dia = dia.AddDays(-i);

DateTime data = dia;

var valorDia = (from coin in \_context.Criptomoeda

join criptohoje in \_context.CriptomoedaHoje

on coin.Id equals criptohoje.CriptomoedaId

where coin.Nome == nome && criptohoje.Data.Date.Equals(data.Date)

select criptohoje.Valor).Single();

var regra3 = ((valorDia \* calculo) / valor);

var resultado = regra3 - 100;

valor = valorDia;

var duasCasas = resultado.ToString("F2");

var duasCasasDouble = Convert.ToDouble(duasCasas);

porcentagem.Add(duasCasasDouble);

}

return porcentagem;

}

}

}

**Controller – TransacoesController**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using Microsoft.AspNetCore.Mvc;

using Microsoft.AspNetCore.Mvc.Rendering;

using Microsoft.EntityFrameworkCore;

using BlockTechMVC.Data;

using BlockTechMVC.Models;

using Microsoft.AspNetCore.Authorization;

using System.Diagnostics;

namespace BlockTechMVC.Controllers

{

[Authorize]

public class TransacoesController : Controller

{

private readonly ApplicationDbContext \_context;

public TransacoesController(ApplicationDbContext context)

{

\_context = context;

}

// GET: Transacoes

public async Task<IActionResult> Index(int? Busca, string searchString, string sortOrder)

{

var user = User.Identity.Name;

ViewBag.Data = sortOrder == "Data" ? "Data\_desc" : "Data";

ViewBag.Quantidade = sortOrder == "Quantidade" ? "Quantidade\_desc" : "Quantidade";

ViewBag.Criptomoeda = sortOrder == "Criptomoeda" ? "Criptomoeda\_desc" : "Criptomoeda";

ViewBag.Valor = sortOrder == "Valor" ? "Valor\_desc" : "Valor";

if (user == "Administrador")

{

ViewBag.Nome = sortOrder == "Nome" ? "Nome\_desc" : "Nome";

List<SelectListItem> itens = new List<SelectListItem>();

SelectListItem item1 = new SelectListItem() { Text = "Cliente", Value = "1", Selected = true };

SelectListItem item2 = new SelectListItem() { Text = "Criptomoeda", Value = "2", Selected = false };

itens.Add(item1);

itens.Add(item2);

ViewBag.Busca = itens;

var applicationDbContext = \_context.Transacao

.Include(t => t.ContaCliente)

.Include(t => t.CriptoSaldo)

.Include(t => t.CriptomoedaHoje)

.Include(t => t.ContaCliente.ApplicationUser)

.Include(t => t.CriptomoedaHoje.Criptomoeda)

.Include(t => t.Saldo)

.OrderBy(t => t.Data);

if (sortOrder != null)

{

var orderName = applicationDbContext.OrderBy(t => t.ContaCliente.ApplicationUser.Nome);

switch (sortOrder)

{

case "Nome\_desc":

orderName = orderName.OrderByDescending(s => s.ContaCliente.ApplicationUser.Nome);

break;

case "Nome":

orderName = applicationDbContext.OrderBy(s => s.ContaCliente.ApplicationUser.Nome);

break;

case "Data":

orderName = applicationDbContext.OrderBy(s => s.Data);

break;

case "Data\_desc":

orderName = applicationDbContext.OrderByDescending(s => s.Data);

break;

case "Quantidade":

orderName = applicationDbContext.OrderBy(s => s.CriptoSaldo.Quantidade);

break;

case "Quantidade\_desc":

orderName = applicationDbContext.OrderByDescending(s => s.CriptoSaldo.Quantidade);

break;

case "Criptomoeda":

orderName = applicationDbContext.OrderBy(s => s.CriptomoedaHoje.Criptomoeda.Nome);

break;

case "Criptomoeda\_desc":

orderName = applicationDbContext.OrderByDescending(s => s.CriptomoedaHoje.Criptomoeda.Nome);

break;

case "Valor":

orderName = applicationDbContext.OrderBy(s => s.Valor);

break;

case "Valor\_desc":

orderName = applicationDbContext.OrderByDescending(s => s.Valor);

break;

default:

orderName = applicationDbContext.OrderBy(s => s.ContaCliente.ApplicationUser.Nome);

break;

};

return View(orderName.ToList());

}

if (Busca != null)

{

itens.Where(i => i.Value == Busca.ToString()).First().Selected = true;

}

if (Busca == 1)

{

if (!String.IsNullOrEmpty(searchString))

{

var usuarioSelecionado = \_context.Transacao

.Include(t => t.ContaCliente)

.Include(t => t.CriptoSaldo)

.Include(t => t.CriptomoedaHoje)

.Include(t => t.ContaCliente.ApplicationUser)

.Include(t => t.CriptomoedaHoje.Criptomoeda)

.Include(t => t.Saldo)

.Where(t => t.ContaCliente.ApplicationUser.Nome.Contains(searchString));

return View(usuarioSelecionado.ToList());

}

}

if (Busca == 2)

{

if (!String.IsNullOrEmpty(searchString))

{

var usuarioSelecionado = \_context.Transacao

.Include(t => t.ContaCliente)

.Include(t => t.CriptoSaldo)

.Include(t => t.CriptomoedaHoje)

.Include(t => t.ContaCliente.ApplicationUser)

.Include(t => t.CriptomoedaHoje.Criptomoeda)

.Include(t => t.Saldo)

.Where(t => t.CriptomoedaHoje.Criptomoeda.Nome.Contains(searchString));

return View(usuarioSelecionado.ToList());

}

}

return View(await applicationDbContext.ToListAsync());

}

else

{

var usuario = \_context.Transacao

.Where(t => t.ContaCliente.ApplicationUser.UserName == user)

.Include(t => t.ContaCliente)

.Where(t => t.ContaCliente.ApplicationUser.UserName == user)

.Include(t => t.CriptoSaldo)

.Where(t => t.CriptoSaldo.ContaCliente.ApplicationUser.UserName == user)

.Include(t => t.CriptomoedaHoje)

.Include(t => t.ContaCliente.ApplicationUser)

.Where(t => t.ContaCliente.ApplicationUser.UserName == user)

.Include(t => t.CriptomoedaHoje.Criptomoeda)

.Include(t => t.Saldo)

.Where(t => t.Saldo.ContaCliente.ApplicationUser.UserName == user);

if (sortOrder != null)

{

var orderName = usuario.OrderBy(t => t.CriptomoedaHoje.Criptomoeda.Nome);

switch (sortOrder)

{

case "Data":

orderName = usuario.OrderBy(s => s.Data);

break;

case "Data\_desc":

orderName = usuario.OrderByDescending(s => s.Data);

break;

case "Quantidade":

orderName = usuario.OrderBy(s => s.CriptoSaldo.Quantidade);

break;

case "Quantidade\_desc":

orderName = usuario.OrderByDescending(s => s.CriptoSaldo.Quantidade);

break;

case "Criptomoeda":

orderName = usuario.OrderBy(s => s.CriptomoedaHoje.Criptomoeda.Nome);

break;

case "Criptomoeda\_desc":

orderName = usuario.OrderByDescending(s => s.CriptomoedaHoje.Criptomoeda.Nome);

break;

case "Valor":

orderName = usuario.OrderBy(s => s.Valor);

break;

case "Valor\_desc":

orderName = usuario.OrderByDescending(s => s.Valor);

break;

default:

orderName = usuario.OrderBy(s => s.CriptomoedaHoje.Criptomoeda.Nome);

break;

};

return View(orderName.ToList());

}

if (!String.IsNullOrEmpty(searchString))

{

var criptoSelecionada = \_context.Transacao

.Include(t => t.ContaCliente)

.Include(t => t.CriptoSaldo)

.Include(t => t.CriptomoedaHoje)

.Include(t => t.ContaCliente.ApplicationUser)

.Include(t => t.CriptomoedaHoje.Criptomoeda)

.Include(t => t.Saldo)

.Where(t => t.CriptomoedaHoje.Criptomoeda.Nome.Contains(searchString) && t.Saldo.ContaCliente.ApplicationUser.UserName == user);

return View(criptoSelecionada.ToList());

}

return View(await usuario.ToListAsync());

}

}

private bool TransacaoExists(int id)

{

return \_context.Transacao.Any(e => e.Id == id);

}

public IActionResult Error(string message)

{

var viewModel = new ErrorViewModel

{

Message = message,

RequestId = Activity.Current?.Id ?? HttpContext.TraceIdentifier

};

return View(viewModel);

}

}

}