Sistema de Moeda Estudantil: Reconhecimento e Recompensa

O sistema de moeda estudantil é uma iniciativa inovadora que visa premiar alunos com moedas virtuais por bom comportamento e participação ativa em atividades acadêmicas.

Essas moedas podem ser trocadas por vantagens e benefícios exclusivos oferecidos por empresas parceiras, incentivando o engajamento e o desempenho dos estudantes.

Integrantes: Carlos Roberto Alves de Almeida Júnior, Davi Rosa Gentil, João Vitor Pedersoli Rajão, Otávio Salomão Ferreira Silva.



Visão Geral do Sistema

1 Reconhecimento e Recompensa

Professores premiam alunos com moedas por mérito acadêmico e bom comportamento. 2 Vantagens e Benefícios

Alunos trocam moedas por descontos e produtos de empresas parceiras.

3 Fomento à Participação

Incentiva a participação ativa e o bom desempenho dos estudantes.



Funcionalidades Principais



Cadastro de Usuários

Alunos, professores e empresas se cadastram facilmente.



Distribuição de Moedas

Professores enviam moedas com mensagens personalizadas.



Consulta de Saldo

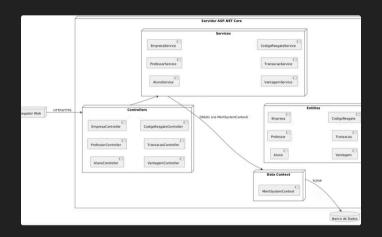
Acompanhamento de saldo e extrato de transações.

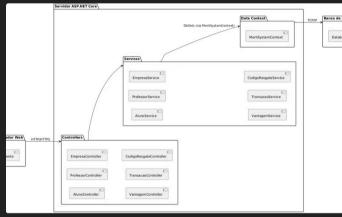


Troca de Moedas

Alunos trocam moedas por vantagens exclusivas.

Arquitetura do Sistema







MVC

Padrão arquitetural que separa a aplicação em camadas, promovendo organização e modularidade.

Ferramentas

.NET Core para backend, Docker para banco de dados e Swagger UI para documentação da API.

Entity Framework

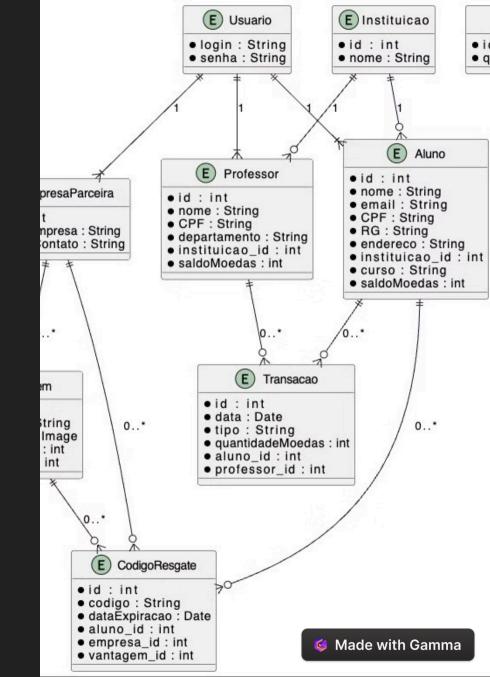
Simplifica o acesso e gerenciamento do banco de dados, otimizando a interação entre código e dados.

Diagrama ER

O diagrama demonstra as entidades e seus relacionamentos, com chaves primárias e estrangeiras.

Entidades: Aluno, Professor, Empresa, Vantagem e Transação.

Gerenciado pelo Entity Framework para acesso eficiente aos dados.



```
Fast database acciss FenitySt:
     Task: Eufebase Ugsies:
      Rale:: Palbars =SS= (Database EastifyBile:
       Dertlerf.ts -ht:
       Typile: suppel: factl fon Prsemmert, of
            KeryingUast: Rupell = Totbe";
            Sertiing:
             Mussicf:: cdade: fappefors=:of7FTL,con/805, for to, RBK)
                  lalt ) = crfact = lotp,
             Net: locdf:
             Statacive: fame phate: Coriel Keigper) (
                 stadcNtore-Puper: =Fougtef: Faad Francesfork/bc/Onbelinger
                 Camt:
         Stew(ine habiler - focole, fart)
               therinoker wettings: = ojer facter -- perie cories data significa
               Naste/firnibettor intation: ff) with feething for the select und by
            Meladlast: Alpe: Retielizafor Fesfacle Cotiget The IW)
               detac tesss:
        Conneblance: Feassins(Anp:
            Stoles: Idaper writamel() - Captabags, fart recipile with the manual
              Shoplast: pligg!: )
        foypudt: esterts focesab()

Deschenanst: -dationus yyaten top:
```

Acesso ao Banco de Dados

MeritSystemContext

Centraliza o acesso ao banco de dados.

CRUD

Operações de criação, leitura, atualização e exclusão.

LINQ e Entity Framework

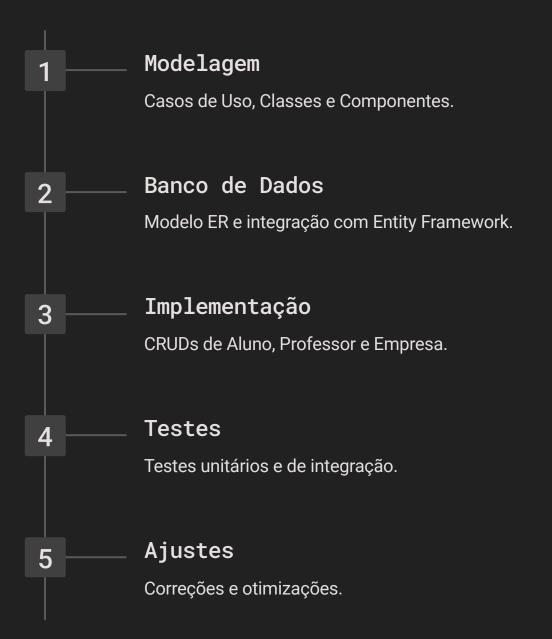
Consultas e transações eficientes.



Tecnologias Utilizadas

Backend	.NET Core, C#
Banco de Dados	Docker, SQL Server/MySQL, Entity Framework
Ferramentas	Swagger UI, Docker Desktop, Git
Testes	Swagger UI

Processo de Desenvolvimento



CorbertonRsuppion:
Tecderler vardesttUrarsjlerrang = = -6 = 2 4.7.27.1:1
Cotberter:Stldugn.ce
Techdets.CounLftt.]ACT51766026U55975880270240, 00, 12, 00, 10:3 201

Como Rodar o Projeto

Requisitos

1

2

3

4

Docker Desktop e .NET Core SDK.

Configuração

Clonar, subir o banco, aplicar migrações.

Execução

Rodar com 'dotnet run'.

Acesso

Via Swagger UI.

Conclusão

1 Requisitos Atendidos

Sistema funcional e eficiente.

Reconhecimento e Vantagens

Implementadas com sucesso.

3 Próximos Passos

Integração com empresas e melhorias na interface.

