SISTEMA DE CONTROLE DE CONTAS DE ENERGIA

Documentação do Projeto

Aluno: _____

Professor: _____

Disciplina: Programação Orientada a Objetos

Data: ____ / ____ / _____

1. INTRODUÇÃO

Este trabalho apresenta o desenvolvimento de um sistema para controle de contas de energia elétrica, criado em C# utilizando Windows Forms, Programação Orientada a Objetos e persistência de dados em JSON e XML. O sistema foi motivado pelo cenário de crise hídrica no Brasil e pela necessidade crescente de consumidores acompanharem e reduzirem seus gastos energéticos.

2. REQUISITOS DO SISTEMA

2.1 Requisitos Funcionais

• RF01 – Cadastrar consumidores (CPF ou CNPJ).
• RF02 – Cadastrar contas de energia (residencial ou comercial).
• RF03 – Registrar leituras atual e anterior da instalação elétrica.
• RF04 – Calcular automaticamente o consumo mensal (kW/h).
• RF05 – Calcular valor da conta sem impostos.
• RF06 – Calcular valor total com impostos.
• RF07 – Consultar informações de um consumidor por ID (GUID).
• RF08 – Exibir detalhes completos de cada conta cadastrada.
• RF09 – Salvar automaticamente dados em arquivo JSON.
• RF10 – Permitir exportar manualmente os dados para XML.

2.2 Requisitos Não Funcionais

• RNF01 – Interface gráfica desenvolvida em Windows Forms.
• RNF02 – Persistência em arquivos JSON e XML.
• RNF03 – Código implementado seguindo princípios de POO.
• RNF04 – Aplicação de padrões de projeto (Design Patterns).
• RNF05 – Compatibilidade com .NET 7.

3. DESIGN PATTERNS UTILIZADOS

3.1 Factory Method

Este padrão foi utilizado para criar contas de energia de tipos diferentes (Residencial ou Comercial). Dependendo da escolha do usuário na interface, o sistema instancia automaticamente a classe correta. Isso reduz acoplamento e facilita futuras expansões.

Exemplo no código:

```
Account acc = rbResidencial.Checked ?
    new ResidentialAccount() :
    new CommercialAccount();
```

3.2 Repository Pattern

O Repository Pattern centraliza toda a lógica de persistência de dados (salvar, ler, manipular). Em vez de espalhar lógica de JSON/XML pelo sistema, todo o acesso aos arquivos é feito exclusivamente pelo objeto Repository, separando a lógica da interface gráfica e facilitando manutenção.

Exemplo de métodos no repositório:

- AddConsumer()
- AddAccount()
- SaveJson()
- SaveXml()
- Load()

4. MODELAGEM (UML)

O sistema foi modelado utilizando classes que representam consumidores e contas. A classe Account é abstrata e possui duas especializações: ResidentialAccount e CommercialAccount. O Repository gerencia toda a persistência de dados.

Diagrama conceitual (texto):

• Consumer
  – Id: GUID
  – Name: string
  – Document: string (CPF/CNPJ)
  – AccountIds: List

• Account (abstract)
  – Id
  – RegistrationNumber
  – PreviousReading
  – CurrentReading
  – Methods: ValueWithoutTax(), TotalValue(), Consumption

• ResidentialAccount : Account
• CommercialAccount : Account
• Repository

5. PERSISTÊNCIA DOS DADOS

O sistema armazena automaticamente os dados em JSON sempre que uma nova conta ou consumidor é criado. O usuário também pode exportar todos os dados para XML através de um botão na interface.

Arquivos gerados pelo sistema:

• data.json – salvo automaticamente.
• data.xml – salvo quando o usuário seleciona 'Salvar XML'.

Local dos arquivos:
bin/Debug/net7.0-windows/

6. CONCLUSÃO

O sistema de Controle de Contas de Energia cumpre todos os requisitos propostos: utiliza programação orientada a objetos, implementa padrões de projeto, possui interface gráfica, realiza cálculos energéticos corretamente e armazena dados em arquivos JSON e XML. O projeto está apto para apresentação e entrega acadêmica.