|  |  |
| --- | --- |
| **Análise de requisitos** | **Funcionais:** O sistema appComercio oferece as seguintes funcionalidades principais:   * **Autenticação de Usuários**: Permite que usuários com diferentes perfis (Administrador, Vendedor, Estoquista) acessem o sistema por meio de uma tela de login. * **Gerenciamento de Usuários (CRUD)**: Possibilita o cadastro, edição, exclusão e consulta de usuários, com controle de perfis de acesso. A senha do administrador para cadastro é fixada em "1234". * **Cadastro de Produtos/Serviços (CRUD)**: Permite o registro de novos produtos ou serviços, incluindo categorias pré-definidas (Fermentada, Destilada, Licor, Não Alcoólica, Outras) e fornecedores (Distribuidora A, Atacadista B, Produtor C, Outro Fornecedor). Oferece funcionalidades para atualização e exclusão de itens. * **Visualização e Edição de Dados de Produtos/Serviços**: Permite a consulta detalhada, edição e salvamento de informações de produtos e serviços, incluindo preço e quantidade. * **Registro de Vendas**: Funcionalidade para registrar vendas, associando produtos e suas quantidades, e enviando esses dados para a API de backend. * **Registro de Compras**: Capacidade de registrar compras de produtos, com seleção do produto e quantidade, e logando essas operações. * **Gestão de Estoque com Operações Aritméticas**: Permite realizar operações como soma, subtração, multiplicação e divisão sobre as quantidades de produtos, além de registrar entradas e saídas de estoque. |
| Não funcionais: Não Funcionais:  * **Confiabilidade e Desempenho**: A aplicação depende fortemente da **disponibilidade e desempenho da API de backend** (http://127.0.0.1:5000). Qualquer indisponibilidade ou lentidão da API afetará diretamente a usabilidade do aplicativo. * **Segurança**:   + A autenticação de usuários é baseada em uma API externa, e a **segurança dos dados de usuário é gerenciada pelo backend**.   + Existe uma **senha de administrador hardcoded ("1234")** para novos cadastros de usuário, o que representa uma **vulnerabilidade de segurança** e deve ser corrigida em um ambiente de produção.   + A comunicação com a API parece ser via **HTTP não seguro**, o que pode expor dados sensíveis se não houver outras camadas de segurança. * **Usabilidade**: A interface é desenvolvida em **Windows Forms (WinForms)**, o que indica uma aplicação desktop com uma interface gráfica de usuário padrão. * **Manutenibilidade**: O código C# é modularizado por formulário, o que facilita a manutenção de cada tela individualmente. No entanto, a dependência direta de URLs de API hardcoded pode dificultar a mudança de ambientes (desenvolvimento, teste, produção). |

**DOCUMENTO DE EVIDÊNCIAS DE TESTES DO PROJETO**

Esta seção seria preenchida com os **casos de teste, resultados, evidências (prints de tela, logs)** e os status de cada teste (passou/falhou). Dada a natureza dos arquivos fornecidos (apenas código-fonte), não é possível apresentar evidências de testes reais. No entanto, o código sugere interações com uma API, o que implicaria a necessidade de testes de integração, testes de unidade para a lógica de negócios e testes de interface do usuário.

**CAMINHO DE ACESSO AO APLICATIVO:**

O appComercio é uma **aplicação desktop** (Windows Forms). Para acessá-lo, o executável (.exe) da aplicação compilada deve ser iniciado em um ambiente Windows. A aplicação cliente se conecta a uma **API de backend que deve estar rodando localmente** no endereço http://127.0.0.1:5000.

**CONCLUSÃO:**

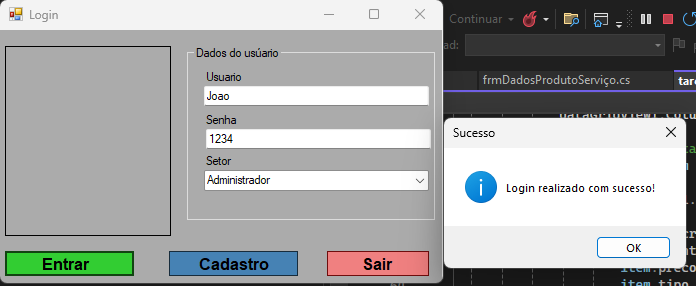
O projeto appComercio é uma **aplicação de gerenciamento comercial** desenvolvida em C# (Windows Forms) que atua como **cliente de uma API RESTful**. O sistema permite a gestão essencial de um negócio de vendas e estoque, abrangendo desde o cadastro de usuários com diferentes níveis de acesso até o controle detalhado de produtos/serviços, registro de compras e vendas, e manipulação de inventário. A arquitetura é cliente-servidor, onde o aplicativo desktop (appComercio) consome serviços de uma API de backend, que por sua vez seria responsável pela persistência dos dados em um banco de dados. A aplicação demonstra um bom nível de modularidade na interface e na lógica de interação com a API.

**A) INTERFACE:**

1 – Tela de login:

A tela de login é a porta de entrada para o sistema.

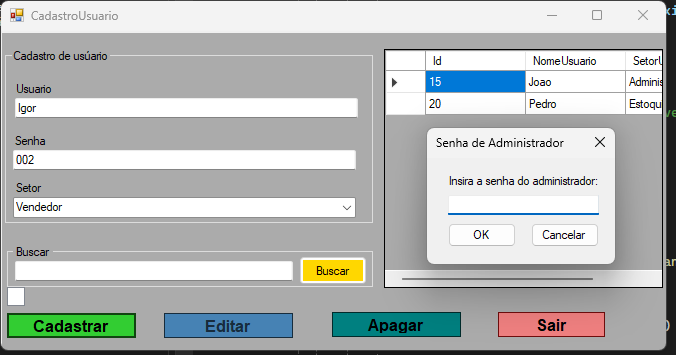
* **Elementos**: Caixas de texto para **Nome de Usuário** e **Senha**, um **ComboBox** para seleção do **Setor** (Administrador, Vendedor, Estoquista), e botões **"Entrar"** e **"Cadastrar"**.
* **Funcionalidade**: Valida as credenciais do usuário e o perfil selecionado com a API (/CadastroUsuario). Em caso de sucesso, direciona o usuário para a tela principal (frmProdutoServico). O botão "Cadastrar" permite o acesso à tela de cadastro de usuários.



2 – Tela cadastro usuários:

Esta tela é responsável pelo gerenciamento dos usuários do sistema.

* **Elementos**: Caixas de texto para **Nome de Usuário** e **Senha**, um **ComboBox** para seleção do **Setor** do usuário, um **CheckedListBox** para indicar se o usuário está ativo, um campo de busca, e botões para **"Cadastrar"**, **"Editar"**, **"Apagar"**, **"Buscar"** e **"Sair"**. Um **DataGridView** exibe a lista de usuários.
* **Funcionalidade 1**: Permite a adição de novos usuários com seus respectivos perfis, a edição de dados de usuários existentes e a exclusão de registros. Os dados são persistidos através da API (/CadastroUsuario).
* **Funcionalidade 2**: Mas Só e permitido o cadastro e a exclusão de usuário com a senha do Administrador.

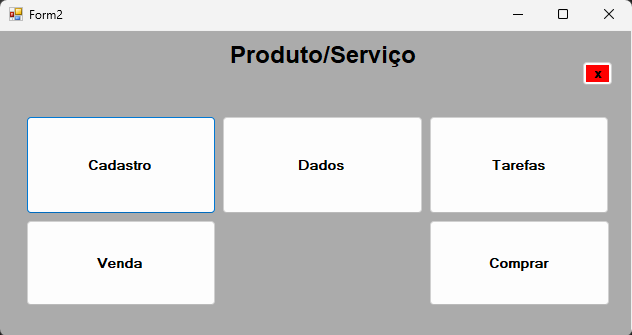


3 – Tela principal:

 **Identificada como:** frmProdutoServico

 **Função (presumida):** Painel de controle com acesso às demais funcionalidades

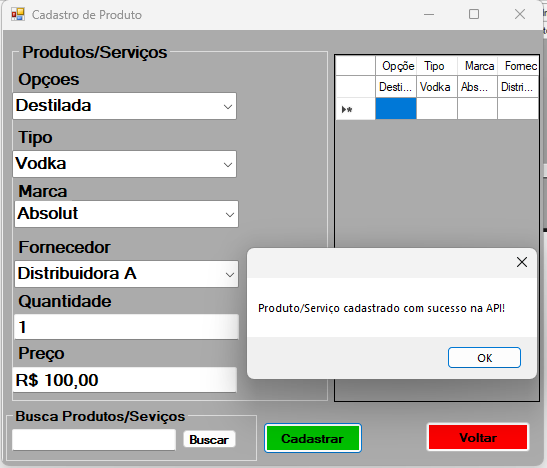
Como compra Venda tarefas Cadastro de produtos Dados de Produtos etc...



4 – Tela de cadastro produto/serviço:

Utilizada para registrar novos produtos ou serviços.

* **Elementos**: Campos de texto para **Quantidade** e **Preço**, ComboBoxes para **Opções (Categoria)**, **Tipo**, **Marca** e **Fornecedor**. Botões **"Voltar"**, **"Cadastrar"** e **"Buscar"**. Um **DataGridView** exibe os produtos/serviços.
* **Funcionalidade**: Permite inserir detalhes de um produto ou serviço, formatar o campo de preço automaticamente para "R$ X,XX", e interagir com a API para persistir, buscar, atualizar e excluir dados de produtos.



5 – Tela de dados produto/serviço:

Esta tela serve para visualizar e manipular os dados existentes de produtos e serviços.

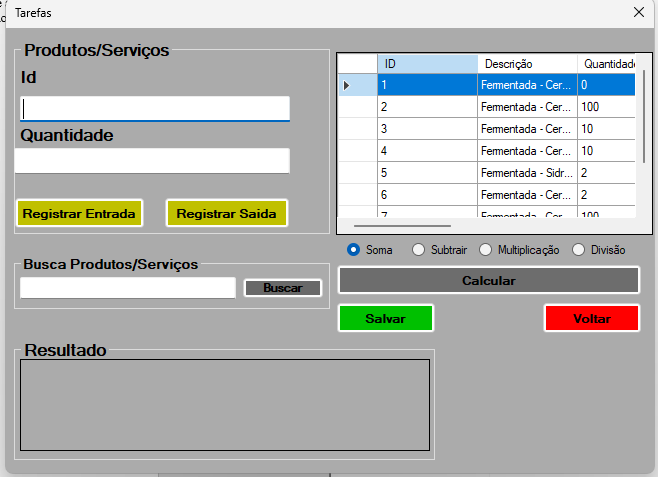
* **Elementos**: Campos de texto para **ID**, **Categoria**, **Tipo**, **Marca**, **Fornecedor**, **Quantidade** e **Preço**. Botões para **"Voltar"**, **"Exibir Dados"**, **"Buscar"**, **"Editar"**, **"Salvar"** e **"Baixar"** (provavelmente para exportar dados). Um **DataGridView** apresenta os dados.
* **Funcionalidade**: Exibe uma lista de produtos/serviços, permite a busca por filtros, a edição direta dos dados exibidos e o salvamento das alterações via API. Também inclui a funcionalidade de download dos dados.



6 – Tela de tarefas produtos/serviços:

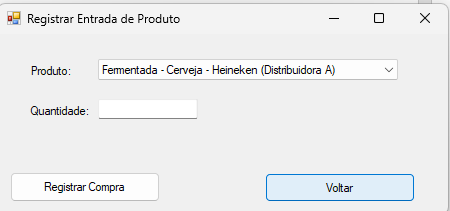
Focada na gestão de estoque e operações aritméticas com quantidades de produtos.

* **Elementos**: Caixas de texto para **ID** e **Quantidade**, botões de rádio para **Soma**, **Subtrair**, **Multiplicação** e **Divisão**, botões **"Calcular"**, **"Entrada"** e **"Saída"**. Dois **DataGridViews**, um para exibir os produtos e outro para exibir resultados.
* **Funcionalidade**: Carrega dados de produtos, permite selecionar um produto e aplicar operações aritméticas em sua quantidade. Os botões "Entrada" e "Saída" indicam a gestão de fluxo de estoque.



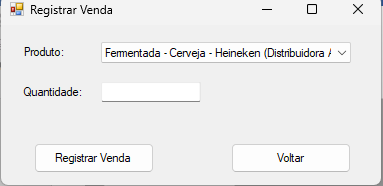
7 – Tela de Compra/produtos:

Tela Simples de Compra de Produtos onde o usuário ira poder realizar compra de produtos



8 – Tela de Venda/produtos:

Tela de Vendas de Produtos Para que também no estoque seja controlados a siada de produtos.

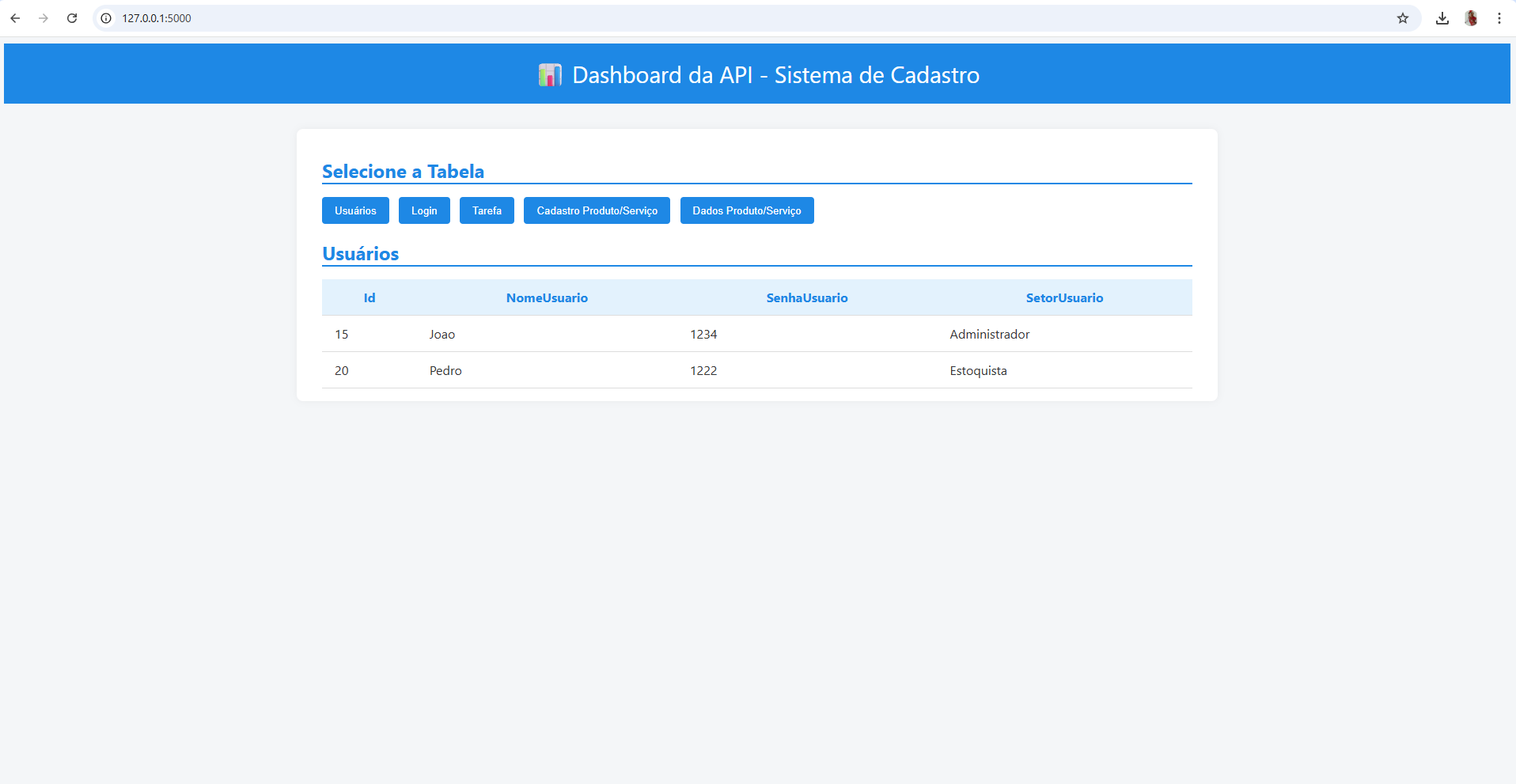


**B) BACK-END (API):**

A aplicação appComercio interage com uma **API de backend**, presumivelmente desenvolvida em Flask (dada a URL padrão http://127.0.0.1:5000), responsável pela lógica de negócio e persistência de dados. Os endpoints identificados nas chamadas HTTP incluem:

* **/CadastroUsuario**: Utilizado para gerenciar usuários (cadastro, consulta, autenticação). A aplicação realiza requisições **GET** para obter a lista de usuários para autenticação e exibição, e provavelmente **POST/PUT/DELETE** para criar, atualizar e remover usuários.
* **/DadosProdutoServico**: Usado para acessar e manipular dados de produtos e serviços. A aplicação faz requisições **GET** para carregar produtos e **POST/PUT** para registrar ou atualizar.
* **/RegistrarLogCompra**: Recebe dados de compras para registro. A aplicação envia requisições **POST**.
* **/RegistrarVenda**: Recebe dados de vendas para registro. A aplicação envia requisições **POST**.

A API atua como um **intermediário** entre a aplicação desktop e o banco de dados, processando as solicitações do cliente e gerenciando a persistência dos dados.

****

**C) BANCO DE DADOS:**

### Banco de Dados:

Com base nos arquivos C# fornecidos, a aplicação cliente (appComercio) **não interage diretamente com o banco de dados**. Toda a persistência e recuperação de dados são realizadas por meio das chamadas à **API de backend**.

Portanto, os detalhes sobre o tipo de banco de dados (SQL Server, MySQL, PostgreSQL, SQLite, etc.), o esquema das tabelas e os procedimentos de acesso ao banco de dados **não estão presentes no código-fonte do cliente C# analisado**. Essas informações residiriam no código-fonte da API de backend, que não foi fornecido.