Preparação de Artigos no Formato de Duas Colunas

Centro Universitário Padre Anchieta

Joaquim da Silva dos Santos RA: 2502410

Gabriel Filipe dos Santos RA: 2517645

Italo Guilherme Pinheiro Rodrigues RA: 2526314

Pedro Henrique Assis de Oliveira RA: 2508585

Matheus Gomes Nagy RA: 2508094

Resumo - Este artigo apresenta o desenvolvimento e implementação de um sistema de gestão para uma pizzaria utilizando a linguagem TypeScript e organização de dados em arquivos CSV (Comma-Separated Values), que foi desenvolvido pela equipe de alunos do curso de Ciências da computação. Onde o objetivo é apresentar a estruturação e construção de um prompt ao qual é simulado a montagem de cadastro de clientes e produtos, gestão de pedidos e geração de recibos, isso tudo estará em um menu interativo dentro de um comércio de pizzaria. Também será apresentado um protótipo do design de como será o site para uso dos clientes desta pizzaria através do uso de HTML (Hypertext Markup Language).

I. INTRODUÇÃO

Nos dias atuais é raro encontrar algum comércio ou estabelecimento de prestação de serviço que não tenha acoplado aos seus processos, um software para gestão de pessoas e demandas. O grupo entende que essa gestão eficiente de estabelecimentos comerciais, especialmente no setor de alimentação, representa um desafio constante para empreendedores que buscam se manter atualizados tecnologicamente dentre os concorrentes. A necessidade de controlar cadastros, processar pedidos e gerar relatórios de forma organizada é fundamental para o sucesso operacional e financeiro. Tradicionalmente, muitas pequenas empresas ainda utilizam planilhas eletrônicas, dependendo, até meios manuais podem são utilizados, o que pode levar a erros, perda de informações e ineficiência operacional.

Diante disso, este trabalho propõe uma solução tecnológica acessível para uma pizzaria, como exemplo, desenvolvida em TypeScript (uma linguagem que adiciona tipagem forte ao JavaScript) com persistência de dados em arquivos CSV (Comma-Separated Values). A escolha dessas tecnologias justifica-se pela portabilidade, simplicidade de implementação e baixo custo de manutenção. Além de ter sido um dos requisitos para estruturação deste artigo.

O sistema implementado agrega funcionalidades essenciais para a operação diária de uma pizzaria, organizadas em módulos lógicos que permitem a expansão e adaptação conforme as necessidades do negócio.

II. Revisão da arquitetura do sistema

II.I Estrutura de Dados

É importante entendermos que o sistema que foi desenvolvido e analisado pela equipe, foi construído usando três tipos principais de dados escritos em TypeScript. Esses tipos de dados representam “Produtos”, “Clientes” e “Pedidos”. Cada um deles define exatamente quais informações existem no menu interativo e o que o usuário deve fornecer de informação, como o nome do produto, o telefone do cliente, após isso o valor total de um pedido será apresentado. Essa definição mais “básica” ajuda a evitarmos erros comuns, como colocar um valor errado em um campo ou tentar acessar algo que não existe.

As três entidades principais definidas pelo sistema (mencionadas acima) através de tipos TypeScript serão demonstradas a seguir:

```
  typescript
type Produto = {
  id: number;
```

```
    tipo: 'pizza' | 'bebida' | 'sobremesa';
    nome: string;
    valor: number;
};

type Cliente = {
    id: number;
    nome: string;
    telefone: string;
    endereco: string;
};

type Pedido = {
    data: string;
    idcliente: number;
    idproduto: number;
    custototal: number;
    formapagamento?: 'dinheiro' | 'cartao' | 'pix';
};
```

Como exemplificado, foi a montagem do código foi planejada e arquiteta desta maneira para redução de erros ao qualquer usuário utilizar a interface, os chamados *"usability error"* (quando a estrutura do sistema leva o usuário ao engano) e o *"input error"* (quando o usuário digita dados inválidos).

II.II Estratégia de Persistência

Uma parte muito relevante para a versatilidade de implementação do código em diferentes lugares e seu compartilhamento, é o formato de arquivo em que ele está. Para guardar as informações, o sistema montado usa arquivos CSV (um formato de texto simples em que cada linha é um registro). Essa escolha torna o sistema fácil de usar e manter, pois os arquivos podem ser abertos em ferramentas como Excel ou Google Sheets. Também facilita fazer backups (cópias de segurança) e elimina a necessidade de um servidor de banco de dados.

Os arquivos ficam organizados em pastas específicas: um arquivo para produtos, outro para clientes, outro para pedidos e uma pasta para guardar recibos que também são gerados em TXT (bloco de notas). Os motivos de essa ser a alternativa escolhida pelo grupo são:

· Simplicidade: Formato legível e amplamente suportado
· Portabilidade: Compatível com diversas ferramentas (Excel, Google Sheets, etc.)
· Backup: Facilidade de cópia e recuperação de dados
· Baixo overhead: Não requer servidor de banco de dados dedicado

O sistema organiza os arquivos em estrutura de diretórios de forma específica:

· produtos.csv: Catálogo completo de produtos
· clientes.csv: Base de clientes cadastrados
· pedidos.csv: Histórico de vendas
· recibos/: Diretório para comprovantes de pedidos

III.. Metodologia de implementação

III.I Inicialização e Configuração

Logo ao iniciar, o sistema verifica se todas as pastas e arquivos necessários existem e se estão sem alguma inconformidade. Caso não existam essas pastas, ele os cria automaticamente com seus cabeçalhos (os nomes das colunas). Fizemos isso para que nada "quebre" ao tentar gravar informações, como por exemplo, guardar o pedido.

Outro ponto importante é a geração de IDs únicos (identificadores que não se repetem). Como tudo é salvo em arquivos e não em um banco de dados, o sistema precisa "ler" os arquivos, descobrir quais IDs já existem e calcular qual é o próximo disponível. Isso evita duplicações, mesmo que o programa seja fechado e aberto várias vezes.

III..III Gestão de Identificadores

Um desafio significativo em sistemas baseados em arquivo é a geração de IDs únicos. O sistema resolve isso através das funções inicializaIdProdutos() e inicializaIdClientes(), que:

1. Leem o arquivo correspondente
2. Extraem todos os IDs existentes
3. Calculam o próximo ID disponível
4. Garantem a unicidade mesmo após reinicializações

III.III Interface de Usuário

A interface baseada em linha de comando é organizada hierarquicamente:

IV. FUNCIONALIDADES IMPLEMENTADAS

IV.I Módulo de Cadastros

Clientes: Implementa operações CRUD (Create, Read, Update, Delete) completas com validação de telefone (11 dígitos) e persistência imediata.

Produtos: Gerencia produtos categorizados (pizza, bebida, sobremesa) com validação de tipo e valor numérico positivo.

IV.II Módulo de Pedidos

O processo de pedidos é particularmente sofisticado:

1. Seleção de Cliente: Verificação de existência na base
2. Seleção de Produtos por Categoria: Interface organizada por tipo
3. Cálculo Automático: Soma dos valores dos produtos
4. Promoção de Primeira Compra: Sistema detecta clientes novos e aplica 20% de desconto
5. Seleção de Pagamento: Suporte a dinheiro, cartão e PIX
6. Geração de Recibo: Arquivo texto formatado com detalhes completos

IV.III Sistema de Relatórios

O módulo analítico oferece:

· Vendas Diárias/Mensais: Agregação por período
· Produtos Mais Vendidos: Ranking por valor total
· Clientes Mais Frequentes: Identificação dos melhores clientes
· Análise por Forma de Pagamento: Distribuição dos métodos de pagamento

V. ANÁLISE DE CÓDIGO E DECISÕES TÉCNICAS

V.I Tratamento de Erros

O sistema implementa tratamento robusto de exceções através de blocos try-catch, garantindo que erros de leitura/escrita de arquivos não interrompam a execução.

V.II Validação de Dados

Validações em tempo real são implementadas para:

· Formato de telefone (regex ^\d{11}$)
· Tipos de produtos (enumeração restrita)
· Valores numéricos positivos
· Existência de registros referenciados

V.III Programação Assíncrona

O uso consistente de async/await garante que operações de I/O não bloqueiem a interface, mantendo a responsividade do sistema mesmo com arquivos grandes.

VI. RESULTADOS E DISCUSSÃO

O sistema desenvolvido atende com eficácia aos requisitos operacionais de uma pizzaria, demonstrando que soluções baseadas em tecnologias web podem ser adaptadas para aplicações desktop através de CLI.

Vantagens Identificadas:

· Baixo Custo: Não requer licenças de software
· Simplicidade: Interface intuitiva mesmo para usuários não técnicos
· Manutenibilidade: Código bem estruturado e documentado
· Extensibilidade: Arquitetura modular permite adição de funcionalidades

VII. CONCLUSÕES E TRABALHOS FUTUROS

O sistema de gestão para pizzaria desenvolvido em TypeScript com persistência em CSV mostrou-se uma solução viável e eficiente para pequenos estabelecimentos. A arquitetura proposta atende aos requisitos fundamentais de cadastro, pedidos e relatórios, enquanto a tipagem estática do TypeScript contribui para a robustez do código.

Como trabalhos futuros, sugere-se:

1. Implementação de interface web
2. Migração para banco de dados SQLite
3. Desenvolvimento de módulo de entregas
4. Integração com APIs de pagamento
5. Sistema de backup automático

REFERÊNCIAS BIBLIOGRÁFICAS

https://www.typescriptlang.org/

https://www.opservices.com.br/dicionario-da-ti/

https://www.devmedia.com.br/guia/linguagem-java/38169

https://www.significados.com.br/id/

https://www.freecodecamp.org/portuguese/news/o-que-e-um-arquivo-csv-e-como-abrir-esse-formato-de-arquivo/

https://help.salesforce.com/s/articleView?id=sf.bi_app_sales_wave_quota_csv.htm&type=5&language=pt_BR

https://github.com/?locale=pt-BR