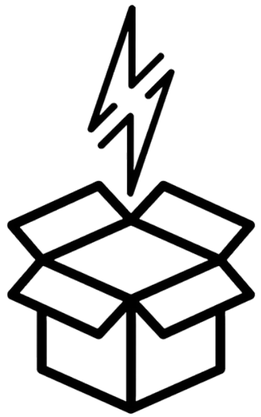


STOCKIFY

A solução para seus problemas de estoque

Gabriel Filipe - Luca Gonzaga - Paulo Henrique - Arthur Camargo

CONTEXTUALIZAÇÃO

O projeto "Stockify" visa solucionar a dificuldade na gestão de estoque em bares e restaurantes, oferecendo uma plataforma que integra e automatiza esses processos. O objetivo é simplificar a gestão de produtos, inventário, pedidos e fornecedores. As principais motivações incluem a redução de desperdícios, a otimização dos processos de pedidos e reposição, e a melhoria da eficiência operacional dos bares e restaurantes.

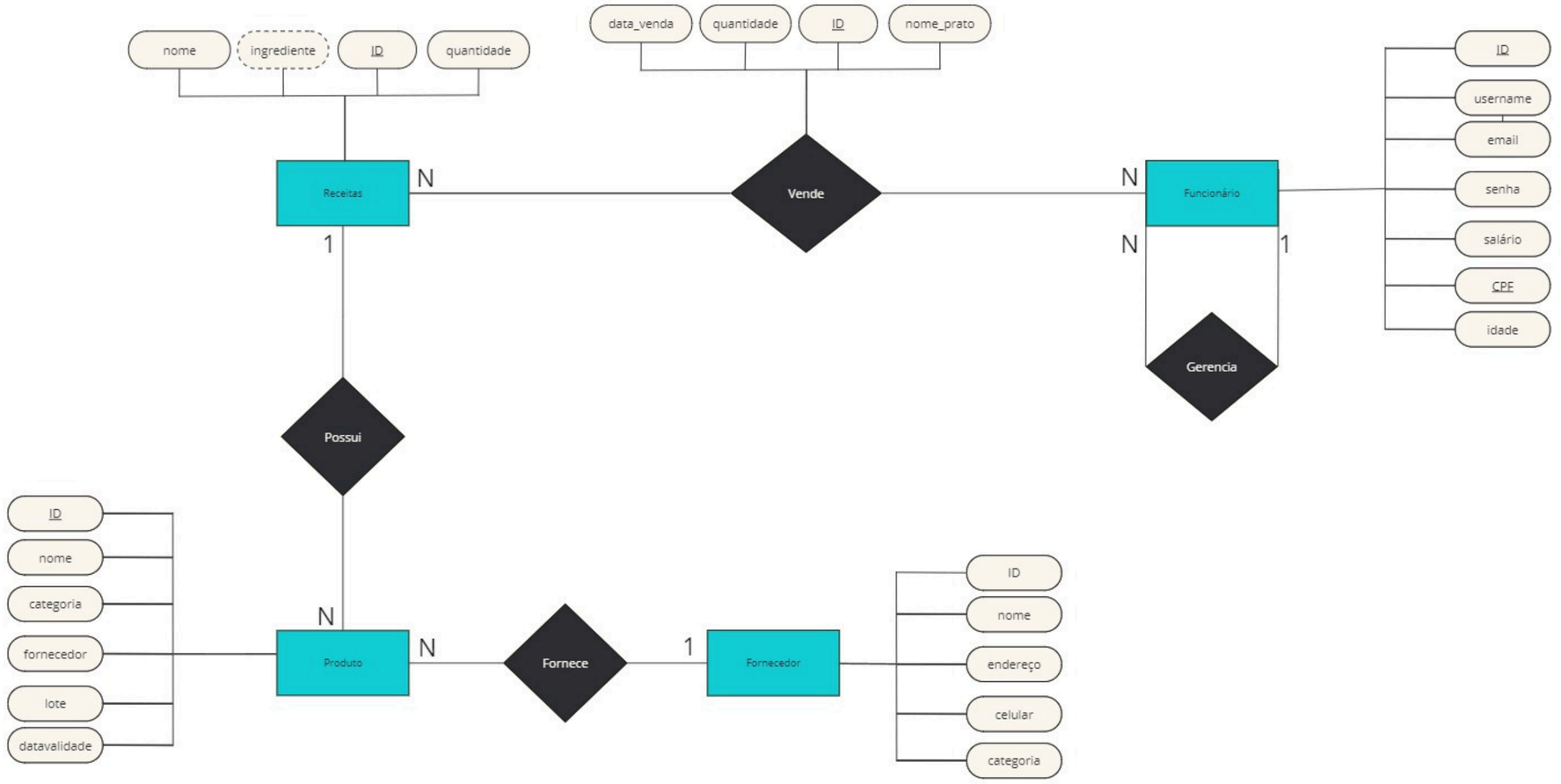
FERRAMENTAS UTILIZADAS

- Discord
- Miro
- PostgreSQL
- VsCode
- GitHub




https://github.com/ICEI-PUC-Minas-CC-TI/plmg-cc-ti2-2024-1-g04-stockify

DIAGRAMA


nome
ingrediente
ID
quantidade
data_venda
quantidade
ID
nome_prato
Receitas
N
Vende
N
Funcionário
ID
username
email
senha
salário
CPF
idade
1
N
1
Gerencia
Possui
ID
nome
categoria
fornecedor
lote
datavalidade
N
Produto
N
Fornece
1
Fornecedor
ID
nome
endereço
celular
categoria

IA CANVAS

Ferramental IA

1. Análise de Séries Temporais: Para detectar padrões de vendas ao longo do tempo.
2. Médias Móveis: Para calcular a média de vendas diárias por dia da semana.
3. Algoritmo de Regressão: Para prever a demanda futura com base em dados históricos.
4. Integração com bancos de dados.

Entradas

1. Data de venda.
2. Nome do prato vendido.
3. Quantidade de itens vendidos.
4. Histórico de vendas.
5. Informações sobre o estoque atual.

Saídas

1. Data prevista de falta do estoque.
2. Quantidade do estoque restante em tempo real.
3. Relatórios de previsão de demanda.

Proposição de valor

1. Redução de rupturas de estoque.
2. Melhor planejamento de compras.
3. Aumento da satisfação de clientes devido à disponibilidade de produtos.
4. Economia de custos operacionais.

Equipe

1. Cientista de dados.
2. Engenheiro de software.
3. Gestor de projeto.

Stakeholders Chaves

1. Gerente do estabelecimento.
2. Diretor financeiro.

Clientes

1. Gerentes de estoque.
2. Equipe de compras.
3. Equipe de vendas,
4. Fornecedores.
5. Clientes finais.

Custos

1. Salários de desenvolvedores.
2. Infraestrutura de TI (banco de dados, hospedagem do site)

Receitas

1. Possível venda da aplicação para outros estabelecimentos.
2. Redução de perdas de estoque.
3. Menor urgência em pedido de reposição.

ROTAS

PRODUTO

```
//Rotas para lidar com os produtos
        post("/produto/insere", (request, response) -> produtoService.insert(request, response));
        get("/produto/getAll", (request,response) -> produtoService.getAll(request, response));
        get("/produto/fornecedores", (request, response) -> produtoService.getAllFornecedores(request, response));
        get("/produto/:id", (request, response) -> produtoService.getById(request, response));
        put("/produto/atualizar/:id", (request, response) -> produtoService.atualizarProduto(request, response));
        delete("/produto/excluir/:id", (request, response) -> produtoService.excluirProduto(request, response));

```



```
public boolean atualizarProduto(Request request, Response response) {
        Gson gson = new Gson();
        Produto produto = gson.fromJson(request.body(), Produto.class);

        try {
            return produtoDAO.update(produto);
        } catch (Exception e) {
            System.out.println("Erro ao atualizar no service: " + e.getMessage());
            return false;
        }
    }
```



```
public boolean update(Produto produto) {
        boolean status = false;
        try {
            String sql = "UPDATE produto SET nome = ?, categoria = ?, quantidade = ?, fornecedor = ?, lote = ?, "
                    + "datavalidade = ? WHERE id = ?";
            PreparedStatement st = conexao.prepareStatement(sql);
            st.setString(1, produto.getNome());
            st.setString(2, produto.getCategoria());
            st.setInt(3, produto.getQuantidade());
            st.setString(4, produto.getFornecedor());
            st.setString(5, produto.getLote());
            st.setDate(6, Date.valueOf(produto.getDatavalidade()));
            st.setInt(7, produto.getId());
            st.executeUpdate();
            st.close();
            status = true;
        } catch (SQLException u) {
            throw new RuntimeException(u);
        }
        return status;
    }
```

ROTAS

FORNECEDOR

```
//Rotas fornecedores
        post("/fornecedor/insere", (request, response) -> fornecedorService.inserirFornecedor(request, response));
        get("/fornecedor/getAll", (request, response) -> fornecedorService.getAll(request, response));
        get("/fornecedor/:id", (request, response) -> fornecedorService.getById(request, response));
        put("/fornecedor/atualizar/:id", (request, response) -> fornecedorService.atualizarFornecedor(request, response));
        delete("/fornecedor/excluir/:id", (request, response) -> fornecedorService.excluirFornecedor(request, response));

```

```
public boolean inserirFornecedor(Request request, Response response) {
        Gson gson = new Gson();
        Fornecedor fornecedor = gson.fromJson(request.body(), Fornecedor.class);
        try {
            return fornecedorDAO.inserir(fornecedor);
        } catch (Exception e) {
            System.out.println("Erro ao inserir fornecedor no serviço: " + e.getMessage());
            e.printStackTrace(); // Registrar a pilha de chamadas completa para diagnóstico
            return false;
        }
    }
```

```
public boolean inserir(Fornecedor fornecedor) {
      try {
          if (conexao == null) {
              throw new SQLException("Objeto de conexão não foi inicializado corretamente.");
          }
          String sql = "INSERT INTO fornecedor (nome, endereco, celular, categoria) VALUES (?, ?, ?, ?)";
          PreparedStatement stmt = conexao.prepareStatement(sql);
          stmt.setString(1, fornecedor.getNome());
          stmt.setString(2, fornecedor.getEndereco());
          stmt.setString(3, fornecedor.getCelular());
          stmt.setString(4, fornecedor.getCategoria());
          int linhasAfetadas = stmt.executeUpdate();
          return linhasAfetadas > 0;
      } catch (SQLException e) {
          e.printStackTrace();
          return false;
      }
  }
```

CRIPTOGRAFIA

SENHAS

```java
// Rota para lidar com o login/cadastro do usuário
put("/login/procura", (request, response) -> usuarioService.login(request, response));
post("/login/insere", (request, response) -> usuarioService.criarUsuario(request, response));
```

```java
public Object criarUsuario(Request request, Response response) {
    Gson gson = new Gson();
    Usuario usuario = gson.fromJson(request.body(), Usuario.class);

    // Imprimir os valores dos parâmetros
    System.out.println("Email recebido: " + usuario.getEmail());
    System.out.println("Senha recebida: " + usuario.getSenha());
    System.out.println("Nome recebido: " + usuario.getUsername());
    System.out.println("Salário recebido: " + usuario.getSalario());
    System.out.println("CPF recebido: " + usuario.getCpf());
    System.out.println("Idade recebida: " + usuario.getIdade());

    // Verificar se o usuário já existe
    Usuario usuarioExistente = usuarioDAO.buscarUsuarioPorEmail(usuario.getEmail());
    if (usuarioExistente != null) {
        response.status(400); // Bad Request
        return "Já existe um usuário cadastrado com este email.";
    }

    // Criptografar a senha antes de salvar o usuário
    String senhaCriptografada = BCrypt.hashpw(usuario.getSenha(), BCrypt.gensalt());
    usuario.setSenha(senhaCriptografada);

    usuarioDAO.salvarUsuario(usuario);
    response.status(201);
    return gson.toJson(usuario);
}
```

```java
public void salvarUsuario(Usuario usuario) {
    try {
        String sql = "INSERT INTO users (username, email, senha, salario, cpf, idade) VALUES (?, ?, ?, ?, ?, ?)";
        PreparedStatement st = conexao.prepareStatement(sql);
        st.setString(1, usuario.getUsername());
        st.setString(2, usuario.getEmail());
        st.setString(3, usuario.getSenha());
        st.setDouble(4, usuario.getSalario());
        st.setString(5, usuario.getCpf());
        st.setInt(6, usuario.getIdade());
        st.executeUpdate();
        st.close();
    } catch (SQLException e) {
        throw new RuntimeException("Erro ao salvar usuário", e);
    }
}
```

id	username	email	senha	salario	cpf
1	admin	admin@admin	$2a$10$xT3K19ACcz6mC9uzElL2neokJ5ZaHxfwu	0	12312312312
35	Paulo Henrique	paulohenriqueldp@gmail.com	$2a$10$uYz7XYrGCTa7oix/fwoOmuSzVrlZPxQB	1.200	16203738662
36	Teste	teste@teste.com	$2a$10$y7Wa3JX2HltMvmWxS/pINORZ9MF/jp	1.200	12312312312

OBRIGADO!

Fiquem agora com a apresentação prática da nossa aplicação.