



Storage Solutions

Solução em gestão de estoques

INTEGRANTES



João Vitor Lima de Melo



**Leon Junio Martins
Ferreira**



**Luis Fellyp Madeira
Euzébio e Lacerda**



Vinícius Augusto Moreira Santos

APRESENTAÇÃO

01



Contextualização do projeto

Descrição da proposta do trabalho
do grupo

02



Implementação Completa do Sistema

Conexão front-end, back-end e
banco de dados

03



Sistema Inteligente

Projeto e implementação da
aplicação inteligente

04



Conclusão


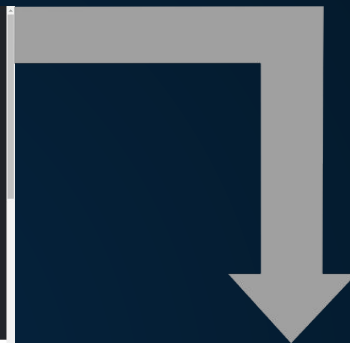
Considerações finais da execução
do projeto



**STORAGE
SOLUTIONS**

O NOSSO PROJETO

Solução de controle de estoque para empresas que não utilizam bancos de dados relacionais para a gestão dos produtos no inventário.



Entrar para seu Storage


Email

Senha

[Registrar-se](#)

[Entrar](#)

Ainda não tem uma conta? [Crie uma](#)



Storage
Solutions


 Usuário


 Dashboard


 Sair

ss@ti

Lista de Estoque

Nome	Descrição	Quantidade	Ações
Estoque São José	Estoque ativo!	100	  
Estoque São gabriel	Estoque ativo!	30	  

 Adicionar Estoque





Cadastro de Estoque

Nome do Estoque

Capacidade de Estoque

Cadastrar Estoque

Meu estoque: **Estoque São José**

Nova Remessa

Listar Produtos

Nova Retirada

Histórico de Transações

Listar Fornecedores

Cadastro de Produto

Nome do Produto

Descrição do Produto

Quantidade

Código de Barras

Unidade

AMPOLA

Ano de Fabricação

dd/mm/aaaa

Ano de Validade

dd/mm/aaaa

Marca do Produto

Peso

Adicionar Produto

Entrada de Produtos

Produtos

CARREGANDO

Quantidade

Data da entrada

dd/mm/aaaa

Criar entrada do produto

CONEXÃO BACK-END COM O BANCO

```
package dao;

import java.sql.*;

import utils.HashUtils;

public class DAO {
    protected Connection conexao;

    public DAO() {
        conexao = null;
    }

    public boolean conectar() {
        String driverName = "org.postgresql.Driver";
        String serverName = "localhost";
        String mydatabase = "StorageSolutionsDB";
        int porta = 5432;
        String url = "jdbc:postgresql://" + serverName + ":" + porta + "/" + mydatabase;
        String username = "ti2cc";
        String password = "ti@cc";
        boolean status = false;

        try {
            Class.forName(driverName);
            conexao = DriverManager.getConnection(url, username, password);
            status = (conexao != null);
            System.out.println("Conexão efetuada com o postgres!");
        } catch (ClassNotFoundException e) {
            System.err.println("Conexão não efetuada com o postgres -- Driver não encontrado -- " + e.getMessage());
        } catch (SQLException e) {
            System.err.println("Conexão não efetuada com o postgres -- " + e.getMessage());
        }

        return status;
    }
}
```


APLICACAO.JAVA

```
/**
 * REQUISICOES DOS SERVIÇOS DE PRODUTO
 */

get("/produto/listar/:id", (request, response) -> produtoService.listar(request, response));
get("/produto/deletar/:idestoque/:idproduto", (request, response) -> produtoService.deletar(request, response));
post("/produto/cadastro", (request, response) -> produtoService.cadastro(request, response));
post("/produto/atualizar", (request, response) -> produtoService.atualizar(request, response));
get("/produto/carregar/:id", (request, response) -> produtoService.carregar(request, response));
post("/produto/retirada/:token", (request, response) -> produtoService.retirada(request, response));
post("/produto/entrada/:token", (request, response) -> produtoService.entrada(request, response));
get("/produto/loadlist/:id/:token", (request, response) -> produtoService.loadOptions(request, response));
get("/produto/getQtd/:idproduto", (request, response) -> produtoService.getQtd(request, response));
get("/produto/historico/:id", (request, response) -> produtoService.getHistorico(request, response));
```

PRODUTO.SERVICE

```
public Object cadastro(Request request, Response response) throws Exception {
    String nome = request.queryParams("nome");
    String descricao = request.queryParams("descricao");
    int qtd = Integer.parseInt(request.queryParams("quantidade"));
    String codigoBarras = request.queryParams("codBarras");
    String unidade = request.queryParams("unidade");

    int estoque = Integer.parseInt(request.queryParams("id-estoque"));

    // pegar datas passadas como string
    java.util.Date fabricacao = dateFormatted.parse(request.queryParams("fabricacao"));
    java.util.Date validade = dateFormatted.parse(request.queryParams("validade"));

    String marca = request.queryParams("marca");
    float peso = Float.parseFloat(request.queryParams("peso"));

    boolean resp = daop.insert(new Produto(0, qtd, estoque, nome, descricao, codigoBarras, unidade, marca, peso,
        new java.sql.Date(fabricacao.getTime()), new java.sql.Date(validade.getTime())));

    if (resp) {
        response.status(201);

        return "<script>alert('Produto cadastrado com sucesso!'); window.location.href = '" + app.Aplicacao.url
            + "/pages/home-produtos.html?id=" + estoque + "';</script>";
    } else {
        response.status(203);

        return "<script>alert('NÃO foi possível cadastrar o Produto!'); window.location.href = '"
            + app.Aplicacao.url + "/pages/home-form-produto.html?id=" + estoque + "';</script>";
    }
}
```

DAOPRODUTO.JAVA

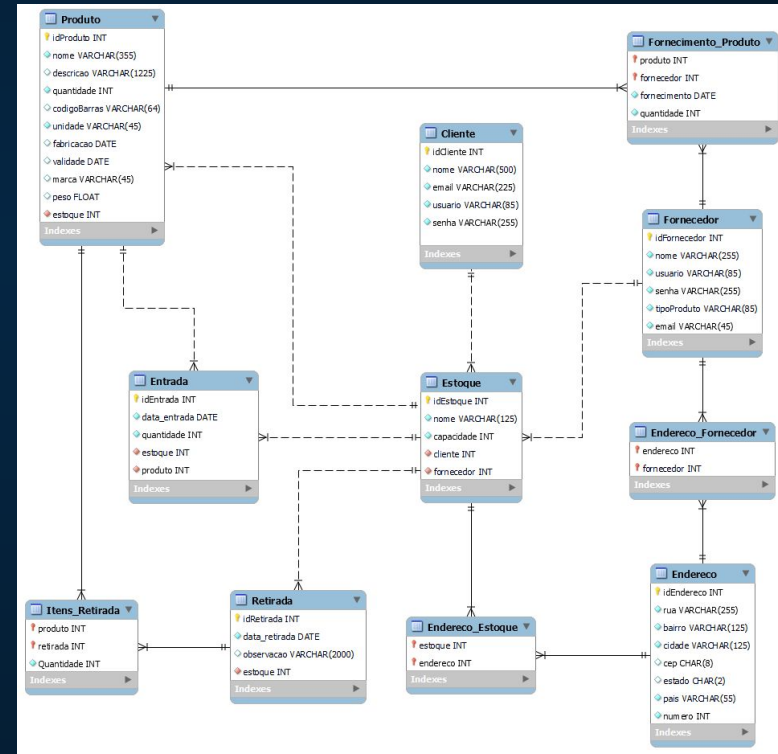
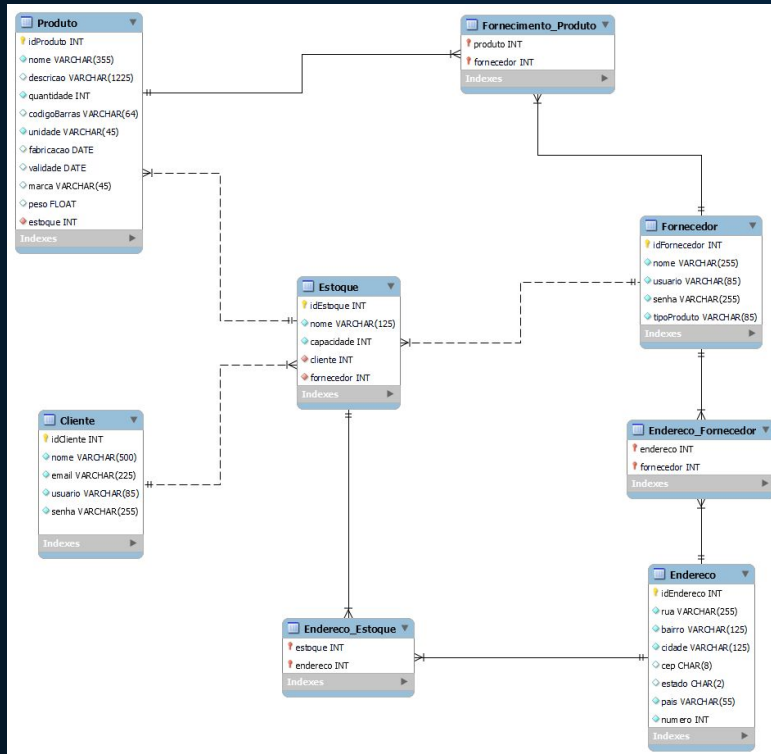
```
public boolean insert(Produto produto) {
    boolean status = false;
    try {
        String sql = "INSERT INTO StorageSolutionsDB.produto (quantidade,estoque,nome,descricao,codigoBarras,unidade,marca,peso,fabricacao,validade) "
            + "VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";
        PreparedStatement st = conexao.prepareStatement(sql);
        st.setInt(1, produto.getQuantidade());
        st.setInt(2, produto.getEstoque());
        st.setString(3, produto.getNome());
        st.setString(4, produto.getDescricao());
        st.setString(5, produto.getCodigoBarras());
        st.setString(6, produto.getUnidade());
        st.setString(7, produto.getMarca());
        st.setFloat(8, produto.getPeso());
        st.setDate(9, produto.getFabricacao());
        st.setDate(10, produto.getValidade());
        st.executeUpdate();
        st.close();
        status = true;
    } catch (SQLException u) {
        throw new RuntimeException(u);
    }
    return status;
}
```

SEGURANÇA

```
public boolean insert(Cliente cliente) {
    boolean status = false;
    try {
        String sql = "INSERT INTO StorageSolutionsDB.cliente (nome,email,usuario,senha) "
            + "VALUES (?, ?, ?, ?)";
        PreparedStatement st = conexao.prepareStatement(sql);
        st.setString(1, cliente.getNome());
        st.setString(2, cliente.getEmail());
        st.setString(3, cliente.getUsuario());
        st.setString(4, HashUtils.getHashMd5(cliente.getSenha()));
        st.executeUpdate();
        st.close();
        status = true;
    } catch (SQLException u) {
        throw new RuntimeException(u);
    }
    return status;
}
```

Armazenamento de senhas e prevenção contra ataques SQL Injection

EVOLUÇÃO DO PROJETO DE BD



SISTEMA INTELIGENTE



CHATBOT

Serviço de Atendimento ao Cliente para solucionar as demandas dos usuários



Webservice:

<https://botstoragesolutions-bot-b335.azurewebsites.net/api/messages>

SISTEMA INTELIGENTE

QnA Maker



Desenvolvimento do Banco
de Dados da IA

Language Studio



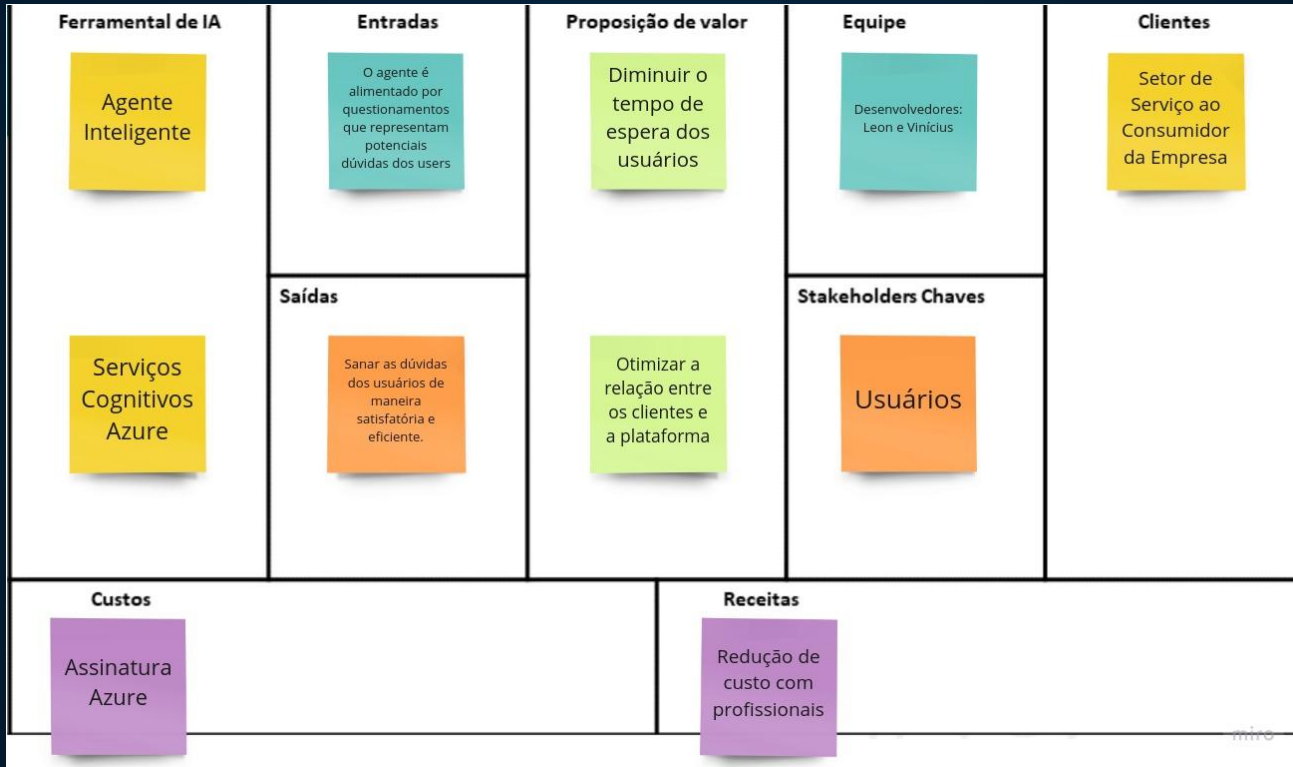
Criação do modelo do
chatbot

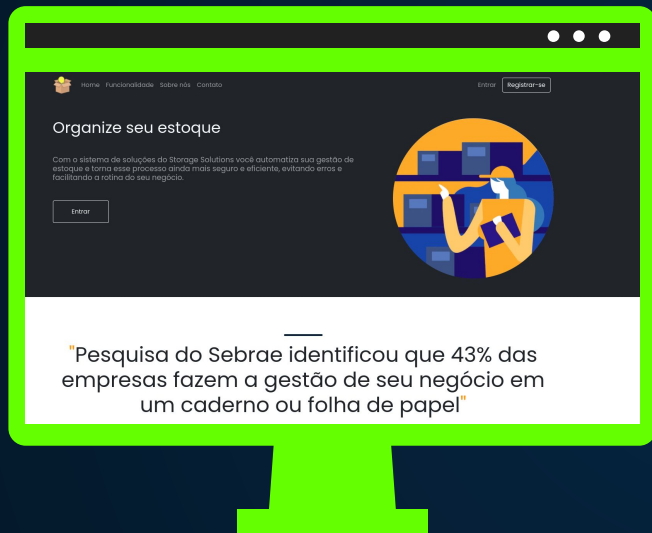
Azure Bot Service



Serviço de host do bot e
aplicativos azure

IS CANVAS





CONSIDERAÇÕES FINAIS

- O grupo executou aquilo que propôs durante a definição do projeto;
- O grupo desenvolveu funções a mais que o proposto como um sistema de autenticação por tokens, controle de sessão e retorno de respostas do servidor em JSON/Scripts;
- A pesquisa de produtos cadastrados por fornecedores para conectar a *supply chain* é uma funcionalidade a ser desenvolvida para aprimorar a plataforma.

Obrigado
pela atenção!