EasyPeasy

EasyPeasy: Facilitando a Vida Digital

Uma solução para promover autonomia e inclusão digital para os avós.

O PROBLEMA

- Interfaces complexas: Dificuldade em navegar em sites e aplicativos de compra.
- Pagamentos confusos: Insegurança e complexidade no uso de cartões online.
- Insegurança online: Medo de errar na hora do processo de compra.

A SOLUÇÃO: EASY PEASY

App Simples

Interface intuitiva, pensada para o usuário sênior.

Lista de Desejos

Cadastro de produtos sem a necessidade de finalizar a compra.

Acesso do Compartilhado

Acesso simplificado para o apoio na conclusão da compra.

O IMPACTO ESPERADO

- **Autonomia do Idoso:** Poder escolher e solicitar produtos de forma independente.
- **Confiança do Familiar:** Segurança de que as transações são gerenciadas de forma responsável.
- **Inclusão Digital:** Superação da barreira tecnológica e acesso ao e-commerce.

Quem e Porquê Usará o EasyPeasy?

Nossas Personas refletem as necessidades centrais que o sistema EasyPeasy busca atender, garantindo que o desenvolvimento seja focado no usuário.

Como Idoso (Adalberto, 62 anos)

Eu, Dona Maria (60 anos), quero registrar um produto que vi na Shopee com nome, loja e link, para que minha neta finalize a compra sem eu precisar usar cartão. Eu só preciso de um lugar simples para anotar o que quero.

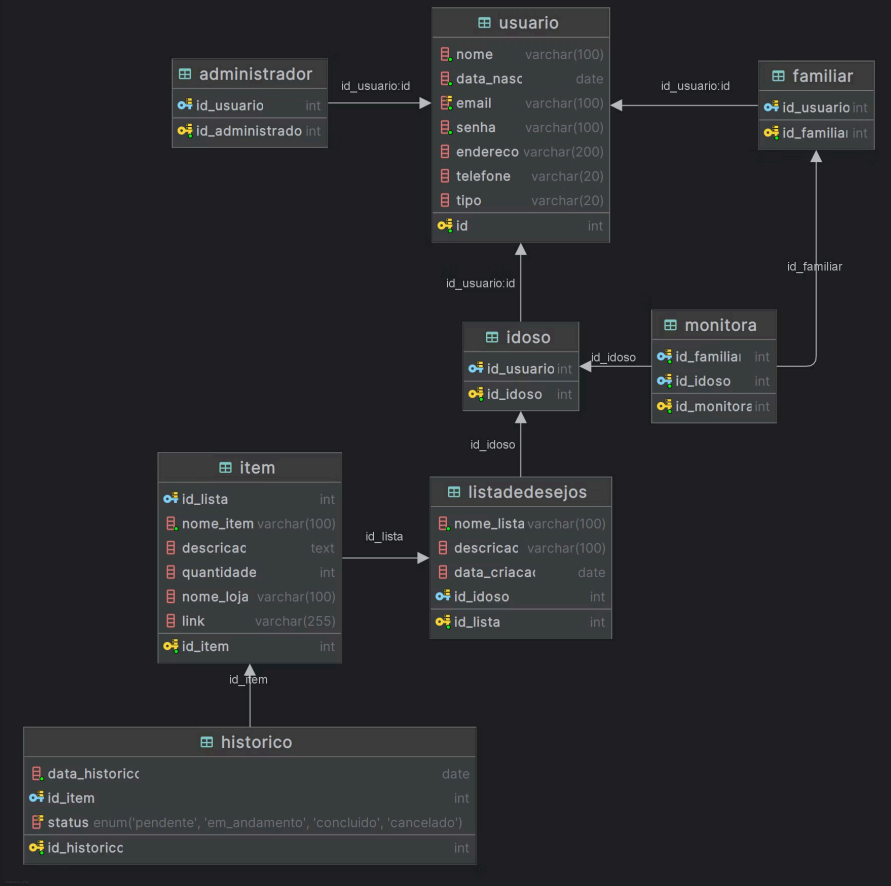
Como Familiar (Maria, 20 anos)

Eu, Maria (20 anos), quero ver a lista de desejos do minha avó e atualizar o status quando eu comprar o item, para que ela saiba que o pedido foi concluído.

Como Administrador do Sistema

Eu, o administrador, quero gerenciar todos os usuários (listar, editar, excluir), para garantir a segurança, integridade e bom funcionamento do sistema como um todo.

Arquitetura do Sistema EasyPeasy



Estrutura do Código: Organização por Pacotes e Responsabilidades

Organização em JAVA, focado em clareza e reuso de código.

1. Model

- **usuario.java** (classe abstrata)
- (idoso.java, familiar.java)
- **listaDeDesejos.java** e
- **item.java**
- **historico.java**
- **status.java (ENUM)**

2. App

- **Main.java: Classe responsável por rodar a aplicação em ordem e funcionamento correto.**

3. DAO

- **Conexao.java**: Gerencia a conexão com o banco.
- **CrudInterface.java** Interface
- **CrudDAO.java**: Classe genérica
- **UsuarioDAO.java**: Pesquisa de atributos dos usuários no banco de dados.
- **MonitoraDAO.java**: Gerencia o vínculo entre Idoso e Familiar.
- **ListaDeDesejosDAO.java**: Gerencia acesso das listas.
- **ItemDAO.java**: Cria itens associados a uma lista.
- **Historico.java**: Atualiza o status do pedido e datas de criação.

4. Controller

- **UsuarioController**: Lógica de gerenciamento de usuários.
- **LoginController.java**: Lógica de autenticação.
- **CadastroController.java**: Lógica de criação de novos usuários.
- **ListaDeDesejosController.java**: Gerencia a lista e itens.
- **MonitoraController.java**: Vinculação de usuários.
- HomeController: Seleciona Home personalizada para o tipo de usuário logado
- ItemListaController: Logica de gerenciamento dos itens associados a uma lista.

5. View

- **Menus.java** e **Listas.java**: Exibição em console

A utilização de um **CrudDAO** genérico garante que o código seja reutilizável e mais limpo, minimizando a repetição de consultas básicas ao banco de dados.

Detalhes da Implementação: CrudInterface e CrudDAO

Interface Genérica: `CrudInterface<T>`

Define o contrato essencial para operações CRUD (Criar, Ler, Atualizar, Deletar) em qualquer tipo de entidade `<T>`, promovendo a padronização e desacoplamento do código.

```
interface CrudInterface<T> {
    void save(T u) throws SQLException;
    void update(int id) throws SQLException;
    List<T> get() throws SQLException;
    void delete(int id) throws SQLException;
}
```

Detalhes da Implementação: CrudInterface e CrudDAO

Para garantir flexibilidade e reusabilidade, o EasyPeasy utiliza uma arquitetura baseada em interfaces genéricas para operações de banco de dados.

Implementação Abstrata: `CrudDAO`

Fornece uma implementação genérica e reutilizável para a interface `CrudInterface`, lidando com a execução de comandos SQL e a gestão de chaves geradas automaticamente.

```
class CrudDAO implements CrudInterface<T> {
    public int save(String sql, Object... params) throws SQLException {
        int generatedId = -1;
        try (PreparedStatement psmt = conn.prepareStatement(sql, Statement.RETURN_GENERATED_KEYS)) {
            for (int i = 0; i < params.length; i++) {
                psmt.setObject(i + 1, params[i]);
            }
            psmt.executeUpdate();

            try (ResultSet rs = psmt.getGeneratedKeys()) {
                if (rs.next()) {
                    generatedId = rs.getInt(1);
                }
            }
        }
        return generatedId;
    }
    // ... outros métodos ...
}
```

Esta abordagem com o tipo genérico `<T>` permite que `CrudDAO` seja usado para diferentes modelos (Usuário, Item, etc.) sem duplicação de código.