**Controle Financeiro Pessoal**

**Muita atenção: Só quero uma instrução por vez**

Contexto, o projeto é um CRUD que visa integrar o Frontend (**HTML**, **CSS** e **JavaScript**) ao backend (**Java** puro e **PostgreSQL**).

Teremos uma tela de login.

Teremos uma segunda tela com as tabelas.

O usuário deverá ser capaz de cadastrar login e senha.

O usuário deverá ser capaz de alterar login de senha.

O usuário deverá ser capaz de cadastrar os valores e textos das tabelas.

O usuário deverá ser capaz de editar os valores e textos das tabelas.

O usuário deverá ser capaz de excluir os valores e textos das tabelas.

O usuário deverá ser capaz de ver o resultado final/parcial da diferença entre custos e renda.

Ao logar na aplicação o usuário deverá encontrar seu os dados das tabelas salvos com as últimas alterações feitas.

O usuário deverá ser capaz de gerar um relatório em PDF sempre que desejar.

O relatório deverá imprimir as tabelas e o resultado final/parcial.

A tabela **CUSTOS** terá 12 linhas e uma linha de **TOTAL** abaixo. O usuário deverá ser capaz de adicionar mais linhas se desejar abaixo da linha 12 e acima da linha **TOTAL.**

A tabela **RENDA** terá 4 linhas e uma linha de **TOTAL** abaixo. O usuário deverá ser capaz de adicionar mais linhas se desejar abaixo da linha 4 e acima da linha **TOTAL.**

**Estrutura básica do projeto**

**Backend (Java + JDBC):**

* Crie um projeto Java com estrutura MVC (Model-View-Controller).
* Use servlets para lidar com requisições HTTP.
* Use JDBC para conectar ao PostgreSQL e executar comandos SQL.
* Configure o web.xml para mapear seus servlets.
* Apache Tomcat

**Frontend (HTML/CSS/JS):**

* Crie páginas HTML com formulários para criar e editar dados.
* Use CSS para estilizar.
* Use JavaScript para melhorar a experiência do usuário (validações, AJAX, etc).

**Exemplo de fluxo CRUD**

1. **Create**: Formulário HTML envia dados via POST para um servlet.
2. **Read**: Servlet consulta o banco e retorna dados para exibir.
3. **Update**: Formulário com dados preenchidos envia via POST/PUT.
4. **Delete**: Botão que envia requisição para deletar um item.

### O que garantir no backend:

* Um servlet mapeado para /salvar-dados no web.xml.
* Esse servlet deve aceitar requisições POST.
* Ele deve ler o corpo da requisição como JSON e salvar os dados no PostgreSQL.