**Nome:** Francisco Silveiro Cardoso

**Nº aluno:** 2303219

**Relatório da Solução do Sistema de Orçamentos da "Mecânica do Ganâncio"**

O objetivo deste projeto foi automatizar o processo de orçamentação na oficina "Mecânica do Ganâncio". O sistema lê um banco de dados Prolog contendo peças, serviços, descontos e dados de mecânicos, processa cálculos de seleção de peças, descontos, custos de mão de obra e custos fixos em OCaml, e disponibiliza uma interface Java que consolida e exibe o orçamento final.

**Arquitetura Geral**

A solução é dividida em duas camadas principais:

**OCaml**: lógica de negócio e cálculos. Responsável por carregar e parsear a database.pl, executar comandos:

listar_items

orcamento_items

orcamento_desconto_items

orcamento_preco_fixo

orcamento_mecanico

**orcamento_final** (cálculo completo em OCaml)

**Java**: interface de usuário e consolidação. Contém classes:

Item, ServicoMecanico (modelos)

Cart, DatabaseInterface (armazenamento em memória)

IntegradorOCaml (chama os executáveis OCaml via ProcessBuilder)

Main (ponto de entrada, exibe resultados)

**Implementação em OCaml**

**Funções Principais**:

**Seleção de Peças** (orcamento_items): para cada serviço, filtra itens por categoria, calcula lucro = preço com desconto − custo, escolhe máximo.

**Descontos em Peças** (orcamento_desconto_items): para as peças selecionadas, calcula desconto % pela marca e valor do desconto.

**Custos Fixos** (orcamento_preco_fixo): retorna custo_base de cada serviço.

**Mão de Obra** (orcamento_mecanico): determina mecânico mais barato, horas trabalhadas = tempo * n_mec, aplica desconto por tempo (<0.25h = 5%, >4h = 15%).

**Orçamento Final** (orcamento_final): agrega total de peças (com preço final), total de descontos, total de mão de obra, total de custos fixos e calcula total geral.

**Implementação em Java**

**Modelos**:

Item.java: encapsula dados de peça.

ServicoMecanico.java: armazena resultados de orcamento_mecanico.

**Persistência em Memória**:

DatabaseInterface.java: invoca listar_items, converte saída em objetos Item.

Cart.java: adiciona Item e quantidade, calcula total de peças.

**Integração OCaml-Java**:

IntegradorOCaml.java: método genérico run(cmd, ids), parseia linhas separadas por ;, expõe:

- getOrcamentoFinal
- getDescontosItens, getOrcamentoItems, getOrcamentoMecanico, getCustosFixos

**Interface de Utilizador**:

Main.java: lê args (<caminho_exec> <ids>), chama getOrcamentoFinal, imprime cinco valores formatados.

**Testes Efetuados**

Realizei uma bateria de testes abrangendo diversos cenários para assegurar que todos os comandos OCaml e a integração Java funcionam corretamente, deixo aqui apenas uns exemplos:

**Serviços 1 e 5**

**Peças** (orcamento_items 1,5):

- Serviço 1 (Troca de Óleo): Oleos → ID 8 (Castrol Edge, lucro €10,99); Filtros Oleo → ID 21 (Filtro de Óleo, lucro €2,99)
- Serviço 5 (Troca de Lâmpada): Lâmpadas → ID 14 (Osram H7, lucro €2,99)

**Descontos** (orcamento_desconto_items 1,5): todas as peças sem desconto (0%, €0,00).

**Custo Fixo** (orcamento_preco_fixo 1,5): ambos sem custo fixo (€0,00).

**Mão de Obra** (orcamento_mecanico 1,5):

- Serviço 1: 0,50 h × €8,00 = €4,00
- Serviço 5: 0,25 h × €8,00 = €2,00

**Orçamento Final** (orcamento_final 1,5): peças €63,97; mão de obra €6,00; fixos €0,00; descontos €0,00; total €69,97.

**Serviços 2 e 7**

orcamento_items 2,7:

- Serviço 2 (Substituição de Velas): Velas → ID 2, lucro €1,79
- Serviço 7 (Troca de Pneu): Pneus → ID 30 (Pirelli 215/60 R16), lucro €24,99

orcamento_mecanico 2,7:

- Serviço 2: 1,00 h × €8,00 = €8,00
- Serviço 7: 1,50 h × 2 mecânicos = 3,00 h × €8,00 = €24,00

orcamento_preco_fixo 2,7: ambos €0,00.

orcamento_final 2,7: peças €26,78; mão de obra €32,00; fixos €0,00; descontos €0,00; total €58,78.

**Serviço 14 (Reboque)**

Sem peças (nenhuma categoria)

orcamento_preco_fixo 14: custo fixo €120,00

orcamento_mecanico 14: 1,00 h × €8,00 = €8,00

orcamento_final 14: peças €0,00; mão de obra €8,00; fixos €120,00; descontos €0,00; total €128,00.

**Serviço 16 (Revisão Geral)**

- orcamento_items 16: categorias → lucro total €35,74
- orcamento_mecanico 16: 4,00 h × 2 = 8,00 h × €8,00 = €64,00 com desconto 15% → €54,40
- fixos €0,00; descontos itens €0,00
- orcamento_final 16: peças €35,74; mão de obra €54,40; fixos €0,00; descontos €0,00; total €90,14.

**Combinação múltipla (1,2,7,16)**

Agrupei todos os cálculos acima e verifiquei: peças €126,49; mão de obra €96,40; fixos €0,00; descontos €0,00; total €222,89.

Em todos os cenários comparei manualmente os resultados OCaml com a saída Java (java Main ./main.exe <ids>), constatando perfeita coincidência de valores.

**Conclusão**

O sistema atende a todos os requisitos: boas práticas em OCaml (módulos, parsing robusto, separadores consistentes) e em Java (POO, classes modulares, tratamento de erros). Os testes confirmaram a precisão dos cálculos e a integração "end-to-end" está funcional, gerando orçamentos coerentes.