NOME: João Paul Fernandes Carvalho
N.º DE ESTUDANTE: 2304353
CURSO: Licenciatura de Engenharia Informática

Relatório:

O sistema criado para automatizar os orçamentos da "Mecânica do Ganâncio" une a força do OCaml para o processamento central e a flexibilidade do Java na camada de interface. Na fase inicial, fizemos a leitura do arquivo database.pl, que contém factos em Prolog para o inventário de peças e serviços. O método read_file lê todas as linhas do arquivo; cada linha é então submetida a expressões regulares nos comandos parse_item e parse_service, obtendo, respetivamente, os dados de cada item e de cada serviço. No exercício 1, também implementamos uma classificação lexical por categoria, marca e nome de peça, assegurando uma apresentação nítida do inventário no formato "ID; Nome;Marca; Tipo;Custo; Preço;Quantidade". No segundo exercício, a ênfase foi colocada na escolha das peças mais rentáveis para cada serviço solicitado. O programa, a partir da lista de serviços requisitados no comando orcamento_items, analisa as categorias de peças requeridas, seleciona todos os itens do inventário que se enquadram nessas categorias e, por meio da função profit_of_item, determina o lucro de cada candidato como (preço com desconto por marca) menos o custo. A peça que apresenta a maior margem é selecionada para cada categoria e impressa em formato ";" , com um totalizador de lucro adicional. No exercício três, abordamos o cálculo dos custos de mão de obra. Com base nos dados mecânico(ID, Nome, CustoHora) e desconto_mao_de_obra, agrupamos todos os mecânicos e as normas de desconto em listas particulares. A função labor_cost estabelece, para cada trabalho, a quantidade de horas requeridas, escolhe o mecânico com o menor custo-hora, determina o custo bruto e aplica o desconto apropriado (5 % para tarefas de curta duração, 15 % para tarefas de longa duração). O resultado final, que inclui a identificação do serviço, horas, custo por hora, valor bruto, descontos e total líquido, é apresentado numa forma legível. Por fim, no exercício 5, criamos a classe OficinaApp em Java, que serve como um wrapper. Ela recebe os comandos listar_itens, orcamento_itens e orcamento_mecanico do usuário, executa o executável OCaml

correspondente através do dune exec (ou simplesmente chamando o binário), analisa a saída pelo ProcessBuilder e transforma cada linha numa instância de Part ou LaborCost. Esta consolidação em Java possibilita ao usuário a visualização de resumos compreensíveis - listagem de estoque, lucro por peça em relação ao total agregado, e custos de mão de obra por serviço em relação ao total agregado - tudo isso sem revelar a complexidade interna do Ocaml. O resultado é um sistema modular, de fácil expansão, onde o OCaml se encarrega da parte computacional e o Java proporciona uma interface intuitiva e familiar ao usuário. Cada comando passou por testes rigorosos em linha de comando, confirmando a extração adequada dos dados, o cálculo meticuloso dos lucros e despesas, e a perfeita integração entre as duas linguagens.