E1 Sistema Comercial Sbornia

INTEGRANTES:
João Vitor Lichston Machado
Larissa Oliveira da Silva
Maria Eduarda Schüler

Porto Alegre, 2025

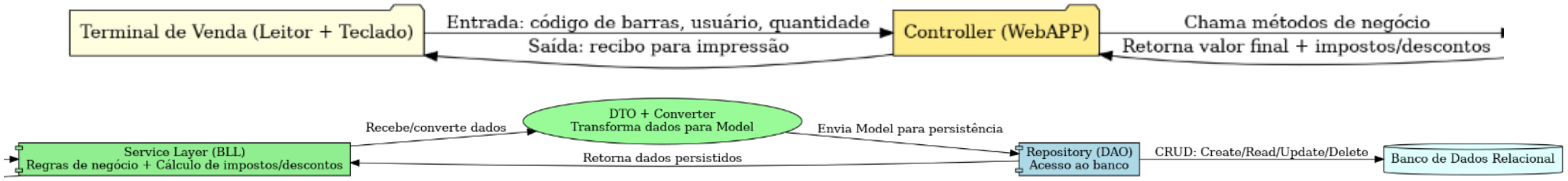
1. PROBLEMA

A Sbørnia, um país praticamente desconhecido pelas gerações atuais, possui um sistema comercial muito peculiar, conforme descrito a seguir. Todas as compras são realizadas em grandes superfícies comerciais. As compras podem ser realizadas presencialmente ou pela internet (fora do escopo deste exercício), somente por usuários previamente cadastrados. Você foi chamado para realizar o planejamento de um novo sistema de vendas para as lojas, considerando que:

a. As compras presenciais devem continuar utilizando os antigos terminais: um leitor de código de barras e um teclado para informações sobre o usuário e quantidades, todas as informações resultantes são impressas em papel, pois não há monitor.

b. Um sistema de controle de estoque mantém o cadastro de produtos. Sobre cada produto armazena-se o código do produto, a descrição, a quantidade em estoque, o preço unitário e a sua categoria. Outro sistema mantém o cadastro de usuários, sobre cada usuário é armazenado, entre outras informações, seu identificador único, nome, data de nascimento e número de dependentes. Todas as informações são persistidas em um banco de dados relacional.

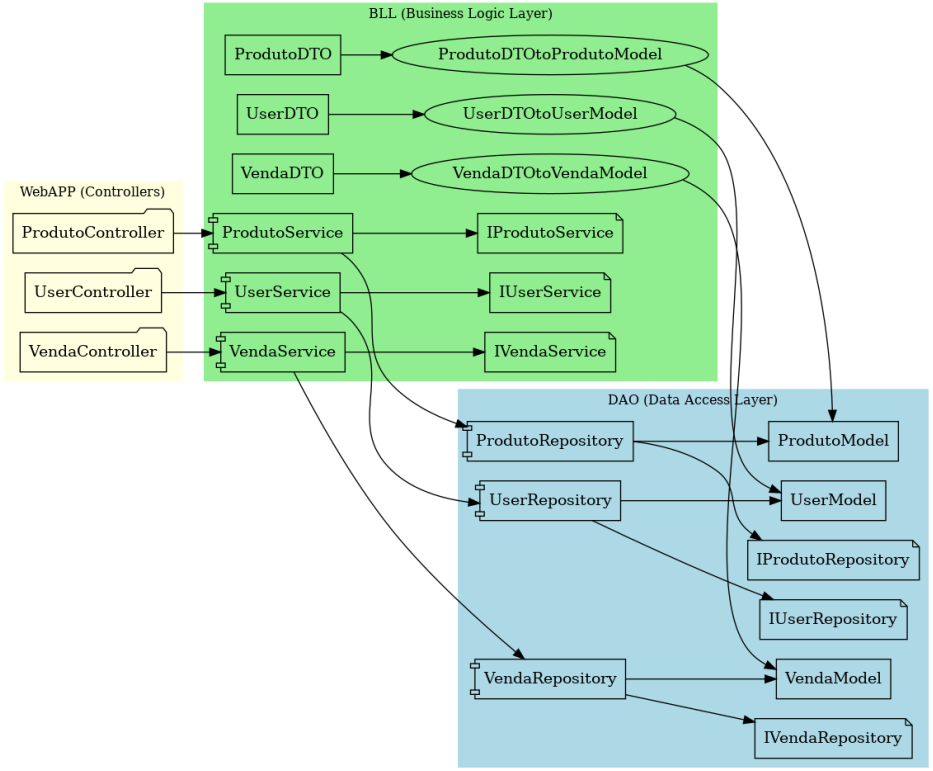
c. Ao realizar uma venda é feito o cálculo do valor final, que é dado pelo produto da quantidade vendida e o preço unitário, somado impostos. Os impostos são calculados a partir das seguintes regras: i. Sobre produtos alimentícios incide um imposto único de 5%. ii. Sobre produtos automotivos incide imposto de 30%. iii. Sobre bebidas alcoólicas incide imposto de 100%. iv. Para outras categorias incide o imposto base de 17%. É importante salientar que novas faixas de impostos podem ser adicionadas sempre que necessário. v. Usuários com mais de 60 anos não pagam imposto, exceto para bebidas alcoólicas. vi. Usuários com mais de três dependentes tem um desconto de 50% sobre o valor de imposto calculado, exceto para bebidas alcoólica.



2. MODELAGEM DO PROBLEMA

No sistema de vendas da Sbørnia, essa arquitetura permite lidar com as regras complexas de impostos e descontos sem misturar código de interface ou de persistência.

- ***DAO/Repository*** gerencia produtos, usuários e vendas no banco de dados relacional, mantendo a estrutura pronta para futuras expansões de categorias e taxas de imposto.
- ***BLL/Service*** aplica as regras de negócio, como isenção de imposto para idosos e descontos para usuários com dependentes, garantindo que a lógica possa ser alterada sem impacto na camada de dados ou na interface.
- ***DTOs e Converters*** asseguram que as informações trafeguem de forma segura e padronizada entre as camadas.
- ***Controllers*** simulam o papel dos terminais antigos, recebendo entradas do leitor de código de barras e teclado, processando via Services e devolvendo o resultado para impressão em papel.



Assim, o sistema fica preparado para evolução, manutenção simplificada e integração com futuras interfaces, como vendas online ou novos dispositivos de ponto de venda.

3. SOLUÇÃO

3.1. DAO (Data Access Object)

- Onde: Pacote DAO.Repository e DAO.Repository.Contracts.
- O que faz: Isola a lógica de acesso ao banco de dados em classes específicas (Repositories).
- Benefício: Facilita a manutenção e troca do mecanismo de persistência sem afetar a camada de negócio.
- Exemplo no código: ProdutoRepository, UserRepository, VendaRepository implementando IProdutoRepository, IUserRepository, IVendaRepository.

3.2. DTO (Data Transfer Object)

- Onde: Pacote BLL.DTO.
- O que faz: Transporta dados entre camadas sem expor diretamente as entidades do banco (Models).
- Benefício: Reduz acoplamento entre a camada de apresentação e a de persistência, aumentando segurança e flexibilidade.
- Exemplo no código: ProdutoDTO, UserDTO, VendaDTO.

3.3. Converter (Assembler/Mapper)

- Onde: Pacote BLL.Converter.
- O que faz: Converte objetos DTO em Model e vice-versa.
- Benefício: Centraliza a lógica de conversão, evitando repetição de código.
- Exemplo no código: ProdutoDTOtoProdutoModel, UserDTOtoUserModel, VendaDTOtoVendaModel.

3.4. Service Layer

- Onde: Pacote BLL.Service e BLL.Service.Contracts.
- O que faz: Encapsula as regras de negócio e orquestra chamadas aos repositórios.
- Benefício: Mantém a lógica de negócio isolada da interface de usuário e da persistência, facilitando testes unitários.
- Exemplo no código: ProdutoService, UserService, VendaService implementando IProdutoService, IUserService, IVendaService.

3.5. Controller (MVC Pattern - parte do Controller)

- Onde: Pacote WebAPP.Controller.
- O que faz: Recebe as requisições, valida dados e chama a camada de negócio (Services).
- Benefício: Segrega responsabilidades, mantendo a camada de apresentação separada da lógica de negócio.
- Exemplo no código: ProdutoController, UserController, VendaController.

3.6. Repository Pattern

- Onde: Pacote DAO.Repository.
- O que faz: Fornece uma abstração para a camada de acesso a dados, encapsulando consultas e operações de banco.
- Benefício: Facilita a troca de tecnologia de persistência e permite a criação de implementações falsas (mocks) para testes.
- Exemplo no código: ProdutoRepository, UserRepository, VendaRepository.

4. CONSIDERAÇÕES FINAIS

A solução está organizada seguindo o princípio de separação de responsabilidades e arquitetura em 3 camadas:

- WebAPP (Controller) → recebe entrada do usuário e delega para o Service.
- BLL (Service + DTO + Converter) → contém as regras de negócio e transformação de dados.
- DAO (Repository + Model) → realiza a persistência e consulta dos dados.
- Essa abordagem garante baixo acoplamento, alta coesão e facilidade de manutenção.