Trabalho de Projeto Detalhado de Software (PDS) - Entrega

Documento de Especificação de Software - NxT

UFC

Documento de Requisitos, Regras de Negócio e Modelagem de Negócio

Versão 0.2

Universidade Federal do Ceará – Campus Russas

Template reutilizado de: **Documento de Requisitos TempCorporation - EOSystem**

Sumário

Informações do Documento de Requisitos

Título do documento	Documento de Especificação de Software - NxT
Subtítulo do documento	Sistema de Gerenciamento de Vendas e Departamentos
Repositório do sistema	https://github.com/csvitor-dev/NxT
Equipe	Avallos Marinho de Oliveira (amo) Francisco Guilherme Ferreira Dias Martins (fgm) João Augusto Silva Ferreira (jaf) Victor Ravel Santos Cavalcante (vrc) Vitor Costa de Sousa (vcs)
Orientador	Profa. D.ra. Jacilane de Holanda Rabelo
Disciplina	Projeto Detalhado de Software
Versionamento	v0.1 – *31/10/24* v0.2 – *02/12/24*
Comentários	Muito do que se encontra na especificação deste documento não reflete a situação atual do sistema

HISTÓRICO DE REVISÕES

Revisão	**Data**	**Descrição**	**Autor**
01	11/11/24	Estruturando os tópicos iniciais do documento	amo, fgm, jaf, vrc, vcs
02	15/11/24	Separando responsabilidades para preenchimento dos tópicos	amo, fgm, vrc, vcs
03	16/11/24	Preenchimento dos tópicos da UML (6 – 8)	vcs
04	19/11/24	Definição dos requisitos funcionais e não-funcionais	amo, vcs
05	20/11/24	Conclusão do tópico 3: *Modelo de Processo para Desenvolvimento*	amo, fgm, jaf, vrc, vcs
06	22/11/24	Conclusão do documento preliminar para primeira entrega	amo, fgm, jaf, vrc, vcs

07	04/12/24	Corrigindo justificação do texto e tópicos "órfãos"	vcs
08	05/12/24	Acréscimo do requisito "Registrar produtos" e a propagação de tal mudança no restante do documento	amo, vcs
09	06/12/24	Correções adicionais aos tópicos levantados no feedback da primeira entrega	fgm, vrc, vcs
10	20/12/24	Acréscimo do tópico de referências	vcs
11	24/12/24	Decisão e divisão de responsabilidades para as partes de *Reutilização de Software* e *Arquitetura de Software*	amo, fgm, vrc, vcs
12	26/12/24	Finalização do documento de revisões (*patch*) da primeira entrega	vcs
13	07/01/24	Finalização do tópico 9 (*Reutilização de Software*)	amo, fgm, vrc
14	09/01/24	Finalização do tópico 10 (*Arquitetura de Software*)	vcs

Documento de Requisitos

1. Introdução ao Projeto

Empresas que operam com vendas, especialmente do setor de varejo, possuem dificuldades inerentes no que tange ao monitoramento de estoque, previsão de demandas, registro de vendas, gestão e atendimento de clientes, elaboração de relatórios e métricas.

Frente às necessidades, muitas soluções existem no mercado com o propósito de sanar tais problemáticas, em diferentes níveis de automatização e processos, com diferentes escopos.

Com isso, nosso projeto busca organizar e automatizar as tarefas relacionadas ao gerenciamento de vendas por departamento, fornecendo um *dashboard* com um panorama de departamentos, vendedores e vendas. Portanto, espera-se que o sistema visa sanar o registro de vendas e a construção de relatórios e métricas que norteiam as metas de vendas.

1.1 – Tema

O projeto fundamenta-se em um sistema de gerenciamento de vendas (NxT).

Para fins ilustrativos, a equipe decidiu contextualizar o uso e o desenvolvimento desta aplicação como um sistema interno de uma empresa fictícia, chamada **NextTrend**, especializada no setor de varejo como uma **loja de departamentos**.

1.2 – Objetivos

- Simplificar e automatizar as atividades ligadas ao controle de vendas dos diferentes departamentos de uma empresa.
- Oferecer um *dashboard* que apresenta uma visão geral de departamentos, vendedores e transações realizadas.
- Gerar relatórios e métricas sobre vendas, desempenho de vendedores e movimentação de departamentos.

1.3 – Delimitação do problema

Reconhecendo a complexidade e a abrangência das soluções possíveis para esse mercado, buscou-se definir o escopo do projeto em termos do registro de vendas e o gerenciamento de vendedores e departamentos, além da possibilidade da geração de relatórios e métricas de vendas.

De tal modo, não considera-se a camada de atendimento ao cliente e estratégias de vendas ou o monitoramento de estoque. O que fora citado compõem o escopo negativo do projeto, ou seja, o compromisso da equipe e a definição do projeto em não se ater a essas funcionalidades na entrega definida.

Apesar disso, um conceito nuclear ao projeto é o **Produto**, já que **departamentos** da loja contém produtos e os **registros de venda** são sobre produtos, implicando que o sistema necessita de integração com dados de estoque – mesmo que a camada de estoque não faça parte do escopo.

Portanto, a proposta é o sistema NxT consumir somente informações necessárias dos produtos que fazem parte do estoque.

1.4 – Justificativa de escolha do tema

O primeiro motivo para a escolha de tal sistema foi o fato de que ele já admite muitas funcionalidades já praticamente completas, o que possibilita à equipe construir uma documentação ampla, estender novas funcionalidades e aplicar boas práticas de projetos.

Já em termos teóricos, existe a vantagem do sistema já ser implementado em **MVC***, um padrão arquitetural, fazendo com que cada membro da equipe aprenda mais sobre este ou qualquer outra arquitetura que venha a ser uma possibilidade de implementação.

Por fim, fica claro que o sistema fornece vantagem em duas vias: possibilitar engenharia reversa para documentação do sistema e está aberto para criação e extensão de funcionalidades, padrões de projeto e arquitetura.

1.5 – Organização da especificação

A seguir são apresentadas as divisões deste documento e uma rápida descrição de cada seção:

- Seção 1 – **Introdução ao Projeto**: especifica o que o nosso projeto faz, os problemas que ele soluciona e o por que ele foi escolhido;
- Seção 2 – **Descrição Geral do Sistema**: descreve o escopo do sistema, suas funcionalidades, objetivos, e os problemas a serem resolvidos de maneira geral; além disso, apresenta as regras de negócio que detalham o comportamento e as restrições de negócio que os requisitos funcionais devem obedecer e, desta forma, garantir a conformidade do sistema com o modelo de negócio;
- Seção 3 – **Modelo de Processo para Desenvolvimento**: fornece, em alto nível, uma descrição das atividades de Engenharia de Software que orientam o projeto em termos de Processo de Software;
- Seção 4 – **Requisitos Funcionais**: especifica os requisitos funcionais planejados para o sistema;
- Seção 5 – **Requisitos Não-Funcionais**: especifica os requisitos não-funcionais planejados para o sistema;
- Seção 6 – **Documentação de casos de uso**: apresenta uma visão geral dos atores e das interações principais entre eles e o sistema;
- Seção 7 – **Diagrama de classes**: apresenta o Diagrama de Classes, retratando os conceitos essenciais do domínio;
- Seção 8 – **Diagrama de sequência**: apresenta o Diagrama de Sequência, ilustrando o fluxo de interações entre objetos para realizar uma funcionalidade ou caso de uso;
- Seção 9 – **Reutilização de Software**: detalhamento e descrição dos em termos do reuso de software e da linha de produto (mapa de produto e o *Feature Model* – modelo **FODA***);
- Seção 10 – **Arquitetura de Software**: trata da estrutura em alto nível do sistema,

descrevendo componentes de software e suas interações, além do levantamento das arquiteturas candidatas e a decisão de qual (ou quais) utilizar – o que reflete os estilos de arquitetura;

- Seção 11 – **Conclusões**: apresenta um resumo das principais lições obtidas ao longo do desenvolvimento do trabalho;
- Seção 12 – **Ata de Reuniões:** contém as datas e descrição do que foi discutido e realizado em relação ao projeto, tanto na parte de requisitos quanto em modelagem;
- Seção 13 – **Glossário**: contém os significados de termos técnicos ou de domínio presentes neste documento em que, quando for necessário fazer referência a eles, cada termo estará acompanhado do caractere '*', indicando que ele consta no glossário;
- Seção 14 – **Referências**: compila os conteúdos pesquisados e utilizados como base para o presente documento.

2. Descrição Geral do Sistema

O NxT é uma aplicação interna da loja de departamentos **NextTrend**, com a responsabilidade de gerenciar e monitorar indivíduos, vendas e métricas que cobrem o escopo. O objetivo do sistema é fornecer uma plataforma na qual será responsável por gerenciar as vendas, vendedores e departamentos da empresa em questão, além do monitoramento de produtos transacionados, tendo como funções principais a criação dos departamentos e vendedores, o monitoramento e análise das vendas, vendedores e departamentos.

2.1 – Principais Envolvidos e suas Características

2.1.1 – Usuários do Sistema

A aplicação é projetada para ser utilizada por três tipos de usuários:

- **Administrador**: pode ser visto como um superusuário, com responsabilidade e permissão para cadastrar ou remover vendedores, gerentes e departamentos.
- **Gerente de vendas**: possui uma visão geral sobre departamentos, vendedores, e vendas. Além disso, possui acesso ao *dashboard* com métricas e relatórios de vendas.
- **Vendedor**: O vendedor tem o papel de registrar e gerenciar suas vendas, estando restrito a isso – sem poder atualizá-las. Seu acesso é limitado apenas às vendas realizadas por ele.

2.2 – Regras de Negócio

Esta seção apresenta as regras de negócio que devem ser obedecidas pelo sistema.

2.2.1 – Identificação das regras de negócio

Quanto ao detalhamento das regras, seguirá uma notação semelhante aos requisitos: [*RN-Número*] *Nome*

Onde *RN* é a abreviação para *Regra de Negócio*, seguida do campo *Número* que é a numeração ordenada crescente, análogo aos requisitos, sem repetição.

Ademais, o campo *Nome* é análogo ao mesmo campo nos requisitos.

2.2.2 – Levantamento das Regras de Negócio

- **[RN001]** ***Hierarquia de Acesso***
    1. O **administrador** tem acesso completo sobre o gerenciamento de vendedores, gerentes e departamentos.
    2. O **gerente de vendas** tem acesso a visualização de departamentos e vendedores cadastrados, como também ao dashboard.
    3. O **vendedor** tem acesso ao registro e consulta de suas próprias vendas.
- **[RN002]** ***Sincronização de Produtos***
    1. O sistema atualiza dados sobre os produtos (nome, descrição, preço e departamento) advindos do estoque;
    2. Somente o **vendedor** tem acesso aos produtos para que possa relacioná-los a uma venda realizada;
    3. Dentre as métricas e relatórios, informações pertinentes sobre os produtos (como os produtos mais vendidos) são apresentadas ao **gerente de vendas**.
- **[RN003]** ***Gerenciamento de Vendedores***
    1. Somente o **administrador** terá acesso ao gerenciamento de vendedores.
    2. Um vendedor não pode ser excluído se houver vendas associadas a ele.
- **[RN004]** ***Gerenciamento de Departamentos***
    1. Somente o **administrador** terá acesso ao gerenciamento de departamentos.

2. Departamentos organizam vendedores e vendas.
3. Um departamento não pode ser excluído se houver vendedores associados a ele.

➢ **[RN005]** ***Registro de Vendas***

1. Somente o vendedor terá acesso ao cadastro de vendas.
2. A atualização de uma venda está restrita ao seu status.
3. Os status de venda podem ser: *Pendente*, *Cancelada* ou *Concluída*.
4. Apenas vendas no status *Pendente* podem ser atualizadas.
5. A remoção de uma venda é estritamente vedada, para fins de histórico.

➢ **[RN006]** ***Gerenciamento de Credenciais***

1. Para cada novo usuário, será gerada uma credencial de acesso (chamada de *NiD**) fornecido pelo administrador.
2. A credencial de acesso é constituída por 10 caracteres; 6 algarismos hexadecimais seguidos de 3 algarismos decimais identificadores, separados por um hífen (*XXXXXX-YYY*).
3. A validação do *NiD* se encontra em Glossário.
4. Para efetuar o login, o usuário deve fornecer seu email cadastrado e sua credencial.

3. Modelo de Processo para Desenvolvimento

Este tópico tem por finalidade apresentar a descrição do modelo de processo adotado no projeto.

3.1 – Técnicas Escolhidas/Ciclo de Vida/Divisão de Equipe

As técnicas utilizadas foram a Engenharia Reversa (ou seja, a partir de um sistema existente, construir sua documentação e modelagem) e *Brainstorming*, com o objetivo de capturar soluções possíveis para cada problema levantado.

Em resposta a isso, a equipe chegou à conclusão de utilizar uma adequação da metodologia ágil XP (***Extreme Programming***), pois se encaixa às necessidades do projeto, que são elas:

- **Entrega Rápida de Funcionalidades**

  Dado que a proposta do XP é a realização de iterações curtas, o nosso objetivo é entregar funcionalidades validadas em cada iteração.

- **Qualidade do Código**

  O projeto exige alta confiabilidade e manutenção no código para evitar falhas em operações críticas, como o registro de vendas. O XP, como parte da metodologia, pressupõe o uso do TDD (Test Driven-Development), porém, não iremos adotá-la integralmente. Faremos o uso de uma camada de testes, mas não seguiremos o escopo **Red, Green** e **Refactor** – ou seja, teste antes do código.

- **Refatoração e Integração Contínua**

  A situação atual do projeto se encontra com alto acoplamento – todo o sistema definido em um único componente de software. Então, a proposta é refatorar para módulos interconectados, o que demanda integração frequente. A prática de integração contínua do XP ajuda a reduzir conflitos e garante que o sistema funcione de forma coesa. E, como complemento, a refatoração permite trazer limpeza e organização para as partes do software.

- **Colaboração da Equipe**

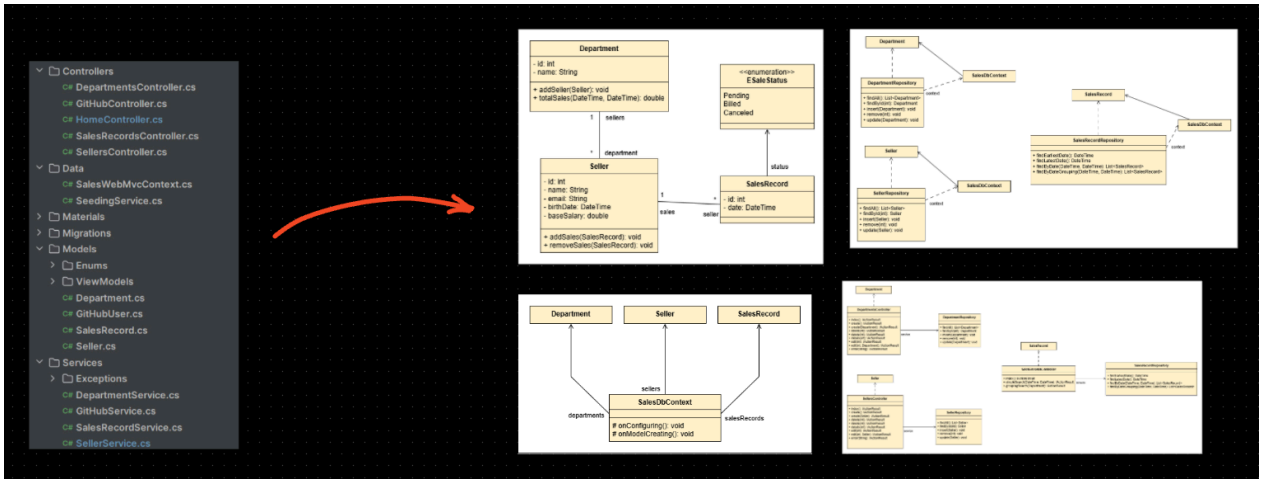
  A equipe precisa de um ambiente de trabalho colaborativo para alinhar desenvolvedores e stakeholders. O XP fomenta a comunicação constante e a programação em par, o que promove sinergia e compartilhamento de conhecimento.

Dado essa escolha, a divisão de atribuições por membro da equipe ficou da seguinte maneira (mesmo que, naturalmente, o XP não admita funções bem definidas, elas podem ser rearranjadas de outras metodologias ou conforme a necessidade da equipe ou projeto):

- **Avallos Marinho**: análise e *tracker*, sendo responsável por elicitar os requisitos e histórias de usuário e garantir consistência das necessidades do cliente, como também monitorar entregas.
- **Francisco Guilherme**: desenvolvimento, envolvido na implementação e integração da solução.
- **João Augusto**: testes, encarregado da camada de testes já citada anteriormente, validando as camadas implementadas.
- **Victor Ravel**: desenvolvimento, envolvido na implementação e integração da solução.
- **Vitor Costa**: *onsite leader*, representando tanto as necessidades do cliente, como liderando o ciclo de vida do projeto.

3.2 – Aplicação das Técnicas

Como primeira característica da implementação destas técnicas foi o fluxo da Engenharia Reversa aplicada para compor a especificação preliminar. Já que a base de código já era existente, construímos a especificação a partir dela e, depois, fizemos as devidas adaptações para que ele venha a refletir a arquitetura esperada.



Outro ponto importante foi a divisão de tarefas nessa etapa preliminar, onde nem estávamos preocupados com o código, mas sim a possibilidade de horizontalizar conhecimento de todo ciclo do processo para cada integrante da equipe.

Sumário

Vitor Costa De Sousa
10:47 18 de nov.
ATUALIZAR SOMENTE APÓS AS ATRIBUIÇÕES FOREM CONCLUÍDAS

Vitor Costa De Sousa
10:09 15 de nov.
@victorravel@alu.ufc.br
Atribuído a victorravel@alu.ufc.br

Vitor Costa De Sousa
10:09 15 de nov.
@guilhermedias1006@alu.ufc.br
Atribuído a guilhermedias1006@alu.ufc.br

Vitor Costa De Sousa
10:08 15 de nov.
@avallos_mo@hotmail.com
Atribuído a avallos_mo@hotmail.com

Atribuído a você

Vitor Costa De Sousa
10:08 15 de nov.
@vitorcostaworks2022@gmail.com
@jaugustosf@alu.ufc.br
* Utilizar Draw.io

Vitor Costa De Sousa
10:12 15 de nov.
@jaugustosf@alu.ufc.br
Atribuído a jaugustosf@alu.ufc.br

3.3 – Resultados Obtidos

Para esta primeira entrega, a equipe atingiu os seguintes resultados:

- **Levantamento de Requisitos**:
  - Identificação das principais funcionalidades esperadas pelo sistema, como registro de vendas, geração de relatórios, gestão de vendedores e gestão de departamentos.
- **Modelagem Inicial**:
  - Criação de diagramas preliminares (como casos de uso, classes e sequência) que representam as interações e funcionalidades principais do sistema.
- **Planejamento do Ciclo de Vida**:
  - Definição da metodologia ágil Extreme Programming (XP) como abordagem de desenvolvimento, devido à necessidade de flexibilidade, entregas incrementais e qualidade do código.
- **Definição da Arquitetura**:
  - Conforme já dito anteriormente, a base de código já existente segue o padrão MVC.
  - Tem-se como perspectiva atual reduzir o acoplamento da aplicação para que ela responda satisfatoriamente à manutenção e organização.
- **Especificação de Funcionalidades Prioritárias**:
  - Descrição detalhada de funcionalidades críticas para a próxima etapa, como:
    - Registro de uma venda.
    - Emissão de relatórios básicos de vendas.
- **Feedback Inicial**:
  - Validação do escopo preliminar e identificando ajustes necessários para iterações futuras.

4. Requisitos Funcionais

Esta seção apresenta em detalhes os requisitos funcionais do sistema.

4.1 – Identificação dos Requisitos

Para a especificação dos requisitos (válido para os funcionais e os não-funcionais) utilizaremos a seguinte representação: [*TipoDoRequisito-Número*] *Nome*

O campo *TipoDoRequisito* poderá ser especificado pelos códigos RF (Requisitos Funcionais) ou RNF (Requisitos Não-Funcionais). Já o campo *Número* será preenchido com um número correspondente à ordem em que os requisitos aparecem no documento (mas não será separado por hífen, é apenas notação).

Já o campo *Nome* seria como o título do requisito (uma breve descrição do que é o requisito com palavras-chave) acompanhado dos campos supracitados, desde que estes mesmos campos sejam únicos, tal como identificadores.

4.2 – Prioridade dos Requisitos

A cada requisito será atribuída uma prioridade. A descrição de cada uma segue abaixo:

- **Essencial** é um requisito imprescindível. Sem ele, o sistema não funcionará.
- **Importante** é um requisito que deve ser implementado, mas, se não for, o sistema funcionará do mesmo jeito, mas de maneira insatisfatória.
- **Desejável** é um requisito que trará um diferencial adicional ao sistema. Por isso, pode ser deixado para ser implementado por último ou em próximas iterações.

4.3 – Levantamento dos Requisitos Funcionais

➢ **RF001 - Sincronização de produtos**

O sistema deve recuperar informações sincronizadas de produtos que compõem o estoque e suas informações (nome, descrição, preço e departamento).

Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

Regras de Negócio: RN002

➢ **RF002 - Registro de vendas**

O sistema deve permitir que usuário vendedor registre suas vendas, vinculadas com informações (produto e departamento).

Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

Regras de Negócio: RN005

➢ **RF003 - Gerenciamento de vendedores**

O sistema deve garantir que somente o administrador tem o direito de cadastrar, excluir e editar vendedores e a associação a um determinado departamento.

Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

Regras de Negócio: RN001, RN003

➢ **RF005 - Gerenciamento de departamento**

O sistema deve permitir o cadastro, exclusão, edição e exibição de departamentos, incluindo informações como nome do departamento, código identificador e descrição..

Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

Regras de Negócio: RN001, RN004

➢ **RF006 - Login de usuário**

O sistema deve permitir que o usuário faça login através do email e da credencial.

Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

Regras de Negócio: RN001, RN006

➢ **RF007 - Acesso ao dashboard**

O sistema deve ter um *dashboard* para o usuário (gerente), que apresente de forma interativa os dados (métricas e desempenhos) de vendas da empresa.

Prioridade: ☐ Essencial ☑ Importante ☐ Desejável

Regras de Negócio: RN001

➢ **RF008 - Permissões do Gerente**

O sistema irá permitir que o usuário gerente tenha acesso aos departamentos e vendedores cadastrados e ao dashboard com métricas e relatórios de vendas.

Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

Regras de Negócio: RN001

➢ **RF009 - Permissões do vendedor**

O sistema irá permitir que o usuário vendedor tenha acesso apenas às funcionalidades relacionadas ao registro e consulta de vendas.

Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

Regras de Negócio: RN001, RN003

➢ **RF010 - Permissões do Administrador**

O sistema irá permitir que o administrador tenha responsabilidade completa sobre o gerenciamento de vendedores, gerentes, departamentos e relatórios.

Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

Regras de Negócio: RN001, RN003, RN004, RN006

➢ **RF011 - Geração de relatórios**

Através do *dashboard*, o sistema deve fornecer ao usuário gerente a geração de relatórios de vendas e métricas atreladas ao vendedor, total de vendas e status.

Prioridade: ☐ Essencial ☐ Importante ☑ Desejável

Regras de Negócio: RN001

➢ **RF012 - Métricas de departamento**

O sistema deve gerar métricas específicas de vendas para cada departamento, incluindo o total de vendas e comparativos entre períodos.

Prioridade: ☐ Essencial ☐ Importante ☑ Desejável

Regras de Negócio: RN001

5. Requisitos Não-Funcionais

Esta seção apresenta o que o sistema deve atender em termos de uso de dados, desempenho, usabilidade, hardware e software utilizados, modelagem e segurança.

5.1 – Levantamento dos Requisitos Não-Funcionais

- **RNF001 - Ambiente de execução**

  O sistema é uma aplicação web, mas está restrito a telas maiores do que 14".

  Regras de Negócio: n/a

- **RNF002 - Persistência de dados**

  O sistema deve utilizar um banco de dados relacional para armazenamento, consulta, manipulação e definição de dados.

  Regras de Negócio: n/a

- **RNF003 - Segurança nos dados**

  Todos os dados sensíveis devem ser armazenados com criptografia SHA256.

  Regras de Negócio: n/a

- **RNF004 - Autenticação de usuário**

  Conforme a hierarquia de acessos, as rotas devem ser autenticadas utilizando tokens JWT Bearer.

  Regras de Negócio: RN001

- **RNF005 - Desempenho do sistema**

  O sistema deve renderizar páginas estáticas em, no máximo, 500ms e, naquelas em que houver consultas ao banco ou API, 5000ms.

  Regras de Negócio: n/a

6. Documentação de Casos de Uso

A documentação dos casos de uso tem como propósito rastrear as principais funcionalidades que o sistema deve atender, acompanhando as necessidades de negócio. Com o **Diagrama de Casos de Uso de Sistema**, busca-se fornecer um panorama do escopo e das interações entre os atores (logo, os usuários) e o sistema. Tal documentação contribui no levantamento dos requisitos e no monitoramento de suas mudanças, na construção de casos de teste e servir de linguagem não ambígua para toda equipe.

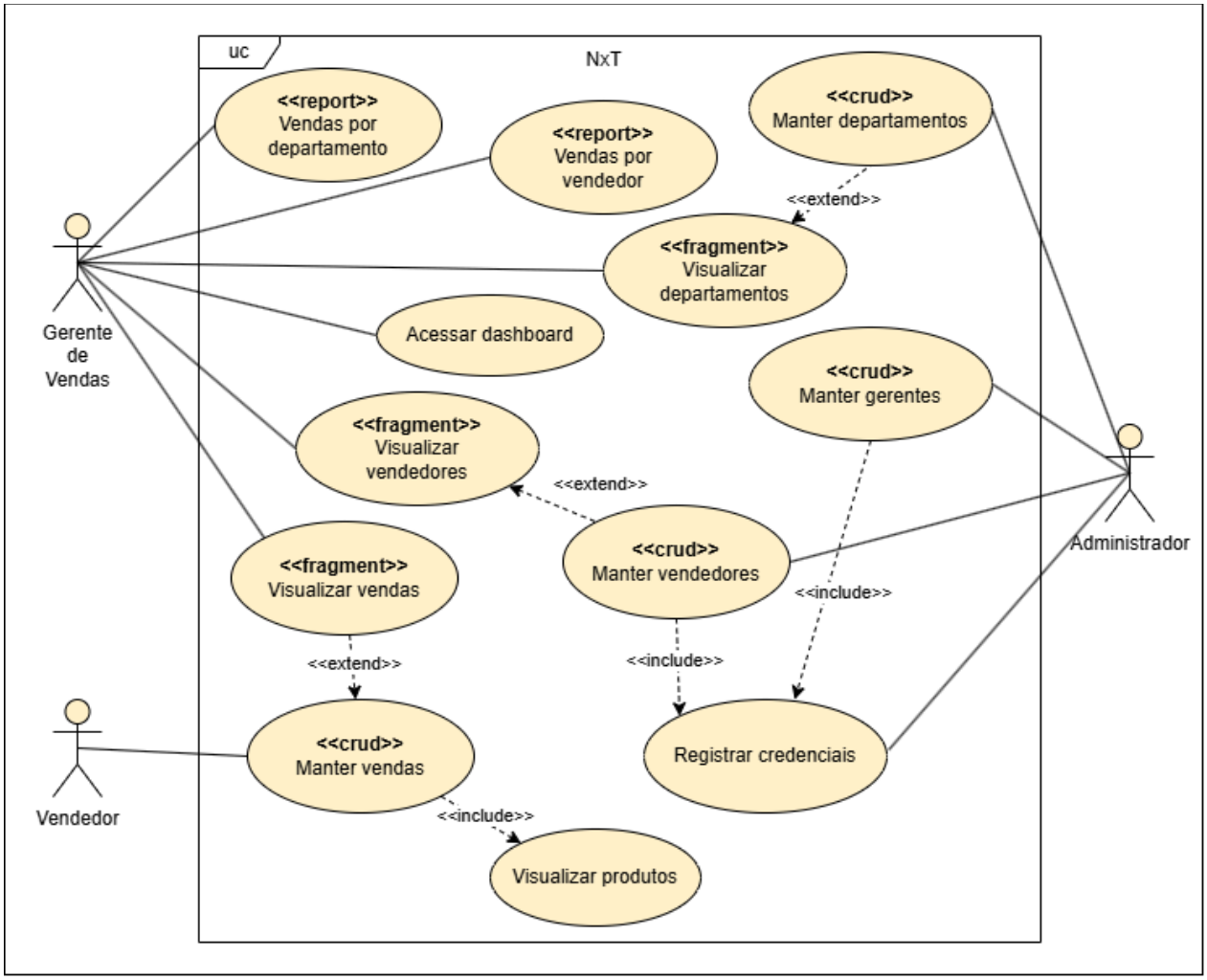
6.1 – Atores

Dado o escopo do sistema, a perspectiva preliminar dos atores se deu em três:

- **Administrador**: gerencia configurações gerais do sistema e realiza manutenção nos dados de alto nível (departamentos, permissões de acesso, relatórios);
- **Gerente de Vendas**: supervisiona o desempenho dos vendedores e dos departamentos, além de acessar relatórios detalhados;
- **Vendedor**: com visibilidade reduzida, realiza operações diretamente relacionadas às vendas e consulta seus próprios resultados.

6.2 – Diagrama de Casos de Uso

Conforme descrito anteriormente sobre a documentação dos casos de uso, assim como a descrição dos atores, tem-se a definição do **Diagrama de Casos de Uso de Sistema** para a perspectiva preliminar:



6.3 – Descrição dos Casos de Uso

- **UC001 – <<report>> *Vendas por departamento***
  - **Atores**: Gerente de Vendas.
  - **Descrição**: O gerente de vendas acessa a página de *dashboard* e seleciona, como filtro, datas mínimas e máximas; esse filtro vai agrupar as vendas que pertencem a cada departamento existente e que estejam no intervalo de tempo.
- **UC002 – <<report>> *Vendas por vendedor***
  - **Atores**: Gerente de Vendas.
  - **Descrição**: O gerente de vendas acessa a página de *dashboard* e seleciona, como filtro, algum vendedor; esse filtro vai agrupar as vendas que pertencem a um determinado vendedor.
- **UC003 – *Acessar dashboard***
  - **Atores**: Gerente de Vendas.
  - **Descrição**: O gerente pode acessar, conforme seu nível de acesso, a página de *dashboard* que possui diversas informações e métricas de valor para o negócio – apuração de vendas, desempenho de departamentos e vendedores –, com a possibilidade de filtros.
- **UC004 – <<fragment>> *Visualizar departamentos***
  - **Atores**: Administrador e Gerente de Vendas.
  - **Descrição**: O usuário pode acessar, conforme seu nível de acesso, a visualização de todos os departamentos cadastrados, admitindo detalhes de quantos vendedores associados e o total apurado no último período. Para o gerente, é uma interface somente leitura, enquanto que o administrador pode realizar outras operações.
- **UC005 – <<fragment>> *Visualizar vendedores***
  - **Atores**: Administrador e Gerente de Vendas.
  - **Descrição**: O usuário pode acessar, conforme seu nível de acesso, a visualização de todos os vendedores cadastrados, admitindo metadados sobre aquele

vendedor (nome, salário base, email e departamento). Para o gerente, é uma interface somente leitura, enquanto que o administrador pode realizar outras operações.

- **UC006 – <<fragment>> *Visualizar vendas***
  - **Atores**: Gerente de Vendas e Vendedor.
  - **Descrição**: O usuário pode acessar, conforme seu nível de acesso, a visualização de todos as vendas cadastrados, admitindo metadados sobre aquela venda (valor, vendedor, departamento). Para o gerente, é uma interface somente leitura, enquanto que o vendedor pode realizar outras operações.
- **UC007 – <<crud>> *Manter vendas***
  - **Atores**: Vendedor.
  - **Descrição**: O vendedor pode, conforme seu nível de acesso, registrar vendas e atualizar seu status.
- **UC008 – Visualizar produtos**
  - **Atores**: Vendedor.
  - **Descrição**: O vendedor, no momento de registrar suas vendas, terá acesso ao catálogo de produtos existentes, para que possa associar quais produtos foram vendidos.
- **UC009 – <<crud>> *Manter departamentos***
  - **Atores**: Administrador.
  - **Descrição**: O administrador pode efetuar a consulta, adição, remoção ou edição de um departamento.
- **UC010 – <<crud>> *Manter gerentes***
  - **Atores**: Administrador.
  - **Descrição**: O administrador pode efetuar a consulta, adição, remoção ou edição de um gerente.
- **UC011 – <<crud>> *Manter vendedores***
  - **Atores**: Administrador.

- **Descrição**: O administrador pode efetuar a consulta, adição, remoção ou edição de um vendedor.

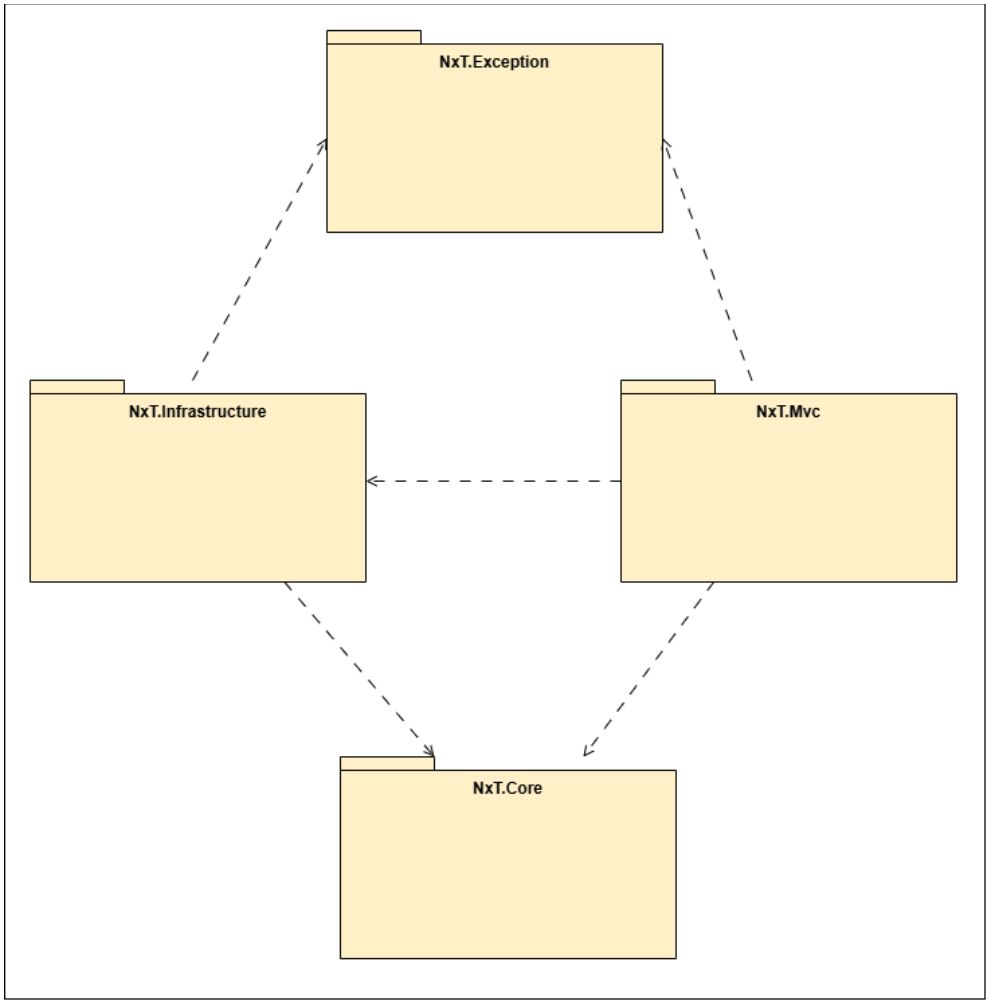
➢ **UC012 – *Registrar credenciais***

- **Atores**: Administrador.
- **Descrição**: Sempre que um gerente ou vendedor é inserido ou removido do sistema, o administrador deve garantir que suas credenciais devem ser inseridas ou removidas, respectivamente.

7. Diagrama de Classes

Para a concepção do **Diagrama de Classes de Sistema**, levou-se em consideração a dimensão do projeto e sua composição em pacotes. Em resposta a isso, o diagrama foi quebrado em unidades significativas com classes que compartilham contextos semelhantes, ou seja, detalhando uma classe em determinado diagrama e, noutro diagrama, apresentando a mesma em alto nível.

Uma forma de ilustrar, como uma visão geral desta composição, elaborei um esboço de um Diagrama de Pacotes que explicita como cada camada depende uma da outra:



Conforme mostra o diagrama acima, a segregação de alguns pacotes podem lembrar um pouco da abordagem da *Clean Architecture* (*Domain* – que chamei de *Core* –, *Infrastructure*, e *Mvc* como uma camada de apresentação). O pacote de exceções contém erros customizados tanto de

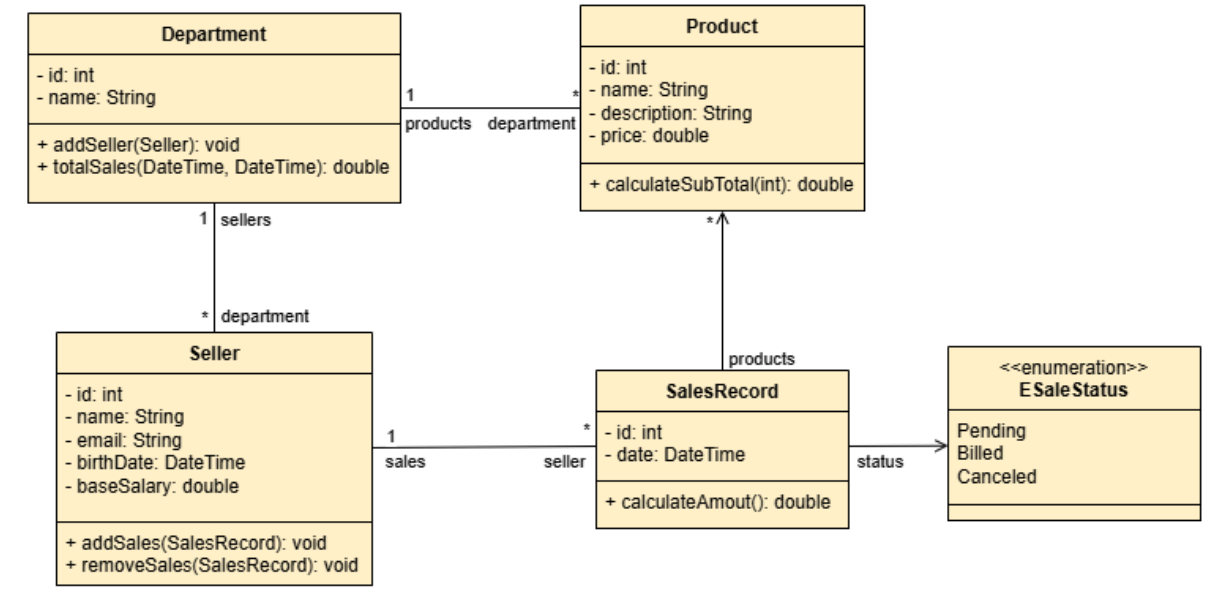
aplicação, quanto de sistema para melhor contextualização ao apresentar erros ao cliente.

Por fim, com este arranjo de pacotes, fica evidente o padrão *MVC + DAO*. Porém, na especificação das classes, não fiz questão de detalhar as *Views*, já que na arquitetura do *ASP.NET MVC*, essa camada é renderizada a partir de arquivos *.cshtml*, que é HTML "puro" praticamente.

7.1 Camada de Modelo (*Models)*

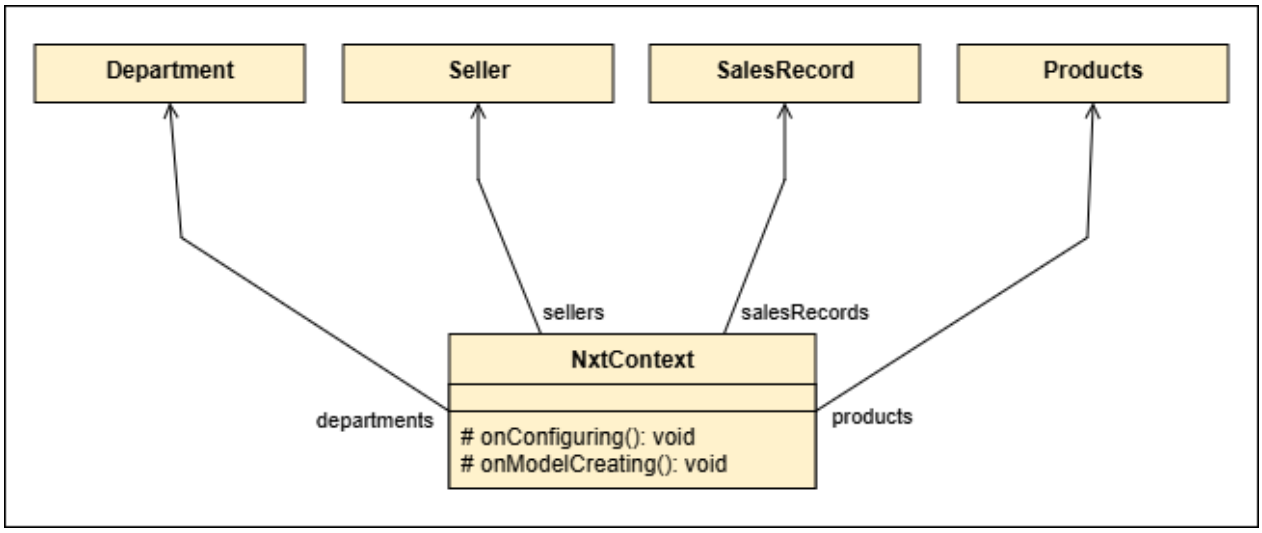
A camada de modelo engloba as classes que são as entidades do domínio, sendo: ***Product***, como conceito central, representa produtos que são vendidos pela empresa, que fazem parte de um departamento e estão presentes em um registro de venda; ***Departament*** é a representação de um departamento de vendas da empresa; ***Seller*** é o conceito de vendedor que trabalha com um certo departamento; ***SalesRecord*** é o conceito de uma venda realizada por um certo vendedor.

* - ***ESalesStatus*** é um enumerador que representa os estados de uma venda (***SalesRecord***).



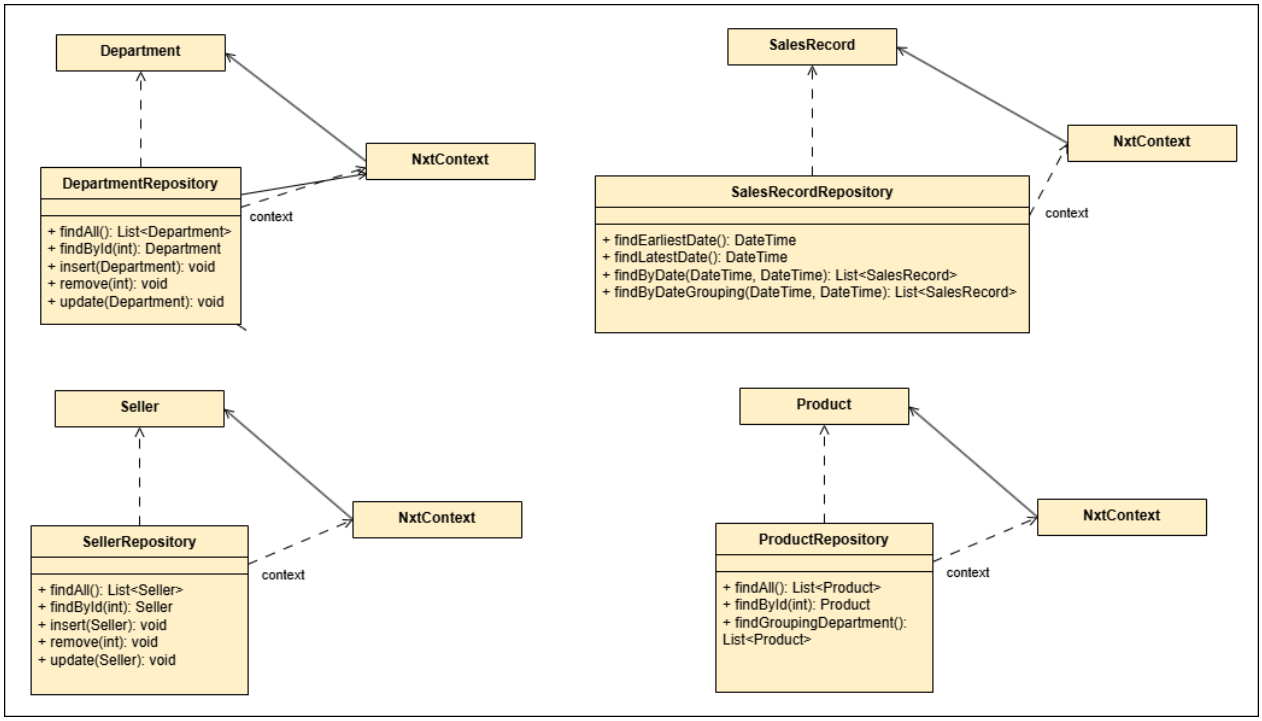
7.2 Camada de *DAO* (Data)

No desenvolvimento em **ASP.NET**, o framework fornece uma abstração nativa para DAO (escrever *queries* manualmente), que é o ORM* **Entity Framework Core**. Para isso, é preciso criar uma classe que é uma fachada à conexão do banco de dados e suas tabelas – chamada de ***DbContext***. A classe de contexto da aplicação, chamada de ***NxtContext***, contém referências para as tabelas significativas do domínio.



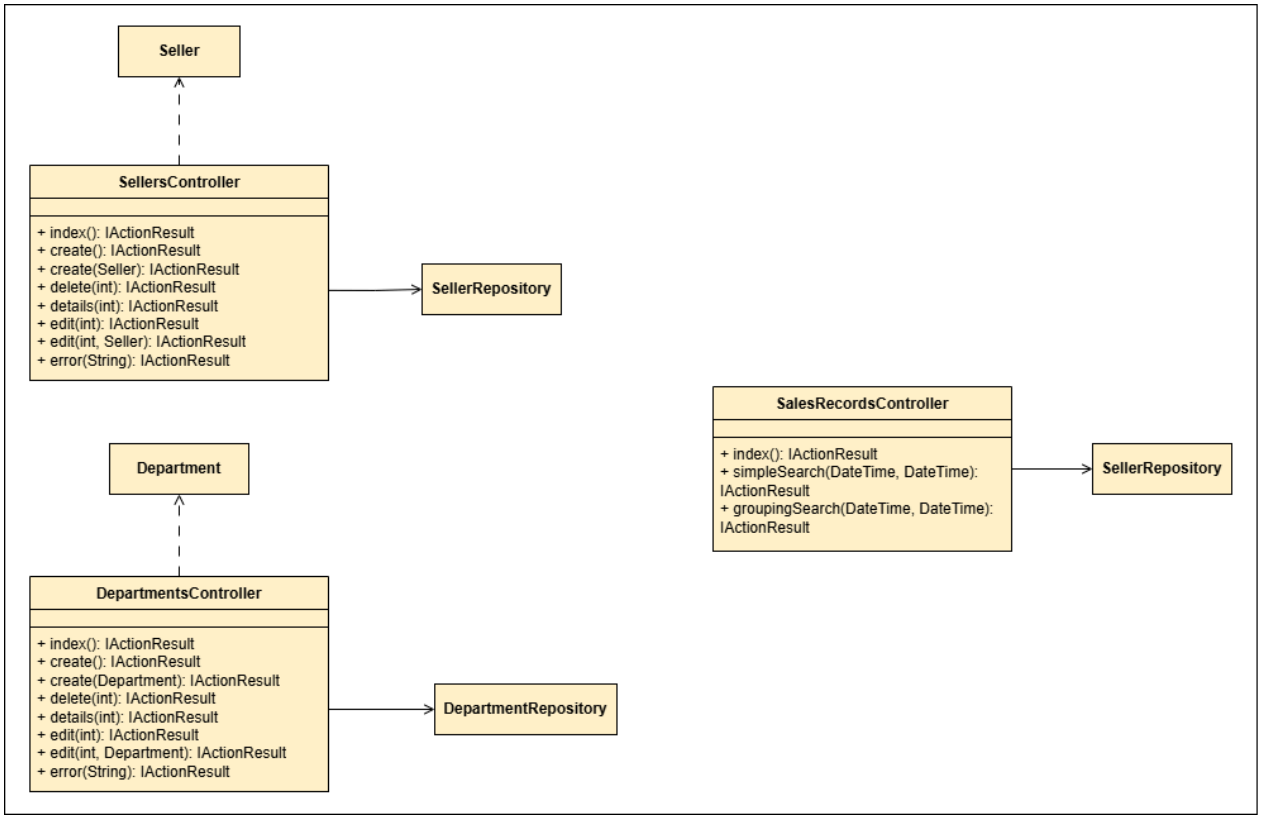
7.3 Camada de Repositórios (*Repositories*)

Com o intuito de segregar responsabilidades e, consequentemente, evitar dos controladores conhecerem todas as tabelas, foi criado uma camada de repositórios que fornecem serviços especializados com cada entidade-classe presentes em ***NxtContext*** e seu respectivo controlador.



7.4 Camada de Controladores (*Controllers*)

Uma das camadas mais importantes no projeto, que modela cada controlador da especificação preliminar.

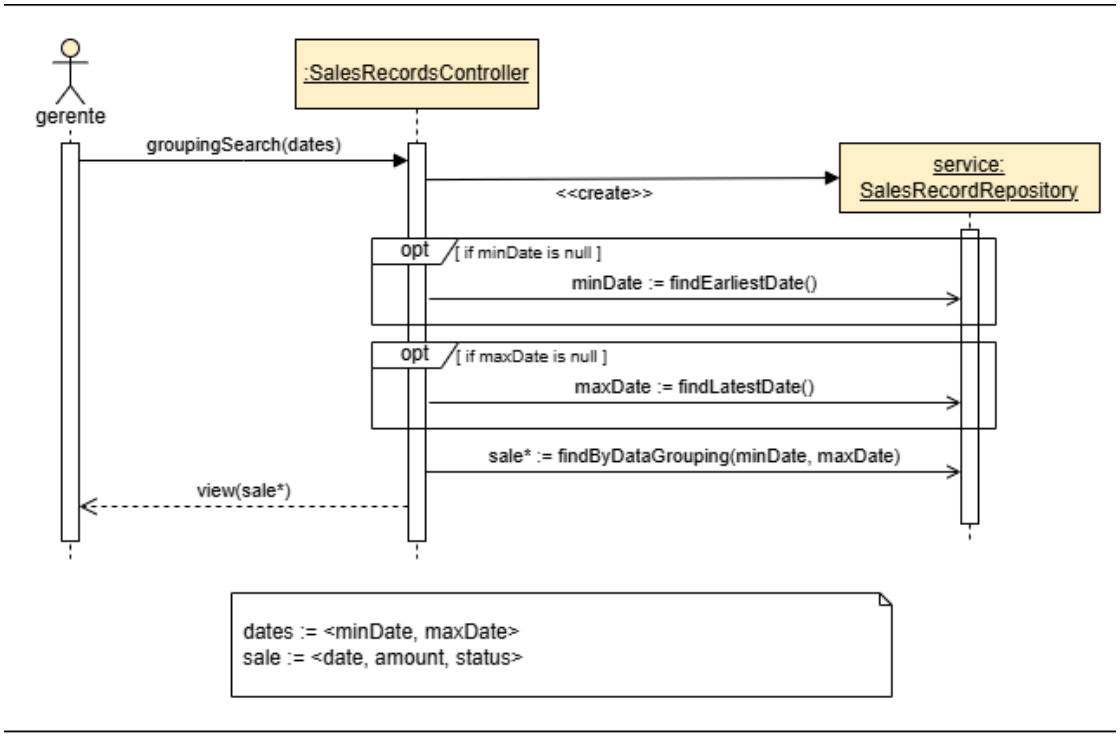


8. Diagrama de Sequência

O **Diagrama de Sequência de Objeto** é uma abordagem do diagrama de sequência da UML com o intuito de descrever um caso de uso através do fluxo de mensagens entre objetos. Segue abaixo a descrição de três casos de uso selecionados para compor a especificação.

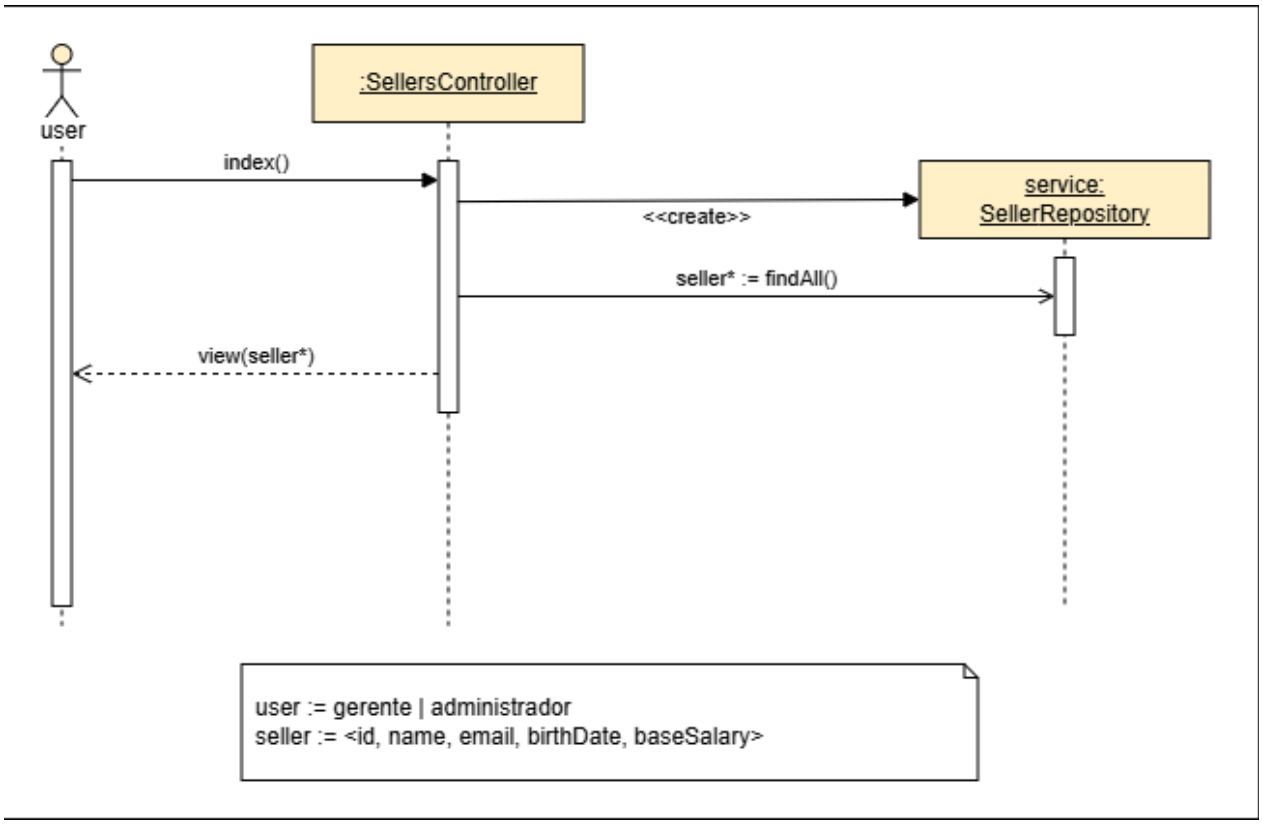
8.2 UC001 – <<report>> *Vendas por* departamentos

O fluxo de buscar vendas por departamentos é baseado no gerente especificar uma data mínima e máxima (caso não seja informado um ou outro, será buscado –através do repositório – a data mais antiga e/ou mais recente das vendas existentes). Com essas essas datas, é realizada uma consulta, agrupada por departamento, das vendas que satisfazem o intervalo.
Por fim, retorna a *view* com esses dados.



8.3 UC005 – <<fragment>> *Visualizar vendedores*

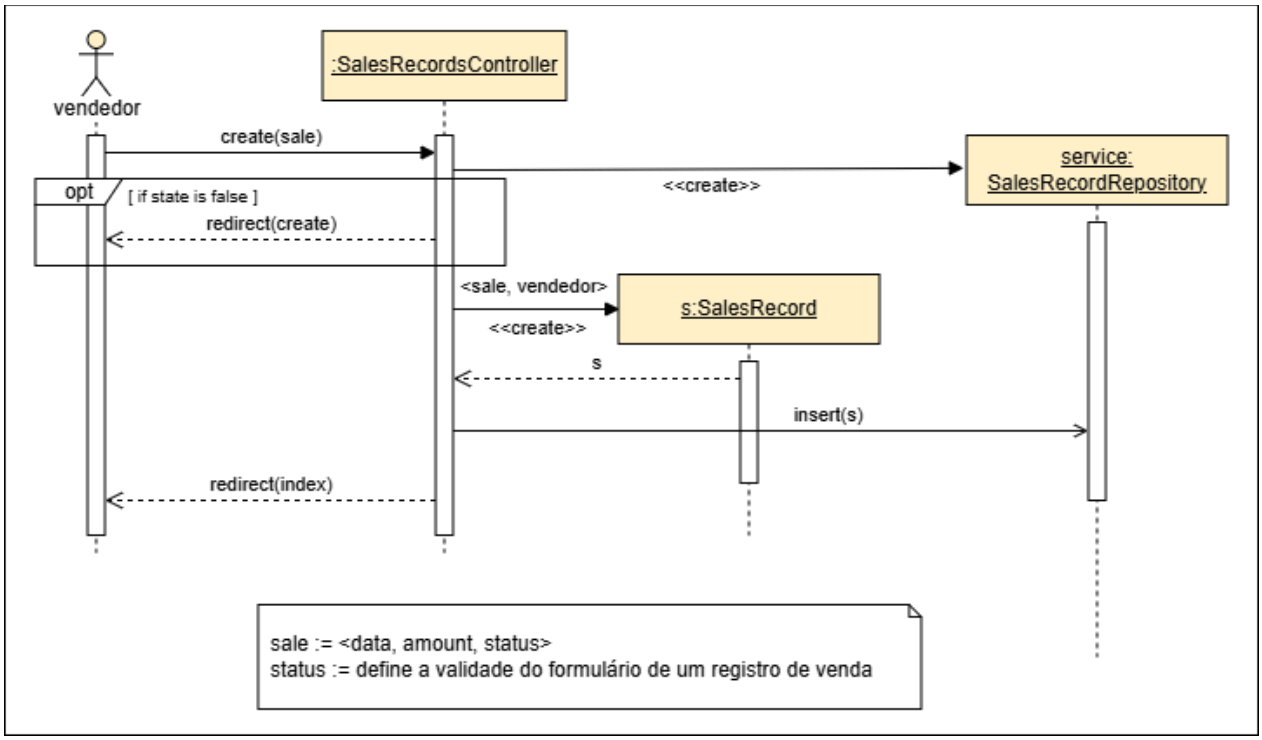
O usuário acessa a página de vendedores que, por padrão, realiza a busca de todos os vendedores cadastrados e retorna a *view* com estes dados.



8.1 UC006 – <<crud>> *Manter vendas* (*registrar venda*)

Para registrar uma venda, o vendedor submete os dados preenchidos no formulário. Caso o formulário esteja inválido (dados incorretos ou faltantes), ele é redirecionado para o formulário novamente. Caso seja válido, uma instância de ***SalesRecord*** é criada e, através do repositório, é inserida no banco de dados.

Em caso de sucesso, é redirecionado para a página inicial que apresenta todas as vendas.



9. Reutilização de Software

No contexto de reutilização de software, a **listagem de features** consiste em identificar e documentar as funcionalidades de um sistema, classificando-as como obrigatórias, opcionais ou alternativas, para entender seu escopo e potencial de reutilização. O **mapa de produtos** é uma representação visual que mostra as relações entre diferentes produtos ou variações de um sistema, destacando funcionalidades compartilhadas e específicas. Já o **modelo FODA (Feature-Oriented Domain Analysis)** organiza e modela funcionalidades comuns e variáveis de um domínio, facilitando a criação de linhas de produtos de software reutilizáveis.

9.1 Listagem de *features*

- **Mandatório**:
  - Login de Usuário – Autenticação por e-mail e senha;
  - Permissões de Usuário;
  - Sincronização de Produtos – Atualização de informações no estoque;
  - Registro de Vendas – Associado a produtos e departamentos;
  - Visualização de Métricas – Indicadores de desempenho por departamento e vendas;
  - Cadastro e Edição de Departamentos – Nome, código e descrição;
  - Criptografia – Proteção de dados sensíveis com SHA256;
  - Autenticação JWT – Controle de acesso a rotas.
- **Opcional**:
  - Acesso Mobile – Funcionalidades adaptadas para dispositivos móveis;
  - Notificações – Alertas de atividades no sistema;
  - Gerenciamento Avançado de Estoque – Controle por lotes e validade;
  - Descontos Personalizados – Aplicação de descontos por vendedor;
  - Histórico de Alterações – Registro de modificações realizadas.

- **Ou**:
    - Geração de Relatórios – Relatórios detalhados por departamento, vendedor ou período;
    - Visualização de dados interativa – o sistema se utiliza de dados de forma dinâmica com gráficos interativos, filtros, e dashboards customizáveis;
    - Exibição Gráfica – Gráficos interativos;
    - Exibição Tabular – Listas detalhadas de dados.
- **Alternativo**:
    - Login com Autenticação Tradicional – E-mail e senha.
    - Login com Biometria – Impressão digital ou reconhecimento facial.

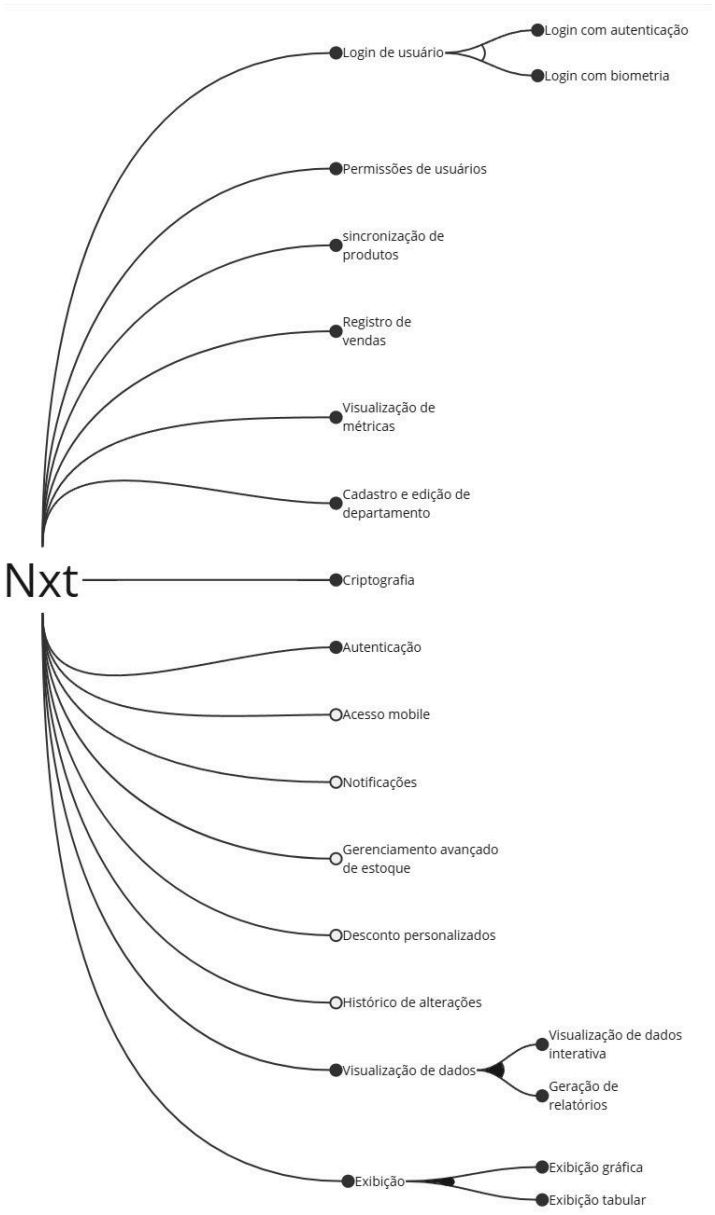
9.2 Mapa de produtos

Mapa de Produto

Features	**Modelo 1**	**Modelo 2**	**Modelo 3**	**Modelo 4**	**Modelo 5**
Permissão	X	X	X	X	X
Sincronização de produtos	X	X	X	X	X
Registro de vendas	X	X	X	X	X
Visualização de métricas	X	X	X	X	X
Cadastro e Edição de Departamentos	X	X	X	X	X
Criptografia	X	X	X	X	X
Autenticação JWT	X	X	X	X	X
Login com Biometria		X	X		X

Login com Autenticação	X			X	
Acesso Mobile	X			X	X
Notificações	X		X		X
Gerenciamento Avançado de Estoque	X	X	X	X	
Descontos Personalizados	X	X		X	
Histórico de Alterações	X	X			X
Gerar Relatório		X	X	X	X
Visualização de dados interativa	X	X			X
Exibição Gráfica	X		X	X	X
Exibição Tabular	X	X		X	

9.3 Modelo *FODA*



10. Arquitetura de Software

A **Arquitetura de Software** descreve a estrutura de alto nível de um sistema, identificando seus componentes principais, suas responsabilidades e como esses componentes interagem entre si. Ela é uma representação abstrata que define a organização do software, seus padrões, tecnologias e os princípios de design utilizados para garantir que o sistema seja escalável, seguro, confiável, de fácil manutenção e, acima de tudo, *testável*.

10.1 Arquiteturas Candidatas

Para decidir quais arquiteturas são adequadas e, mediante este levantamento, definir qual deve ser aquela a ser implementada exige um esforço de análise em termos do escopo de projeto e as necessidades de negócio.

Dito isso – além do que já foi apresentado em tópicos anteriores –, podemos avaliar *três* arquiteturas candidatas junto com suas especificidades (tais especificidades que nos levaram a sua escolha em potencial):

- *Clean Architecture* (Arquitetura Limpa) – baixo acoplamento, alta coesão, definição da **malha de dependências**[1] e orientado a casos de uso (funcionalidades ditam os detalhes de implementação);
- *Hexagonal Architecture* (Arquitetura Hexagonal) – semelhante à *Clean Arch* (até em nível de diagramação, ambos são semelhantes), mas com uma característica notável: fluxos de mensagem são intermediados por **portas** e **adaptadores**;
- *Model-View-Controller* (MVC) – padrão consolidado (nativo em muitos frameworks Web), segregação de responsabilidades e flexibilidade.

Como pode-se ver, as arquiteturas selecionadas são **monolíticas**, isso é uma decisão de projeto. Por quê? Conforme o que já foi especificado sobre o negócio e a solução nos capítulos anteriores, o sistema não visa ser escalável ou atender múltiplos clientes em curto prazo. Pensar em alguma complexidade de arquiteturas distribuídas, como *Microservices* ou *Event-Driven Architecture* representam custos de complexidades desnecessários para o que ele se propõe a atender.

Para melhor contextualização, o que se busca é uma aplicação que mantenha disponibilidade por horário de trabalho (8h/dia) por 6 dias na semana – considerando uma semana com carga horário padrão –, além de um dia quinzenal para manutenção.

[1] Ao falar em **malha de dependências**, me refiro à visão de alto nível das dependências entre os pacotes do sistema. Logo, a *Clean Arch* define claramente como ele deve ser projetado.

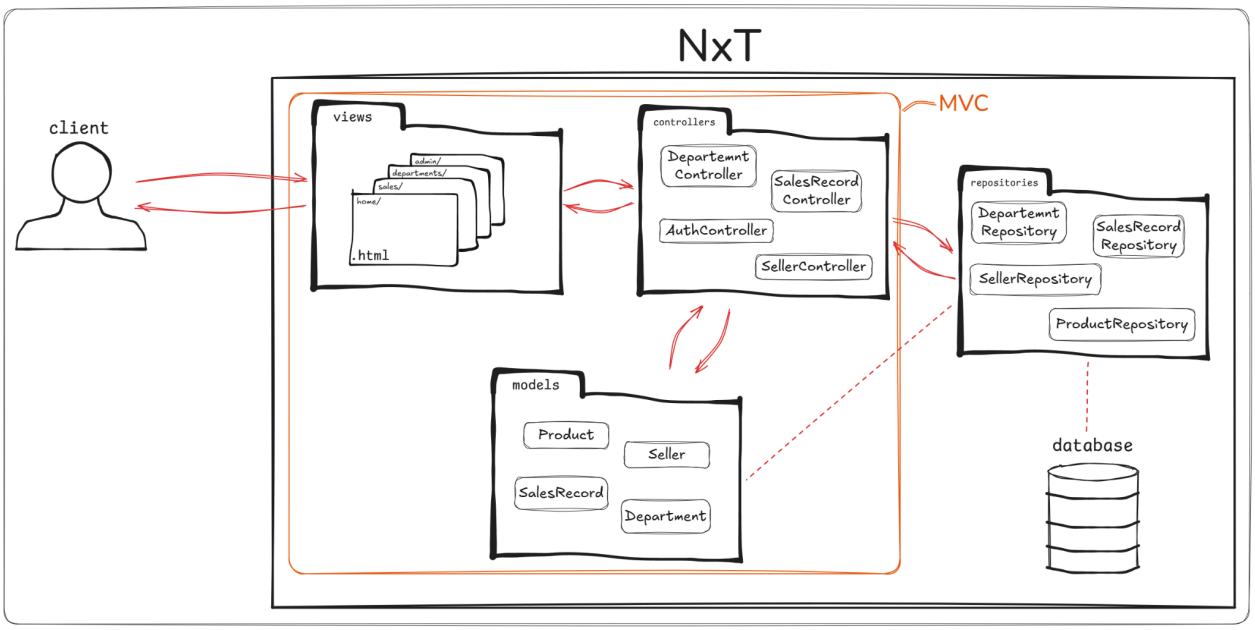
10.2 Decisão de Arquitetura

Mesmo candidatando outras arquiteturas, já é conhecido que o sistema admite uma implementação do padrão arquitetural MVC. Mas, mesmo assim, o que nos leva a manter esse padrão ao invés das outras arquiteturas já apresentadas (fora o esforço de refatoração)?

O primeiro motivo seria a complexidade do sistema: mesmo admitindo um escopo abrangente, a abstração é suficiente para considerá-lo relativamente simples e rápido de implementar e manter. O segundo motivo, sendo o mais importante, são os requisitos de sistema (prioritariamente, aqueles de qualidade) – pois como a modelagem condiz com:

1. poucos usuários;
2. exigência moderada de disponibilidade; e
3. sem perspectiva de escala ou crescimento.

Portanto, o esquema da arquitetura pode ser visualizada abaixo, seguindo uma abordagem leve:



As setas entre cada camada (e com o cliente) representa a troca de dados, enquanto que o segmento tracejado serve para indicar que uma camada faz "uso" de algo – por exemplo, a camada de repositório consome/usa o acesso ao banco de dados. Apesar do sistema em nível de implementação não corresponder ao diagrama acima, a estrutura dos pacotes existentes refletem essa intenção.

10.3 Avaliação das Arquiteturas

Para avaliação das arquiteturas, foi considerado alguns *atributos de qualidade* que estivessem em conformidade com os requisitos não-funcionais, como tempo de resposta sem grande exigências, tempo e custo de desenvolvimento e implantação. A seguir, encontra-se a matriz que contém o mapeamento destes atributos para cada arquitetura:

Arquitetura	*Clean Architecture*	*Hexagonal Architecture*	MVC
Desempenho	F	F	P
Tempo de produção	P	P	F

Segurança	F	F	F
Usabilidade	F	P	F
Buildability	P	P	F

11. Conclusão

O desenvolvimento do NxT tem proporcionado à equipe uma valiosa oportunidade de vivenciar e praticar todas as etapas do processo de especificação de software, desde a identificação de problemas reais no mercado varejista até a definição de requisitos, regras de negócio e a modelagem do sistema.

Ademais, a adoção do padrão arquitetural MVC e da metodologia ágil XP fortaleceu o entendimento sobre como organizar e executar projetos complexos de maneira colaborativa e incremental. O uso de técnicas como brainstorming e engenharia reversa foi essencial para identificar e documentar eficientemente os componentes do sistema – assim como foi essencial na etapa de análise preliminar. Além disso, a clara divisão de responsabilidades entre os membros da equipe aprimorou tanto as habilidades técnicas quanto interpessoais, garantindo que o sistema fosse desenvolvido em conformidade com os requisitos estabelecidos no escopo do projeto.

Por fim, o projeto destacou a importância de documentações rastreáveis, essenciais para a manutenção e a evolução do sistema no futuro.

11.1 Percepção dos alunos

11.1.1 Percepção de João Augusto

Participar do desenvolvimento do NxT foi uma jornada incrível e cheia de aprendizados. Ao mergulhar no projeto, pude explorar diversos modelos e técnicas, o que foi ao mesmo tempo desafiador e enriquecedor. Utilizar a Engenharia Reversa para entender e documentar um sistema existente me fez perceber a complexidade por trás de cada funcionalidade e como elas se conectam de forma harmoniosa.

Adotar a metodologia ágil XP nos ensinou a importância de iterar rapidamente e de forma colaborativa, enquanto o padrão arquitetural MVC proporcionou uma base sólida para estruturar o sistema.

11.1.2 Percepção de Avallos Marinho

Trabalhar com a elicitação dos requisitos, diagramas e usar engenharia reversa foi uma maneira de exercitar meus conhecimentos e garantir mais aprendizado. Olhar para o código e ver o que ele faz para refletir isso na documentação, no começo, foi meio estranho, porém acabou sendo meio intuitivo depois que comecei a realizá-la. As reuniões para decidir o que seria feito, qual metodologia usar. Foi muito gratificante realizar essa atividade.

11.1.3 Percepção do Guilherme Dias

Realizar a documentação foi um desafio, pois não tinha essa experiência ainda. Realizar reuniões e decidir o que seria feito, como seria feito me deu uma breve visão de como é o mercado de trabalho e o que uma equipe que faz a documentação realiza para dar início ao projeto. Achei interessante a utilização da engenharia reversa, pois não tinha ideia que dava pra realizar isso, por fim, foi um desafio.

11.1.4 Percepção de Vitor Costa

Todo e qualquer trabalho que exija esforço coletivo é um desafio. E quando se fala em desenvolvimento de software e os meandros da engenharia de software, o grau de abstração é elevado e, nas fases iniciais de exploração, o contato com o negócio e clientes deve ser constante e a equipe precisa está se contactando frequentemente – isso me faz apreciar o processo, apesar de sua dor inerente.

A concepção do NxT tem me engajado até agora, além das novas funcionalidades que já foram discutidas em equipe!

11.1.5 Percepção de Vitor Ravel

Documentar o NxT me mostrou o quanto uma boa documentação é essencial. Com o sistema já implementado, precisei realizar engenharia reversa, analisando o código e seu funcionamento para extrair o máximo de informações. Apesar do esforço significativo, o

processo destacou a importância de registrar cada detalhe para entender melhor o sistema e identificar oportunidades de crescimento.

12. Atas de Reunião

Data da reunião	**Ata**	**Envolvidos**
11/11/2024	Encontro preliminar da equipe para decidir como reaproveitar o template fornecido pela profa. Jacilane. O integrante Vitor Costa propôs que alguns tópicos do template fossem expandidos em tópicos próprios, como foi o caso do tópico **5. Regras de Negócio**.	amo, fgm, jaf, vrc, vcs
15/11/2024	Após definir o arranjo dos tópicos da documentação, a equipe se dividiu entre algumas responsabilidades, seguindo a abordagem XP: - Avallos Marinho: análise e *tracker;* - Francisco Guilherme: desenvolvimento; - João Augusto: testes; - Victor Ravel: desenvolvimento; - Vitor Costa: *onsite leader*. Deste modo, os tópicos foram preenchidos conforme as responsabilidades: - **2. Descrição Geral do Sistema** – Victor Ravel; - **3. Modelo de Processo para Desenvolvimento** – Francisco Guilherme; - **4. Requisitos Funcionais** e **5. Requisitos Não-Funcionais** – Avallos Marinho; - **6. Documentação dos Casos de Uso**, **7. Diagrama de Classes**, **8. Diagrama de Sequência** – João Augusto e Vitor Costa.	amo, fgm, jaf, vrc, vcs
20/11/24	Como este tópico impacta todo ciclo de vida do projeto, era necessário a reunião da equipe para decidir alguns pontos não definidos na última reunião, como os tópicos *3.2 Aplicação das Técnicas* e *3.3 Resultados Obtidos*.	amo, fgm, jaf, vrc, vcs
22/11/24	Última reunião antes da primeira entrega, onde	amo, fgm,

	não houve tantos acréscimos ou alterações. Porém, a equipe se alinhou ao que foi desenvolvido no documento ao inserir, cada um, sua perspectiva sobre o projeto até este momento, como também sobre as refatorações na estrutura do projeto presente no GitHub.	jaf, vrc, vcs
05/12/24	Conforme especificado no feedback, uma peça central não se encontrava na modelagem: o conceito de produto! Dito isso, os envolvidos refletiram este acréscimo em todo documento que exigia a sua presença: Avallos Marinho adicionou o requisito faltante (**RF001**), enquanto Vitor Costa criou a regra de negócio faltante (**RN002**) e contextualizou a presença deste no escopo.	amo, vcs
06/12/24	Dentre os demais pontos levantados no feedback (ata de reunião ausente, referências bibliográficas faltantes e incompletude dos diagramas UML), o esforço conjunto do grupo foi em corrigir tais pontos.	fgm, vrc, vcs
24/12/2024	O grupo discutiu sobre os tópicos da próxima entrega (*Reutilização de Software* e *Arquitetura de Software*). Então ficou decidido o grupo se separar para desenvolverem as tarefas paralelamente entre cada um dos tópicos; para fins de registro, o integrante Vitor Costa adicionou as tarefas no board do *Projects* no GitHub.	amo, fgm, vrc, vcs
10/01/2025	Última reunião antes da entrega como uma retrospectiva, revisando partes do projeto (como por exemplo, aperfeiçoamento do modelo FODA). Por fim, todos envolvidos concordaram com o envio.	amo, fgm, vrc, vcs

13. Glossário

Uma breve descrição sobre os termos utilizados neste documento que podem gerar dubiedade de interpretação ou que pertencem ao domínio em específico:

- **Departamento**: Unidade organizacional que agrupa vendedores e vendas no sistema, sendo a base para relatórios por setor.
- **Venda**: Transação registrada no sistema, com status como *Pendente*, *Cancelada* ou *Concluída*.
- **Relatório**: Ferramenta gerada pelo sistema para análise de vendas, departamentos e desempenho de vendedores, permitindo consultas por períodos de tempo.
- **Dashboard**: Interface visual que apresenta informações consolidadas sobre departamentos, vendedores e métricas de vendas.
- **FODA**: *Feature-Oriented Domain Analisys*, método de análise de domínio que viabiliza, no contexto da engenharia de software, uma visão abrangente dos componentes de software e seu potencial reuso.
- **MVC** (*Model-View-Controller*)**:** Arquitetura usada no projeto para organizar o código em três camadas principais: Modelo (dados), Visão (interface do usuário) e Controle (lógica).
- **ASP.NET MVC**: Framework usado na implementação do sistema, baseado na arquitetura MVC.
- **Criptografia**: Técnica para proteger informações sensíveis, como dados de usuários, armazenando-os de forma segura.
- **Banco de Dados Relacional**: Estrutura de armazenamento de dados do sistema, utilizada para manter as informações organizadas em tabelas.
- ***NiD***: ou *Nxt Identifier*, é uma credencial de identificação única para os usuários cadastrados no sistema. Para validar um *NiD*, é preciso realizar o seguinte cálculo: seja o formato da chave $x_1x_2x_3$-$\alpha\beta\gamma$, ela é válida se, e somente se, $\alpha = x_3 - x_1 \pmod{10}$; $\beta = x_2 - x_1 \pmod{10}$; e $\gamma = x_1 - (\alpha + \beta) \pmod{10}$.

14. Referências

Aqui se encontrará todo material utilizado para suporte ao aprendizado e consultas externas. Como, até agora, não fizemos uso de fontes externas para consulta, não linkamos nada.

- FOWLER, Martin. ***UML Essencial: um breve guia para a linguagem-padrão de modelagem de objetos***. 3. ed. Porto Alegre: Bookman, 2005. 156 p. Tradução de: João Tortello.
- MARTIN, Robert C.; MARTIN, Micah. **Seção I: Desenvolvimento Ágil**. In: MARTIN, Robert C.; MARTIN, Micah. ***Princípios, Padrões e Práticas Ágeis em C#***. Porto Alegre: Bookman, 2011. Cap. 1-6. p. 31-120. Tradução de: João Tortello; Revisão Técnica de: Daniel Callegari.
- VALENTE, Marco Tulio. **O que é uma Arquitetura Hexagonal?** 2022. Disponível em: O que é uma Arquitetura Hexagonal? – Engenharia de Software Moderna. Acesso em: 28 dez. 2024.
- VALENTE, Marco Tulio. **Construindo Sistemas com uma Arquitetura Limpa**. 2022. Disponível em: Construindo Sistemas com uma Arquitetura Limpa – Engenharia de Software Moderna. Acesso em: 04 jan. 2025.