Documento de Especificação de Software - NxT

UFC

Documento de Requisitos, Regras de Negócio e Modelagem de Negócio

Versão 0.1

Universidade Federal do Ceará – Campus Russas

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*
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	Preenchendo 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

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.

Portanto, não considera-se a camada de atendimento ao cliente e estratégias de vendas ou o monitoramento de estoque. Então, 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.

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 geral do sistema, especificando 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;
- Seção 3 - **Requisitos Funcionais**: especifica os requisitos funcionais planejados para o sistema;
- Seção 4 - **Requisitos Não-Funcionais**: especifica os requisitos não-funcionais planejados para o sistema;
- Seção 5 - **Regras de Negócio**: 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 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 - **Conclusões**: apresenta um resumo das principais lições obtidas ao longo do desenvolvimento do trabalho.
- Seção 10 - **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 11 - **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 registrar as vendas, vendedores e departamentos da empresa em questão, 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** terá acesso completo sobre o gerenciamento de vendedores, gerentes e departamentos.
    2. O **gerente de vendas** terá acesso a visualização de departamentos e vendedores cadastrados, como também ao dashboard.
    3. O **vendedor** terá acesso ao registro e consulta de suas próprias vendas.
- **[RN002]** ***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.
- **[RN003]** ***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.
- **[RN004]** ***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.

➢ **[RN005]** ***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**

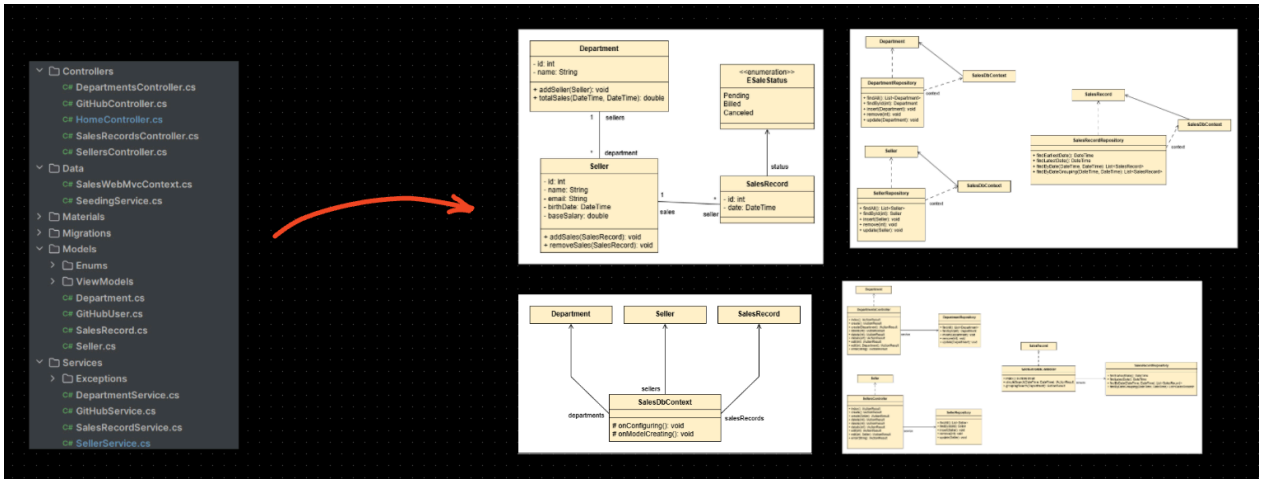
  A equipe precisa de um ambiente de trabalho colaborativo para alinhar desenvolvedores e stakeholders. O XP fomenta a comunicação constante e a programação em par, o que promove sinergia e compartilhamento de conhecimento.

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):

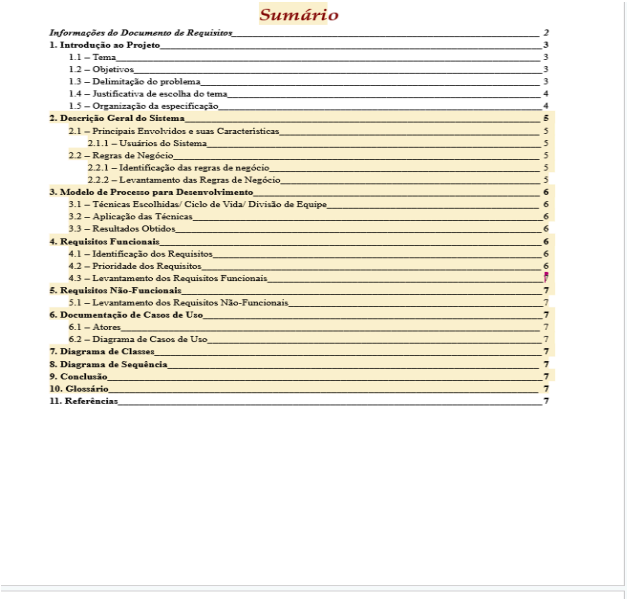
- **Avallos Marinho**: análise e *tracker*, sendo responsável por elicitar os requisitos e histórias de usuário e garantir consistência das necessidades do cliente, como também monitorar entregas.
- **Francisco Guilherme**: desenvolvimento, envolvido na implementação e integração da solução.
- **João 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

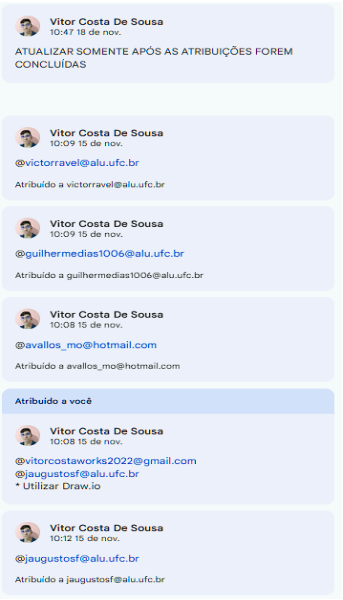
Como primeira característica da implementação destas técnicas foi o fluxo da Engenharia Reversa aplicada para compor a especificação preliminar. Já que a base de código já era existente, construímos a especificação a partir dela e, depois, fizemos as devidas adaptações para que ele venha a refletir a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



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 - Registro de vendas**

O sistema deve permitir que seja registrado o cadastro de novas vendas, vinculadas com informações (Produto, vendedor e departamento)

Prioridade: ☑ Essencial ☐ Importante ☐ Desejável

Regras de Negócio: RN004

➢ **RF002 - 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, RN002

➢ **RF003 - 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, RN003

➢ **RF004 - 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, RN005

➢ **RF005 - 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

➢ **RF006 - 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

➢ **RF007 - 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, RN002

➢ **RF008 - 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, RN002, RN003, RN005

➢ **RF009 - 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

➢ **RF010 - 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.

  Regras de Negócio: n/a

- **RNF004 - Autenticação de usuário**

  Conforme a hierarquia de acessos, as rotas devem ser autenticadas.

  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, 5s.

  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.

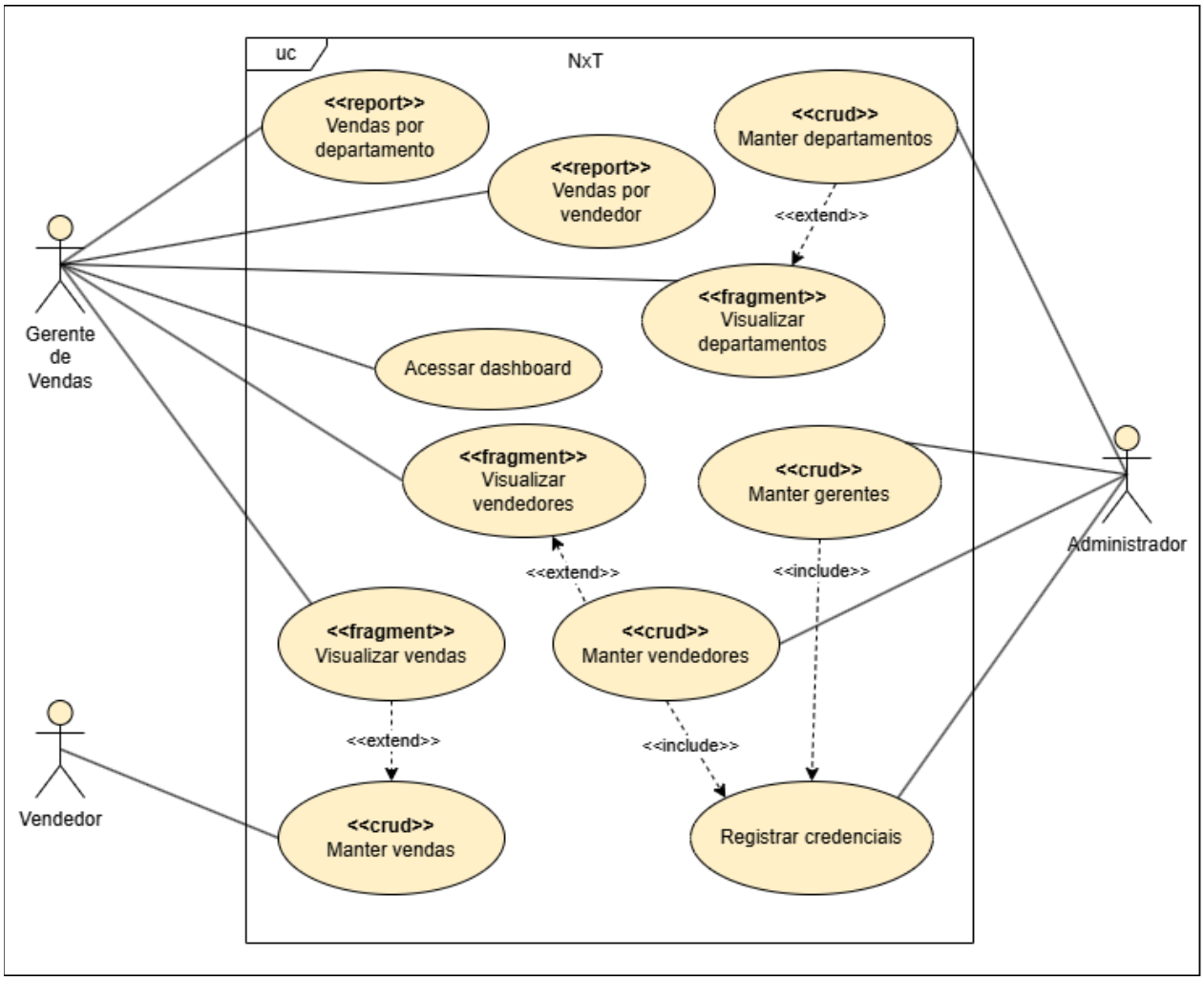
6.1 – Atores

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.

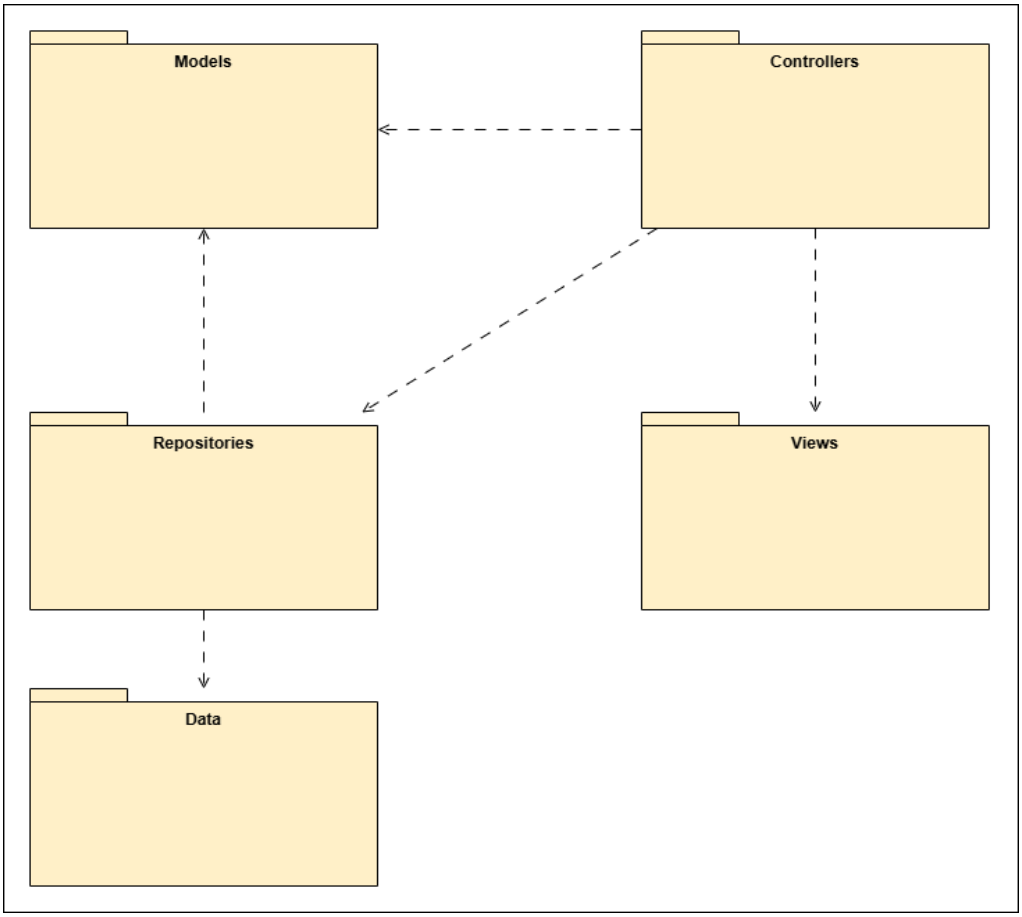
- **UC002 – <<report>> *Vendas por vendedor***
  - **Atores**: Gerente de Vendas.
  - **Descrição**: O gerente de vendas acessa a página de *dashboard* e seleciona, como filtro, algum vendedor; esse filtro vai agrupar as vendas que pertencem a um determinado vendedor.
- **UC003 – *Acessar dashboard***
  - **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 – <<crud>> *Manter departamentos***
  - **Atores**: Administrador.
  - **Descrição**: O administrador pode efetuar a consulta, adição, remoção ou edição de um departamento.
- **UC009 – <<crud>> *Manter gerentes***
  - **Atores**: Administrador.
  - **Descrição**: O administrador pode efetuar a consulta, adição, remoção ou edição de um gerente.
- **UC010 – <<crud>> *Manter vendedores***
  - **Atores**: Administrador.
  - **Descrição**: O administrador pode efetuar a consulta, adição, remoção ou edição de um vendedor.
- **UC011 – *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 concepção de 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:

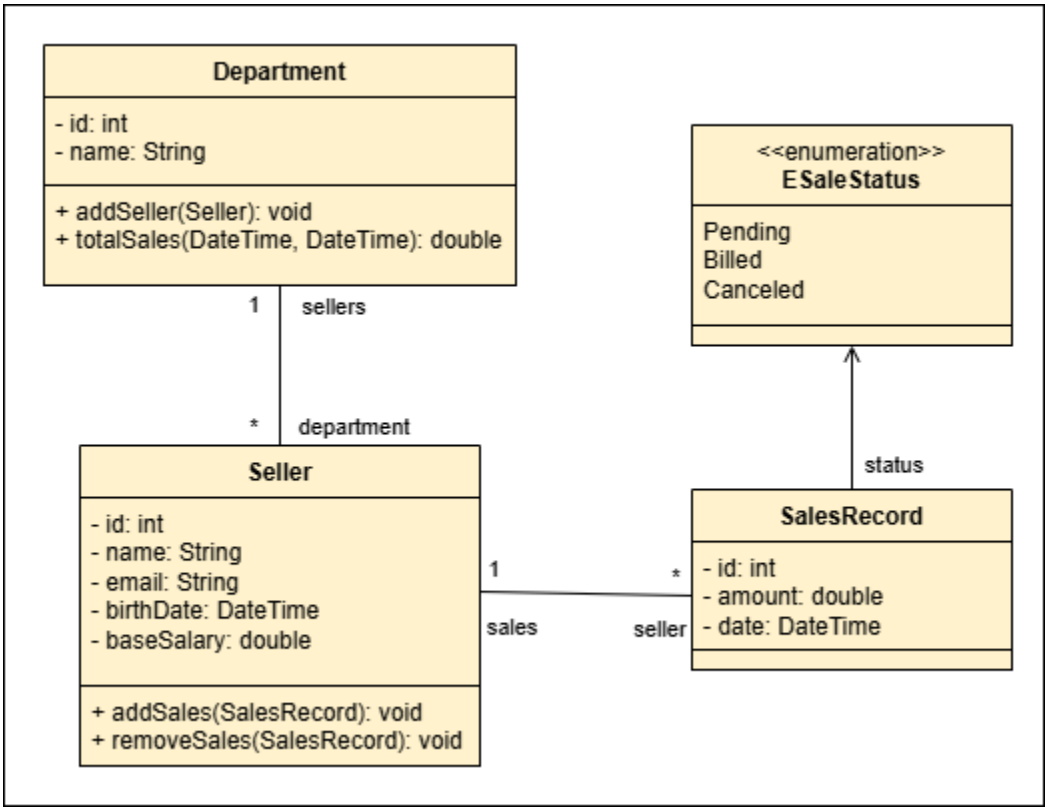


Com este arranjo de pacotes, fica evidente o padrão *MVC + DAO*. Porém, na especificação das classes, não fiz questão de detalhar as *Views*, já que na arquitetura do *ASP.NET MVC*, essa camada é renderizada a partir de arquivos *.cshtml*, que é HTML "puro" praticamente.

7.1 Camada de Modelo (*Models)*

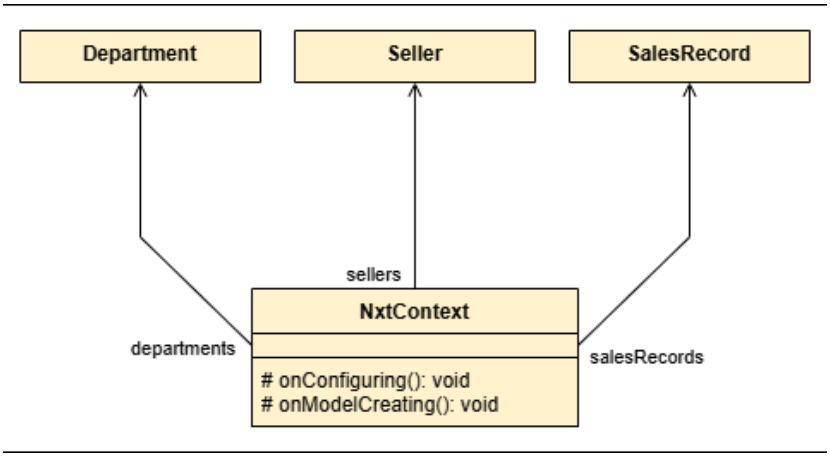
A camada de modelo engloba as classes que são as entidades do domínio, sendo: ***Departament*** é a representação de um departamento de vendas da empresa; ***Seller*** é o conceito de vendedor que trabalha com um certo departamento; ***SalesRecord*** é o conceito de uma venda realizada por um certo vendedor.

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



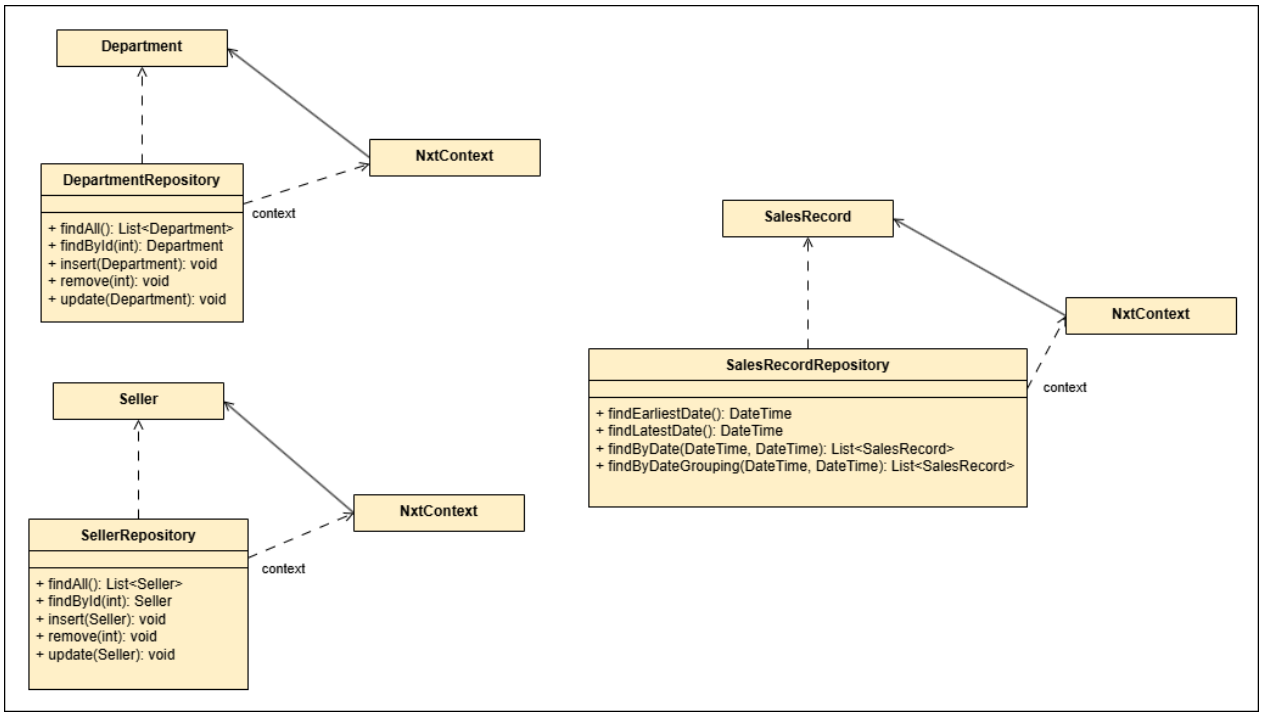
7.2 Camada de *DAO* (Data)

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 três tabelas significativas do domínio.



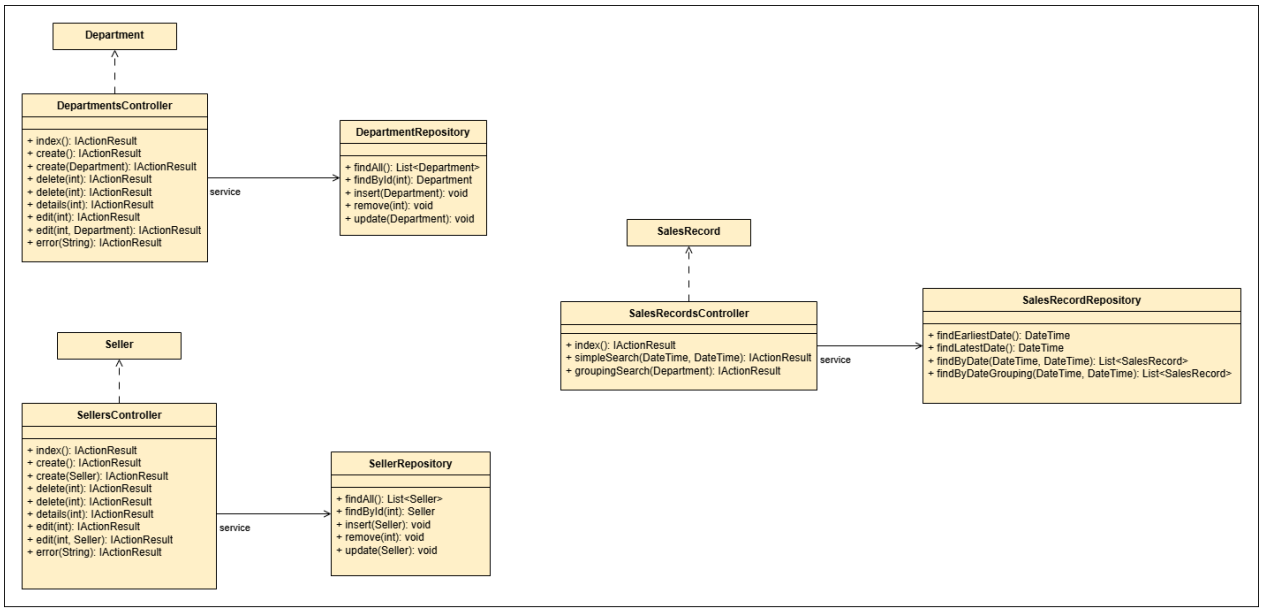
7.3 Camada de Repositórios (*Repositories*)

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.

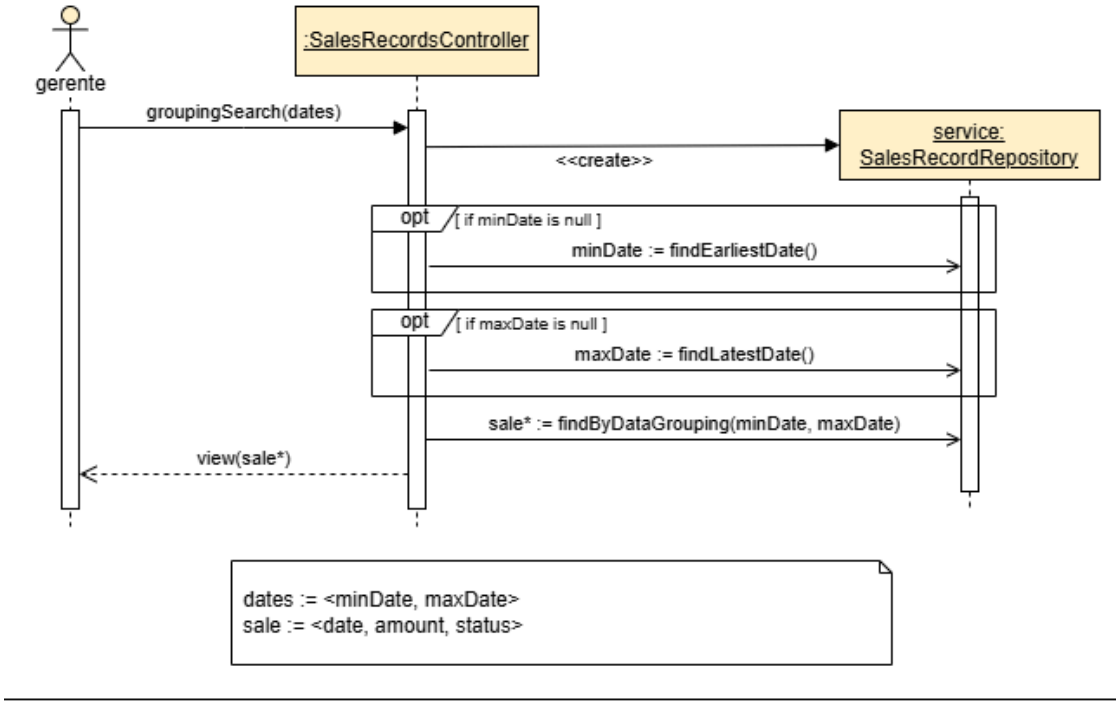


8. Diagrama de Sequência

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.

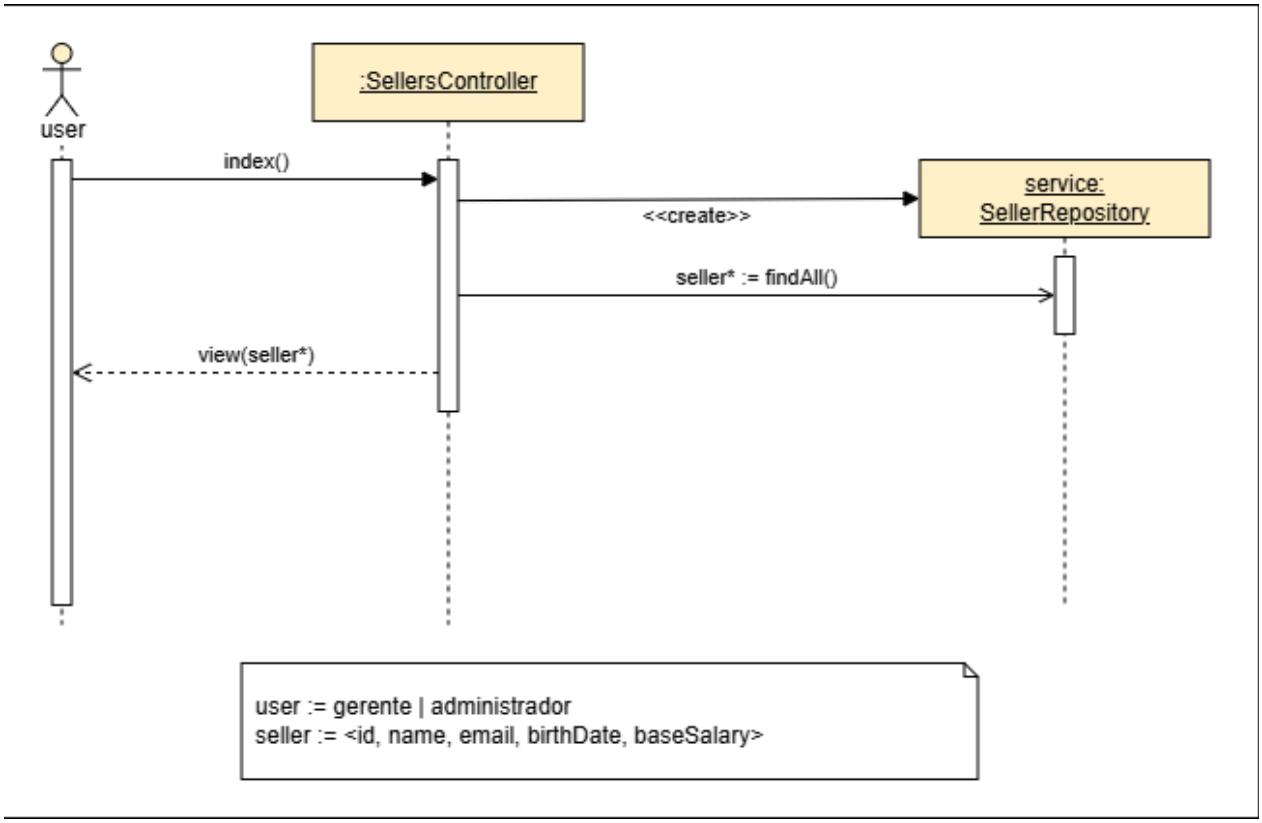
8.2 UC001 – <<report>> *Vendas por* departamentos

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*

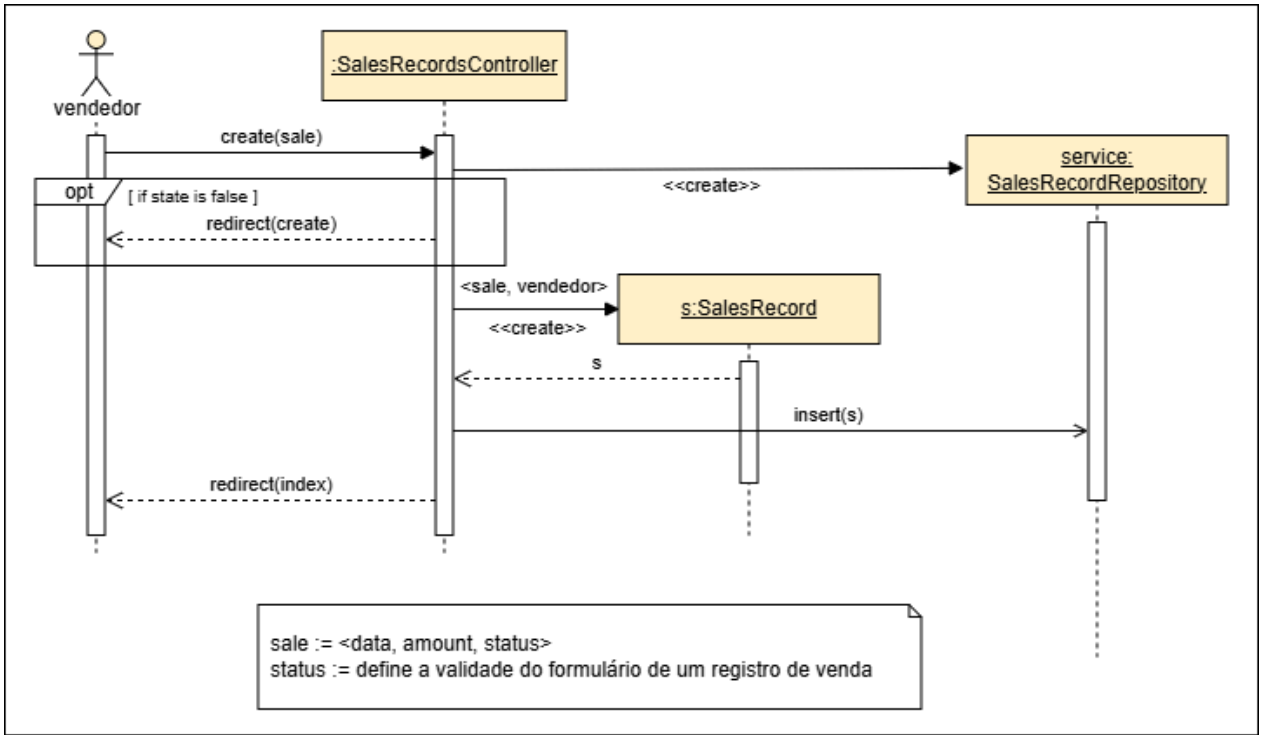
O usuário acessa a página de vendedores que, por padrão, realiza a busca de todos os vendedores cadastrados e retorna a *view* com estes dados.



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. Conclusão

A aplicação de Engenharia Reversa e a elaboração de uma documentação preliminar estruturada estão desempenhando papéis cruciais no desenvolvimento do NxT. A Engenharia Reversa permitiu à equipe entender e decompor o sistema existente, facilitando a identificação de componentes essenciais e a reutilização de ideias em novos contextos. Essa abordagem foi fundamental para garantir uma compreensão satisfatória do domínio e para propor soluções bem alinhadas às necessidades do mercado varejista, especificamente para o modelo de **loja de departamentos**.

Paralelamente, a ênfase em uma documentação inicial clara e rastreável provou ser indispensável, servindo como um guia para o desenvolvimento e fornecendo uma base sólida para futuras manutenções e evoluções do sistema. Esses dois aspectos, quando integrados, não apenas contribuem para a qualidade do software entregue, mas também reforçam a importância de práticas bem fundamentadas na engenharia de software, refletindo diretamente na eficiência e na sustentabilidade do projeto.

9.1 Percepção dos aluno

9.1.1 Percepção de João Augusto

Trabalhar no desenvolvimento do NxT foi uma experiência muito gratificante. Usar Engenharia Reversa me mostrou o quanto um sistema já existente pode ser útil para entender como tudo funciona por dentro. Foi desafiador transformar as funcionalidades em uma documentação clara e objetiva, mas isso acabou sendo um ótimo exercício para melhorar minha atenção aos detalhes e entender melhor como cada parte do sistema se conecta.

9.1.2 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.

9.1.3 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.

9.1.4 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!

9.1.5 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.

10. 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.
- **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 $\gamma = x_1 - (\alpha + \beta) \pmod{10}$.

11. 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.