Desenvolvimento de um software gerenciador de vendas

Bruno F. Bevilaqua[1]

[1]Engenharia de Software – Universidade do Contestado Concórdia(UNC Concórdia)
Concórdia – SC – Brasil

bruno_bevilaqua@protonmail.com.br

Abstract. *This small project aims to develop a sales management software, with simple entries and operating diagrams. Developed in the subject of WEB Development by Professor Max Pezzin valid for the third semester of Software Engineering.*

Resumo. *Este pequeno projeto tem objetivo de desenvolver um software de gerenciamento de vendas, com simples cadastros e diagramas do funcionamento. Desenvolvido na matéria de Desenvolvimento WEB I do professor Max Pezzin válido pelo terceiro semestre de Engenharia de Software.*

1. Introdução e Funcionamento

A ideia deste pequeno software como trabalho final da matéria de Desenvolvimento WEB I surgiu conversando com colegas pensando em algo simples. O objetivo é desenvolver um software de controle de vendas, com login e permissões de usuário básicas.

O software deve possuir um cadastro de pessoas, que terá os seguintes campos: nome, e-mail, telefone e data de nascimento. Juntamente com a pessoa, o cadastro de cliente e funcionário deverão ter uma pessoa vinculada, portanto os campos de cada um destes dois cadastros serão, respectivamente – endereço, UF, cidade, pais, bairro para o cliente e data de admissão e salário para o funcionário.

O software ainda deve permitir um cadastro de usuários com os seguintes campos: nome, e-mail, senha e tipo. Este tipo compreenderá dois valores, “F” de funcionário ou “A” de administrador, tendo a única diferença entre os dois tipos o acesso ou não ao cadastro de usuários, sendo apenas permitido aos administradores.

O software ainda deve guardar um cadastro de produtos com os seguintes campos: descrição, quantidade, valor de venda, valor de compra e unidade de medida. E por último, o registro das vendas efetuadas, vinculando um cliente do cadastro, um funcionário que será o vendedor e um usuário que será quem está efetuando a venda no sistema, além destas informações ainda teremos a data da venda e a observação, caso alguma informação adicional queira ser colocada. Para cada produto vinculado a uma venda é necessário guardar o valor que o produto foi vendido, e a quantidade vendida, para reduzir do estoque do produto.

2. Diagramas de Funcionamento do Software

Inicialmente, com as informações acima sobre o que o sistema deve possuir, é possível desenvolver os diagramas e requisitos sobre o comportamento do software. Iniciando com os requisitos teremos o seguinte.

2.1. Requisitos do Software

Tabela 1. Requisitos funcionais e não-funcionais do software.

Requisitos do Software	
RF: Requisitos Funcionais	
Tipo	Descrição
RF 1	Deve possui um cadastro de pessoas com nome, e-mail, telefone, e data de nascimento que pode ser povoado por qualquer usuário do sistema.
RF 2	Deve possuir um cadastro de funcionário com data de admissão e salário que pode ser povoado por qualquer usuário do sistema.
RF 3	Deve possuir um cadastro de clientes com país, estado, uf, cidade, bairro e endereço.
RF 4	Deve possuir um cadastro de produtos com descrição, valor de venda e de compra, quantidade e unidade de medida que pode ser povoado por qualquer usuário dos sistema.
RF 5	Deve possui um cadastro de usuários com nome, e-mail, senha e tipo(A ou F), podendo ser povoado somente por usuários do tipo administrador(A).
RF 6	Deve possuir um registro de vendas, com o cliente da venda, o funcionário vendedor e o usuário que está registrando a venda, a data da venda e uma observação, bem como os produtos da venda, com o valor e a quantidade que foi vendida.
RF 7	O software deve permitir o login para todos o usuário que possuírem e-mail e senha. Apenas os usuário administradores(A) podem ter acesso ao menu de usuários.
RNF: Requisitos Não-Funcionais	
Tipo	Descrição
RNF 1	O software deve utilizar o banco de dados MySQL 8.0.
RNF 2	O software deve ser desenvolvido utilizando o padrão MVC(Model-View-Controller) em PHP.
RNF 3	O software deve ter uma interface agradável e de fácil entendimento e manuseio.

2.2. Diagramas de comportamento.

Iniciando com dois diagramas de casos de uso simples, o de login e o de venda, os diagramas dos cadastros não detalhei em caso de uso pois o funcionamento é bem simples.

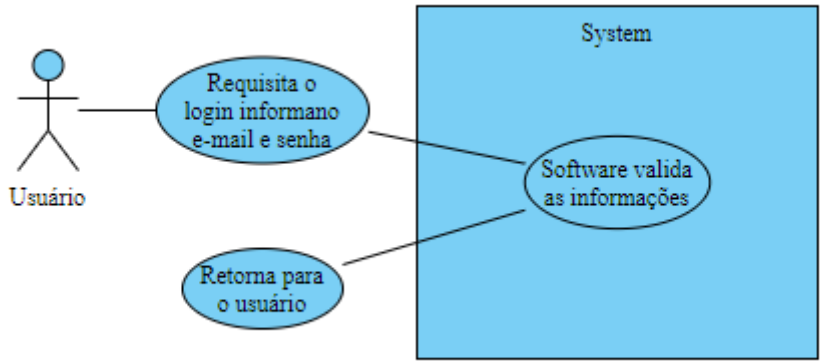

Imagem 1. Diagrama de casos de uso para login.

O diagrama de casos de uso de login funciona dessa forma, o usuário informa os dados de login e o software valida se as informações estão de acordo com algum usuário cadastrado no sistema.

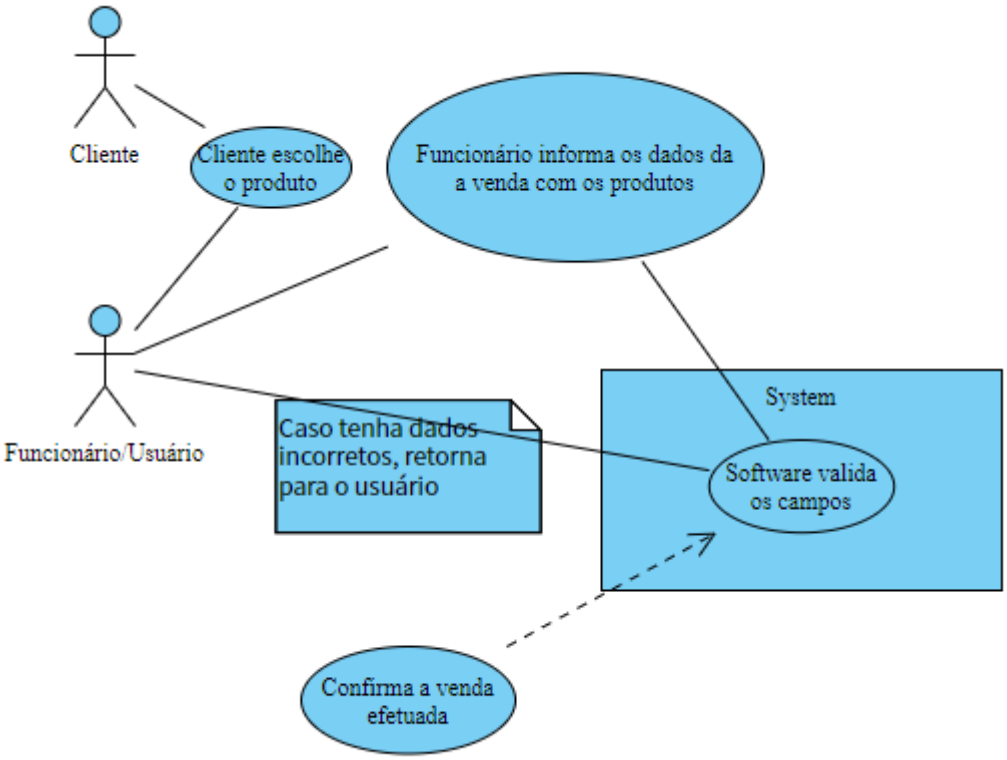

Imagem 2. Diagrama de casos de uso para vendas.

Na venda o processo é bem simples, o cliente informa o funcionário quais produtos deseja comprar e o funcionário, ele mesmo, ou através de outro ator, o usuário, irá registrar a venda informando os dados. O software vai validar os campos e retornar ao usuário um erro ou se foi cadastrado com sucesso.

O diagrama de componentes do sistema tem o simples objetivo de demonstrar quais componentes (classes, interfaces ou afins) do sistema irão comunicar-se entre si. Sendo desenvolvido da seguinte forma.

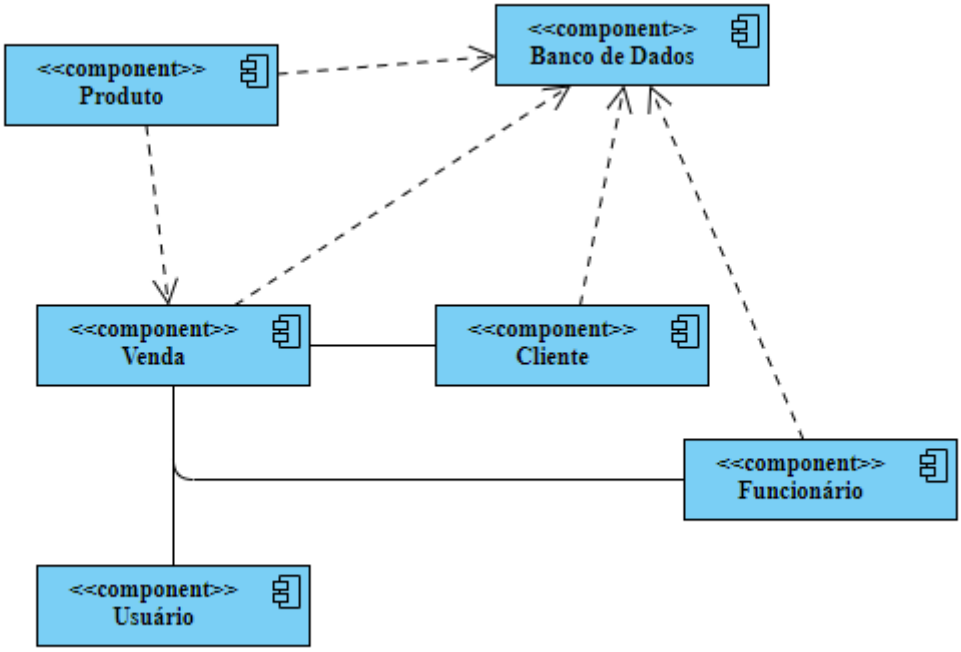

Imagem 3. Diagrama de componentes do sistema.

O diagrama de classes do sistema consiste com quais classes o sistema trabalhará e qual a comunicação delas com as outras classes do sistema. Da seguinte forma.

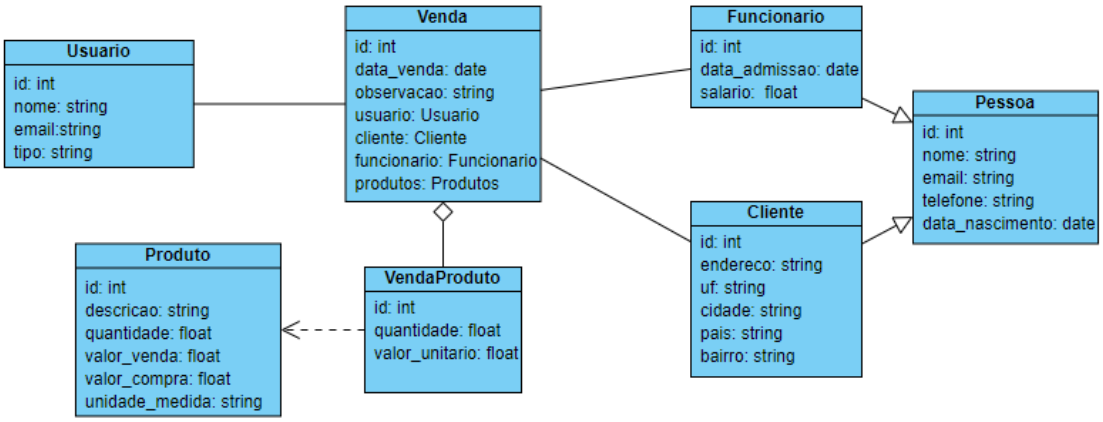

Imagem 4. Diagrama de classes do sistema.

Como foi previsto anteriormente, teremos as classes de funcionário e cliente sendo uma especificação da classe pessoa, e a duas sendo parte da classe venda, assim como o produto que é parte da subclasse VendaProduto, da qual ela compõe a classe da Venda.

Os diagramas de atividades possuem o objetivo de demonstra na prática qual seria o passo a passo das atividades que o sistema executaria ou que o usuário iria interagir com ele. Como esses diagramas são de maior complexidade e exigem um detalhamento maior sobre cada parte do sistema, resolvi desenvolver deles, sendo eles:

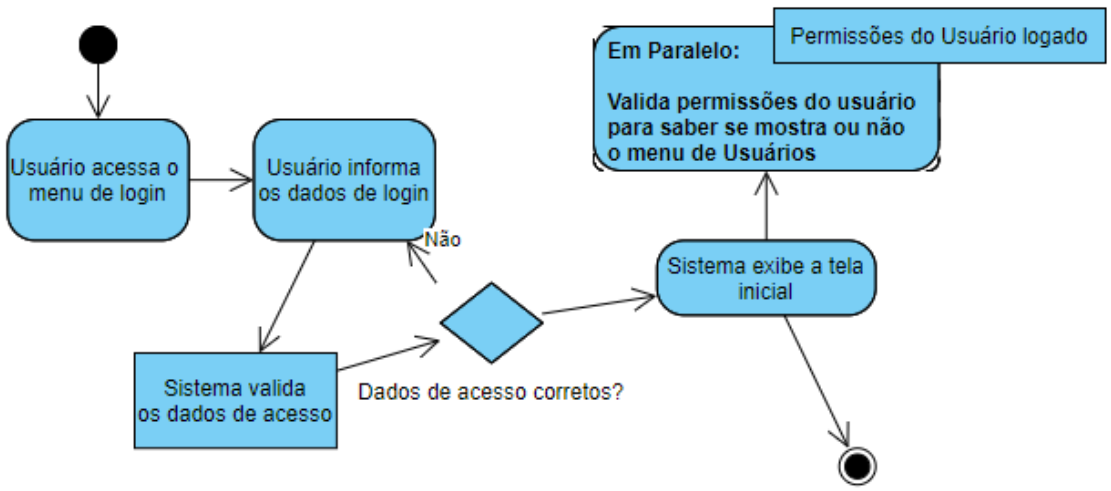

Imagem 5. Diagrama de atividades para login.

O de login – que mostra o passo a passo de como o usuário efetuará o login no sistema.

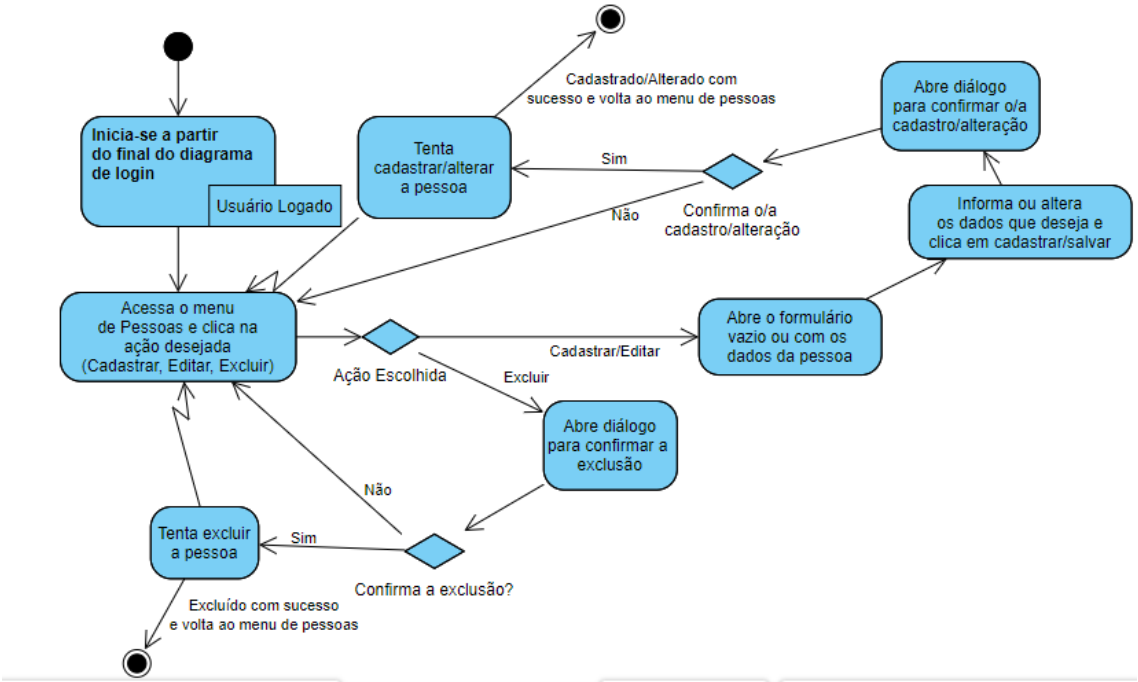

Imagem 6. Diagrama de atividades para o cadastro de pessoa.

O de cadastro de pessoas que detalha o funcionamento completo com as exceções que podem ocorrer no cadastro de uma pessoa no software.

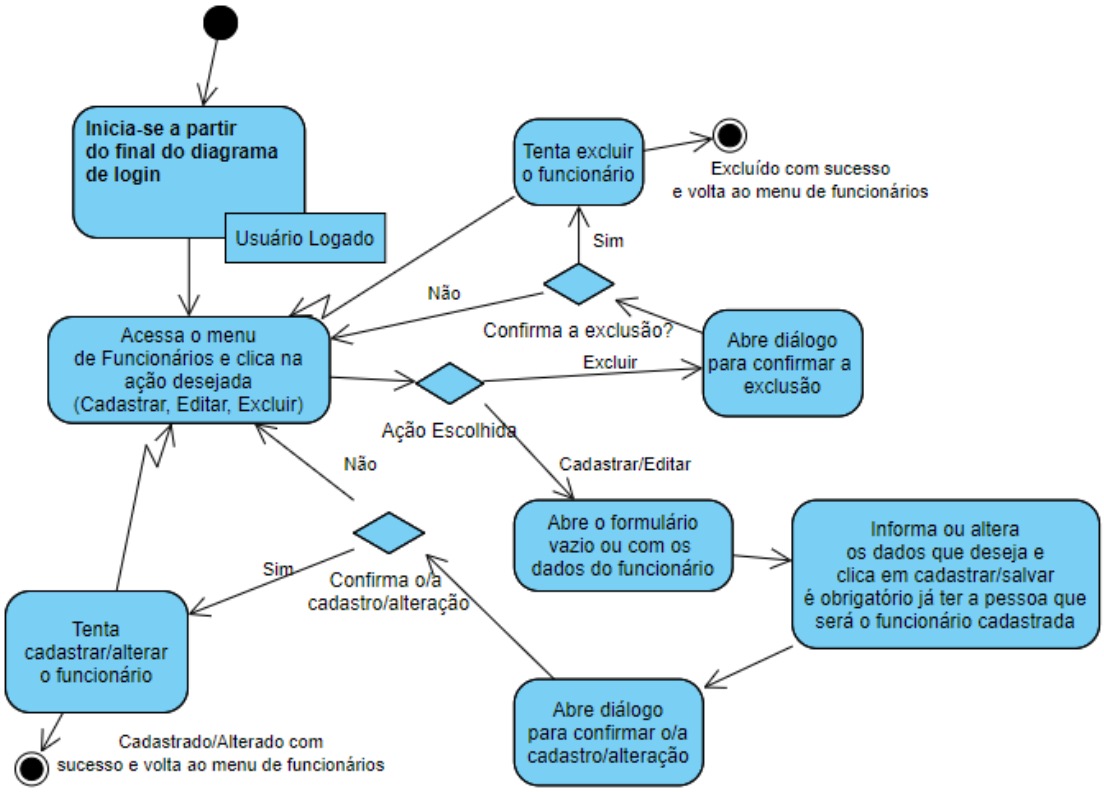

Imagem 7. Diagrama de atividades para o cadastro de funcionário.

O de cadastro de funcionários que detalha as partes envolvidas no cadastro do funcionário.

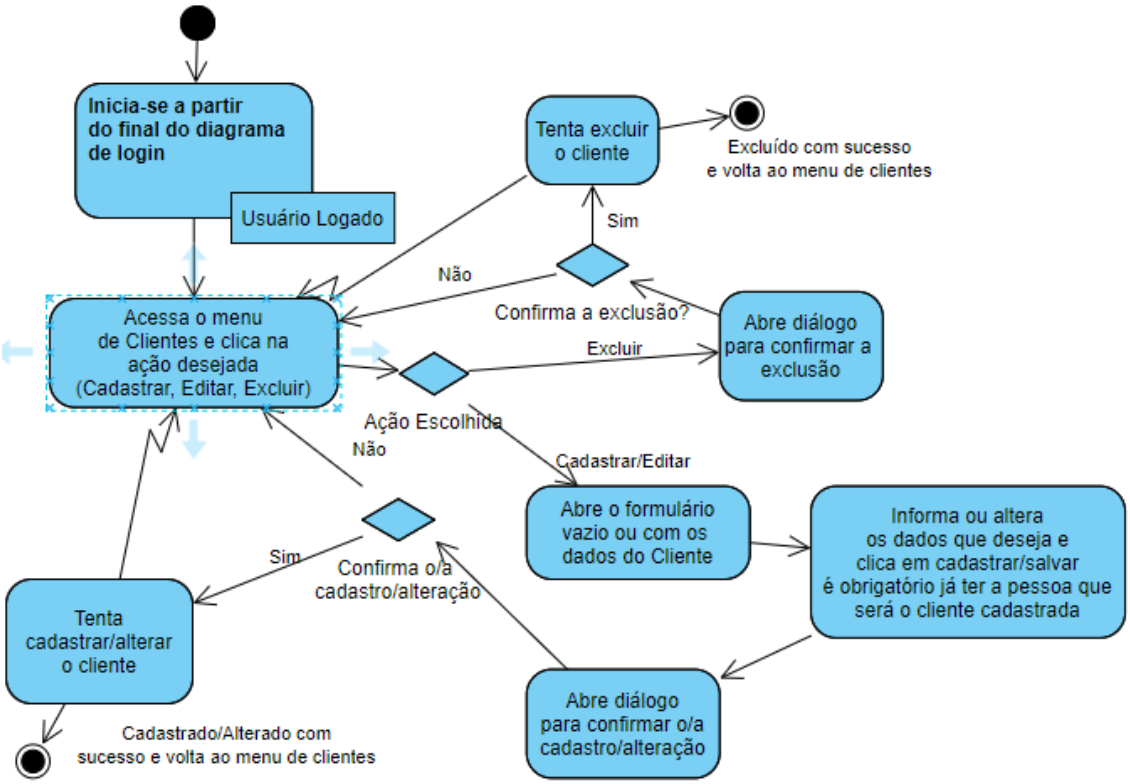

Imagem 8. Diagrama de atividades para o cadastro de cliente.

O de cadastro de clientes que mostra as partes envolvidas no cadastro de um cliente no sistema.

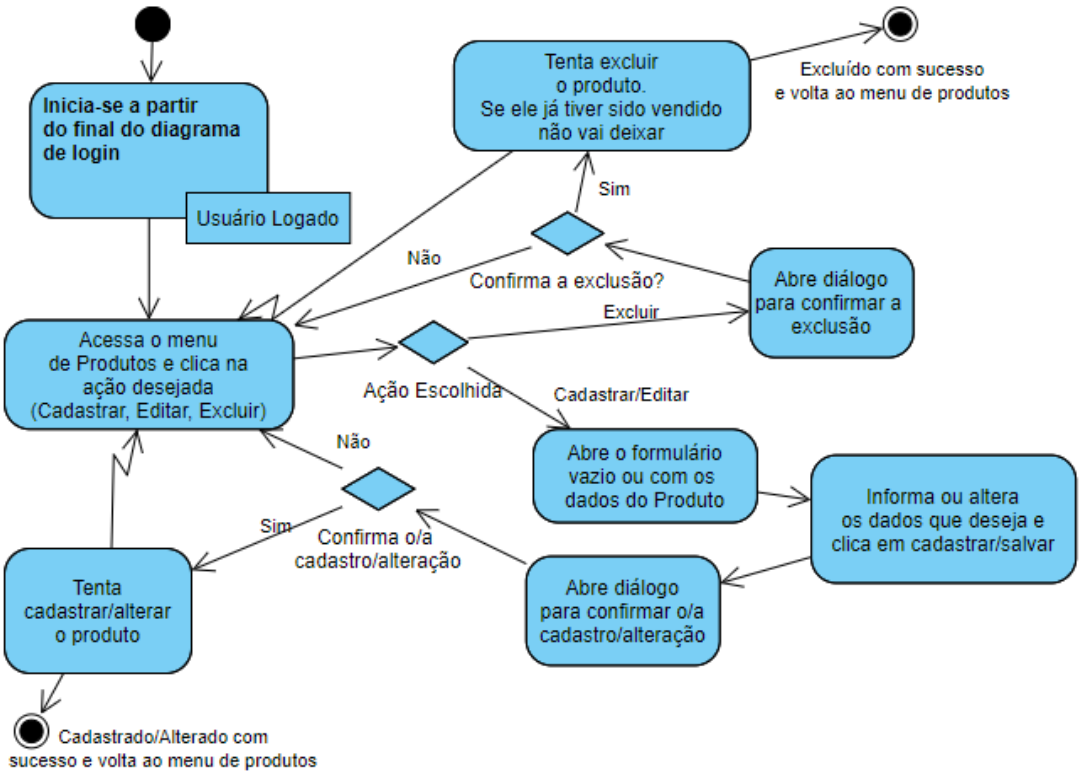

Imagem 9. Diagrama de atividades para o cadastro de produto.

O de cadastro de produto que detalha as validações e informações do cadastro de produtos no sistema.

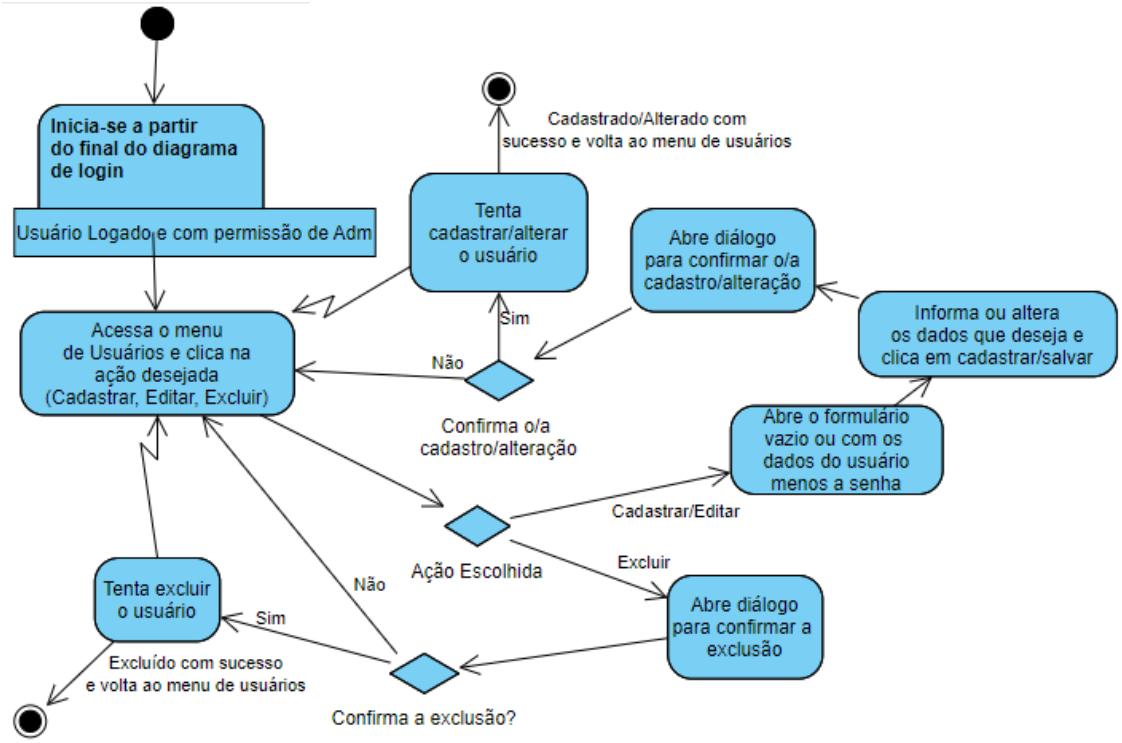

Imagem 10. Diagrama de atividades para o cadastro de usuário.

O de cadastro de usuário que detalha as validações e informações do cadastro de usuários no sistema que só podem ser acessados pelos usuários administradores.

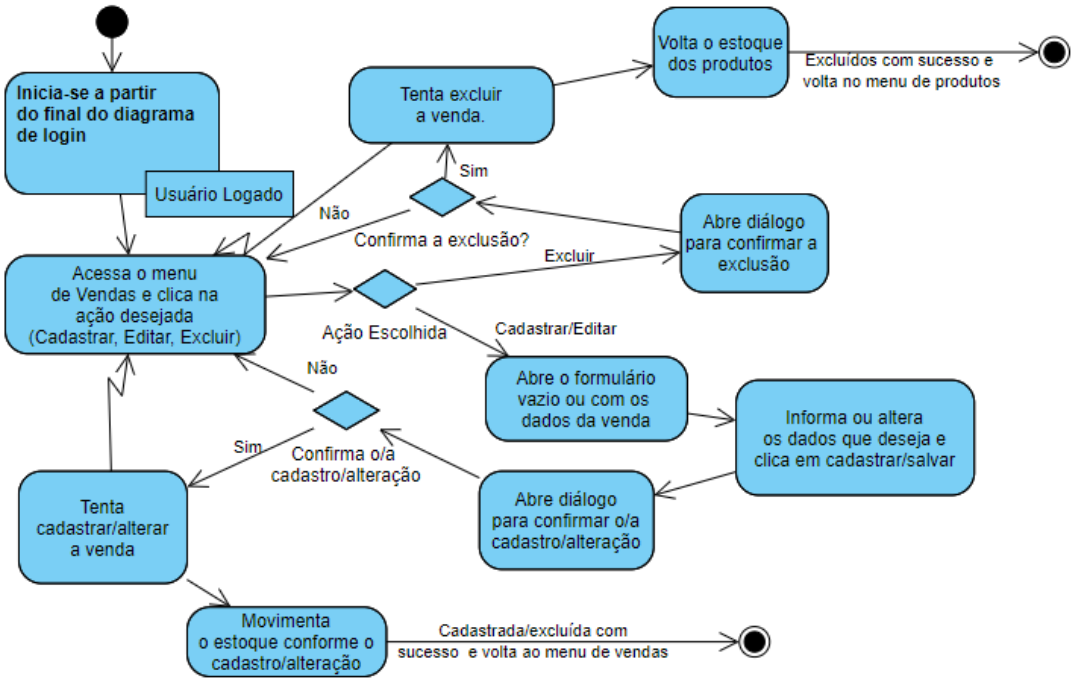

Imagem 11. Diagrama de atividades para o registro de venda.

O de registro de vendas que detalha as informações necessárias para efetuar os movimentos dento do sistema relacionados as vendas. A parte principal e o foco do sistema.

3. A modelagem do sistema

Dados os campos e as informações de comportamento do sistema apresentados anteriormente podemos pensar em uma modelagem conceitual do banco de dados semelhante a seguinte.

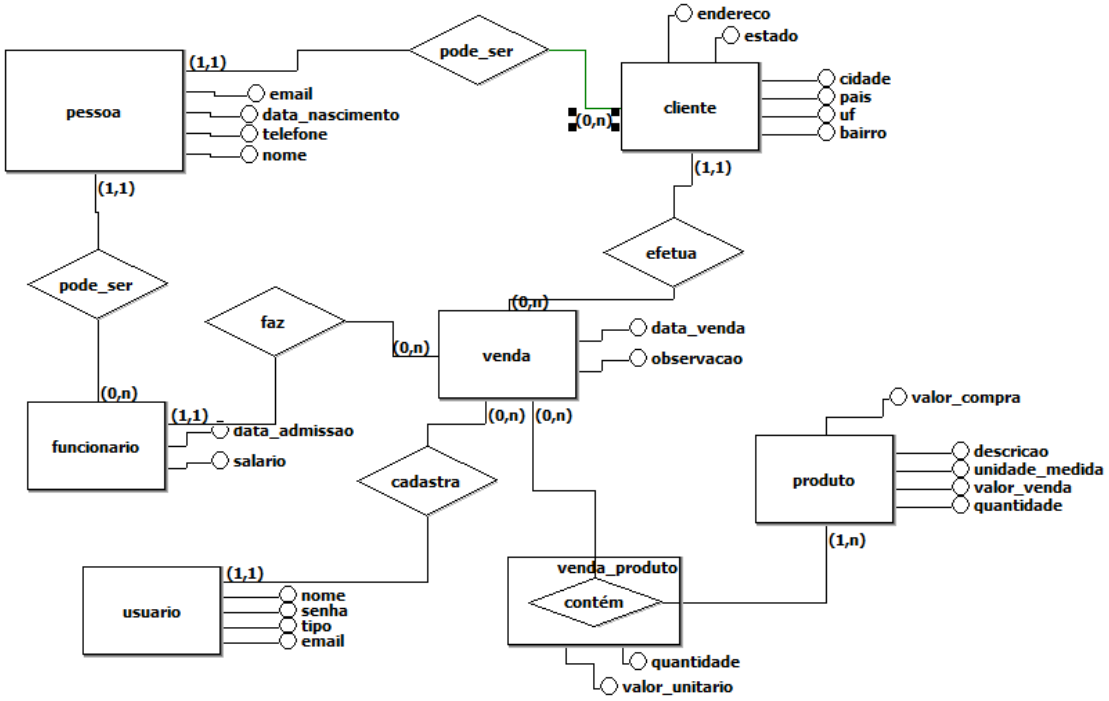

Imagem 2. Modelagem Conceitual do Banco de dados

Baseando-se nesta modelagem conceitual do banco de dados, podemos criar a modelagem lógica bem mais facilmente, sendo assim, o modelo lógico do software pode ser o seguinte:

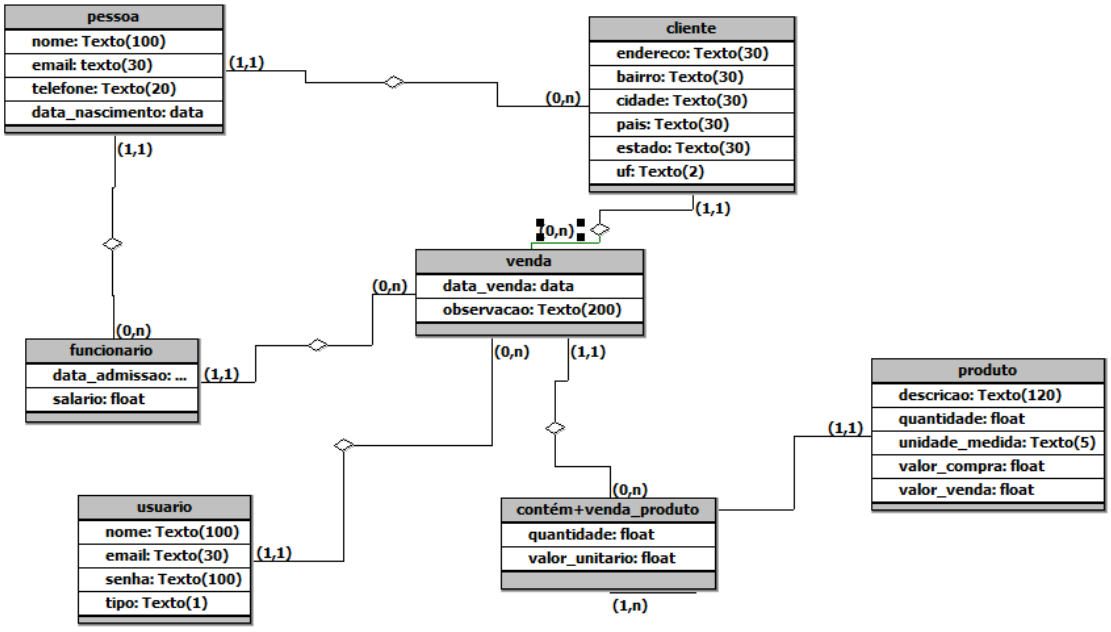

Imagem 3. Modelagem Lógica do Banco de dados

4. Tecnologias escolhidas

Para o desenvolvimento, decidi fazer em MVC (*Model, View e Controller*), uma padrão em que separa as telas do sistema (*views*) dos models, que fazem o contato direto com o banco, com comunicação entre os dois sendo feita através dos controladores (*Controllers*).

Para a linguagem eu escolhi o PHP 8.1 (https://www.php.net/), pois tenho familiaridade com o mesmo, para o banco de dados decidi escolher o MySQL 8.0 (https://www.mysql.com/) pelo mesmo motivo, familiaridade. No front-end juntei algumas tecnologias que eu conhecia para utilizar no software, são elas:

Git com Github (https://github.com/BevilaquaBruno/projeto_dev1) – Para gerenciar o controle de versão e publicar o código gerado.

Simples Mask Money (https://github.com/codermarcos/simple-mask-money) – Para utilizar nas máscaras de dinheiro e valores quebrados no sistema.

Axios (https://axios-http.com/ptbr/docs/intro) – Para realizar as requisições HTTP dos cadastros ou listagens necessárias.

Awesome Notifications (https://f3oall.github.io/awesome-notifications/docs/) – Para criar diálogos de notificação do que foi realizado ou para receber alguma confirmação do usuário.

PureCSS (https://purecss.io/) – Para estilizar tabelas, formulários, botões e inputs do software.

PureMask (https://romulobrasil.com/puremask-js/) – Máscara para campos como telefone, placas de carro, cpf e afins.

Vanilla Datatable (https://www.cssscript.com/lightweight-vanilla-data-table-component/) – Como o próprio nome diz, é um *datatable* com *javascript* puro.

PHP PDO – Para garantir a segurança das transações realizadas no banco de dados.

Para o controle de qual módulo o usuário estava acessando bem como de qual módulo ele queria os dados, criei um sistema que lê duas variáveis GET, o "m" que define o módulo e o "a" que define o action. Dessa forma, passando por exemplo *"m=pessoa&a=listar"* você seria direcionado a listagem de pessoas. E *"m-pessoa&a=cadastro"* você seria direcionado ao formulário de cadastro de pessoas.

5. Conclusão

O desenvolvimento de qualquer software, por mais simples que seja, envolve diversas etapas que, se puladas, podem causar graves problemas no futuro, olhando dessa perspectiva, acredito que esse projeto foi de grande aprendizado sobre o desenvolvimento de uma aplicação do zero.

Como foi visto anteriormente, o meu foco não foi na quantidade de campos em cada cadastro, mas sim deixá-los enxutos e conseguir terminar o projeto a tempo. Dessa forma cumpri o meu objetivo de terminar o projeto dentro do tempo previsto com tudo funcionando.

Durante esse projeto, tive a confirmação (pelo menos pessoal) de que o modelo MVC é um padrão de projeto muito bom e simples de ser implementado, além de deixar o projeto bem organizado e fácil de dar manutenção.

Consegui juntar diferentes tecnologias nesse projeto, que juntas fizeram todo o funcionamento do software, desde os cadastros até as vendas e os gráficos.

7. Referências

PHP Documentation, https://www.php.net/, Internet, 2022.

MySQL Documentation, https://www.mysql.com/, Internet, 2022.

Simples Mask Money, https://github.com/codermarcos/simple-mask-money, Internet, 2022.

Axios, https://axios-http.com/ptbr/docs/intro, Internet, 2022.

Awesome Notifications, https://f3oall.github.io/awesome-notifications/docs/, Internet, 2022.

PureCSS Documentation, https://purecss.io/, Internet, 2022.

PureMask, https://romulobrasil.com/puremask-js/, Internet, 2022.

Vanilla Datatable, https://www.cssscript.com/lightweight-vanilla-data-table-component/, Internet, 2022.

O que é Padrão MVC? Entenda arquitetura de Softwares, Vitor Zucher, Internet, 2020, https://www.lewagon.com/pt-BR/blog/o-que-e-padrao-mvc