Documento de Visão de Projeto

BookStore

Lucas Santos da Silva

Lucas Santos da Silva 1

Objetivo **3**

Necessidades de negócio **3**

Objetivo do projeto **3**

Escopo **3**

Requisitos **3**

Requisitos funcionais 3

Requisitos não-funcionais 4

Premissas **4**

Influência das partes interessadas (stakeholders) **4**

Representação arquitetural **5**

Estrutura MVC **5**

Representação da camada de negócio (Model) **6**

Casos de uso **7**

Implantação **8**

Objetivo

Esse documento trata da documentação de visão para o projeto de BookStore, cujo objetivo é alinhar as expectativas macro entre a software house e o cliente, apresentar uma visão inicial sobre o escopo do projeto, esboço de implementação e visão arquitetural do sistema

Necessidades de negócio

Um sistema de vendas online de livros se faz necessário para que seja possível aumentar o faturamento e retenção de clientes devido as inúmeras vantagens da internet

Objetivo do projeto

Desenvolver uma solução web capaz de:

- Realizar o gerenciamento de vendas de livros
- Realizar o gerenciamento de estoque de livros
- Realizar o gerenciamento de status de entregas dos livros vendidos

Escopo

Como critério de aceite para o sucesso do projecto BookStore deve ser entregue

- Um sistema de software hospedado em um servidor web
- Documentos relativos ao desenvolvimento do projeto

Requisitos

Aqui serão descritos os requisitos funcionais e não-funcionais do **GPS**.
*Obs.: por "**gerenciamento**", deve ser entendido o **cadastro, alteração, exibição e exclusão** de registros*

Requisitos funcionais

O sistema deve ser capaz de

- Cadastrar livro

- Ativar ou inativar um livro
- Inativar um livro de forma automática
- Alterar cadastro de livro
- Consultar livros cadastrados
- Gerenciar cadastro de clientes
- Ativar ou inativar um cliente
- Cadastrar endereços de entregas
- Cadastrar cartões de crédito
- Cadastrar carrinho de compras
- Definir quantidade de itens no carrinho de compras
- Realizar compra
- Calcular frete
- Selecionar endereço de entrega
- Selecionar forma de pagamento
- Finalizar compra
- Definir produtos como entregues
- Solicitação de trocas
- Autorização de trocas
- Confirmar recebimento de itens para troca
- Gerar cupom de trocas
- Gerenciar gerenciamento de estoque

Requisitos não-funcionais

- O sistema deve ser desenvolvido no framework Ruby on Rails
- A base de dados deve ser Postgres na versão 10.0 ou superior
- O sistema deve ser hospedado em servidor web acessível publicamente
- O sistema web deve ser compatível com os navegadores Google Chrome, Firefox, Safari, Microsoft Edge e Opera

Premissas

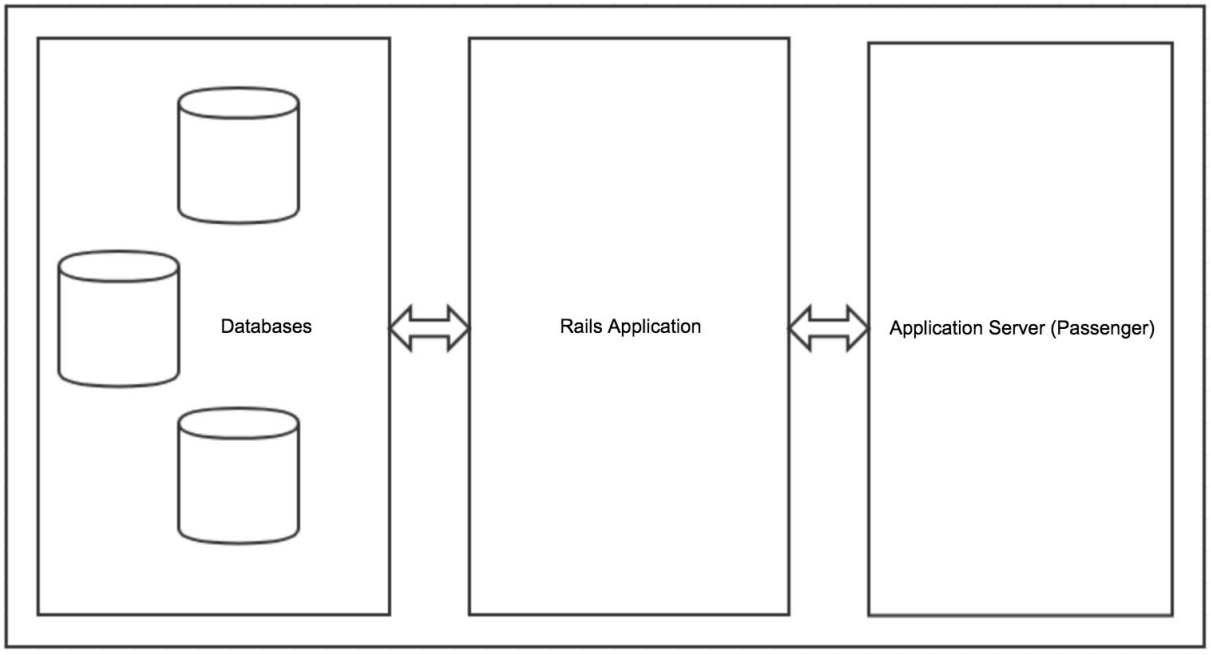
- O desenvolvimento do projeto será orientado pelo professor Rodrigo Rocha

Influência das partes interessadas (*stakeholders*)

- Lucas Santos: desenvolvedor e gerente de projetos
- Rodrigo Rocha: cliente

Representação arquitetural

O diagrama a seguir representa o modelo de implatação do sistema BookStore



- Application Server
  Contém o servidor de aplicação responsável por receber as requisições ao sistema
- Rails Application
  Contém o sistema sendo executado responsável por processar as solicitações do usuário, bem como executar as regras de negócio e outros gerenciamentos de entidades
- Database
  Container onde os dados são persistidos

Estrutura MVC

Uma aplicação Rails segue a estrutura MVC, onde a implementação do código segue três camadas lógicas: Model, View, Controller

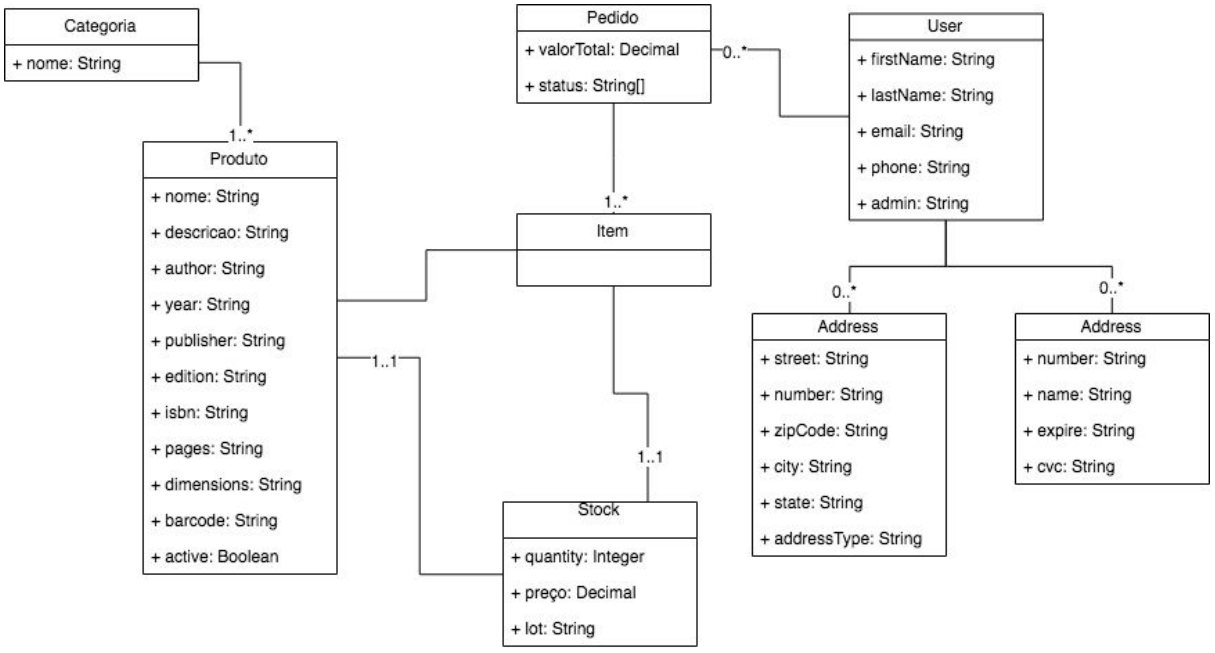
- Model
  A camada de modelo representa o modelo de domínio (como conta, produto, pessoa, postagem, etc.) e encapsula a lógica de negócios específica para seu aplicativo. Em Rails, as classes de modelo suportadas por banco de dados são derivadas de `ActiveRecord::base`. O Active Record permite que você apresente os dados das linhas do banco de dados como objetos e embeleze esses objetos de dados com métodos de lógica de negócios. Embora a maioria dos modelos do Rails seja suportada por um banco de dados, os modelos também podem ser classes

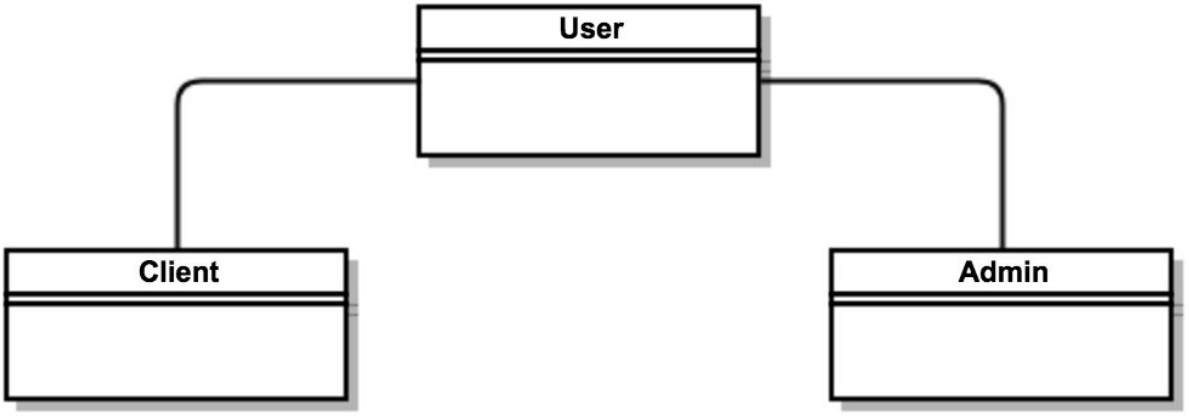
Ruby comuns ou classes Ruby que implementam um conjunto de interfaces, conforme fornecido pelo módulo modelo ativo.

- View
  A camada de exibição é composta de "templates" que são responsáveis por fornecer representações apropriadas dos recursos do seu aplicativo. Os templates podem vir em uma variedade de formatos, mas a maioria dos modelos de exibição são HTML com código Ruby incorporado (arquivos ERB). Templates são normalmente renderizados para gerar uma resposta do controlador, ou para gerar o corpo de um email. No Rails, a geração de templates é manipulada pela ActionView.

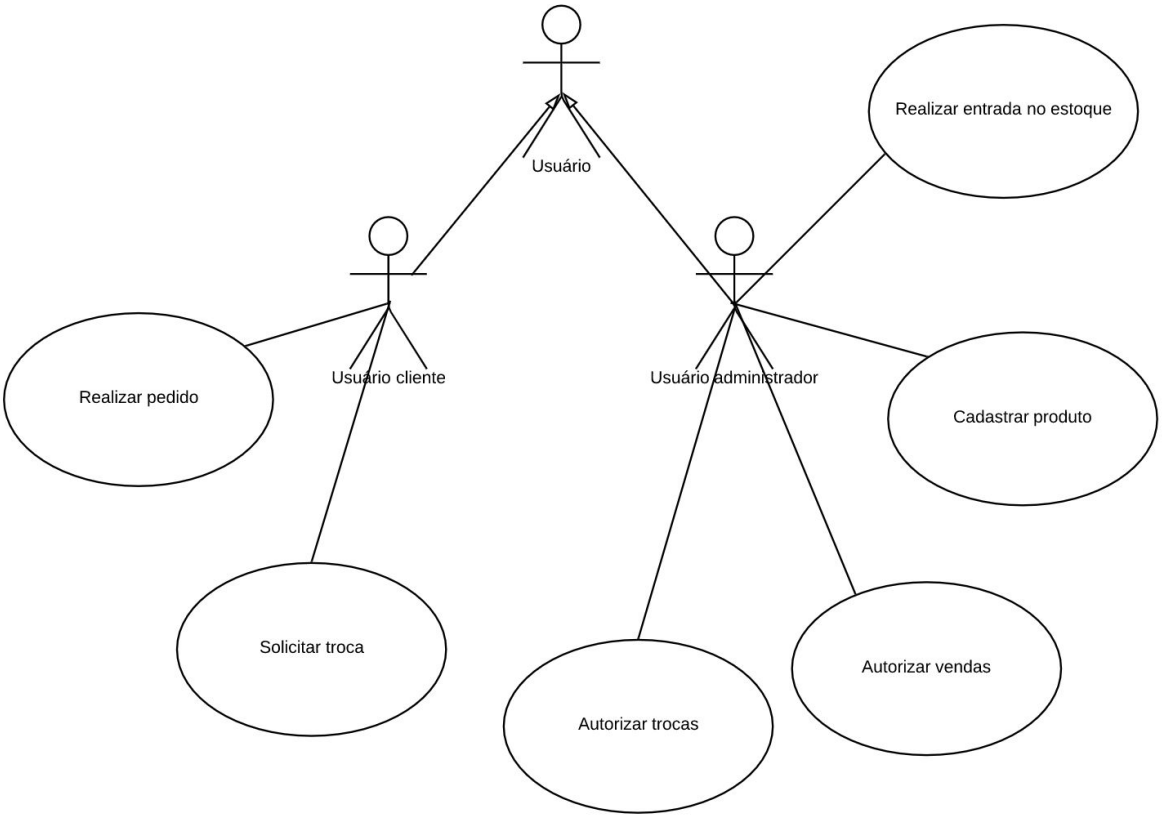
- Controller
  A camada Controller é responsável pelo manuseio de solicitações HTTP recebidas e pelo fornecimento de uma resposta adequada. Normalmente, isso significa retornar HTML, mas os controladores do Rails também podem gerar XML, JSON, PDFs, exibições específicas para dispositivos móveis e muito mais. Os controladores carregam e manipulam modelos e renderizam modelos de exibição para gerar a resposta HTTP apropriada. Em Rails, as solicitações de entrada são roteadas pelo Action Dispatch para um controlador apropriado, e as classes de controlador são derivadas de `ActionController::base`. Action Dispatch e Action Controller são empacotados juntos no Action Pack.

Representação da camada de negócio (Model)

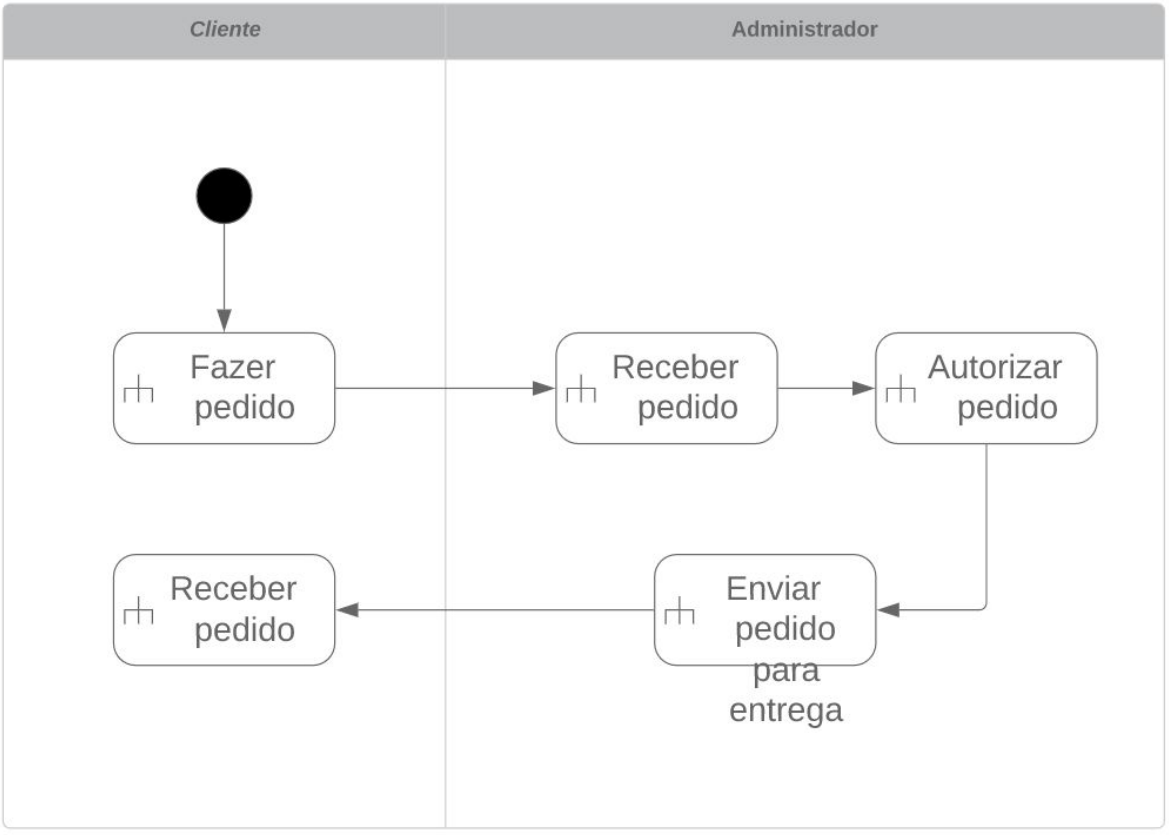


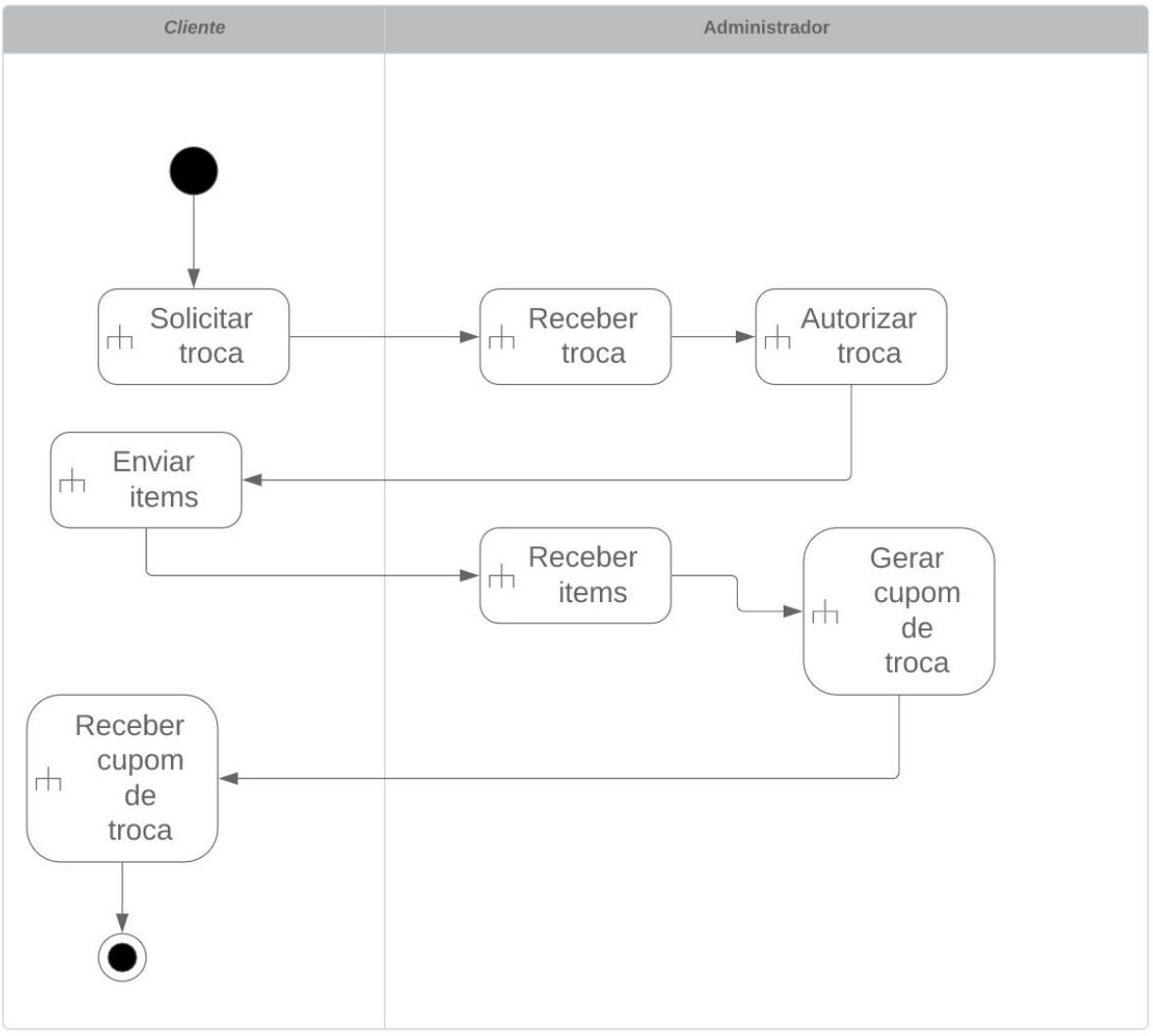


Casos de uso


Usuário
Realizar entrada no estoque
Usuário cliente
Usuário administrador
Realizar pedido
Cadastrar produto
Solicitar troca
Autorizar vendas
Autorizar trocas

Diagramas de caso de uso





Implantação

O sistema será implantado no Heroku, bem como seu banco de dados.