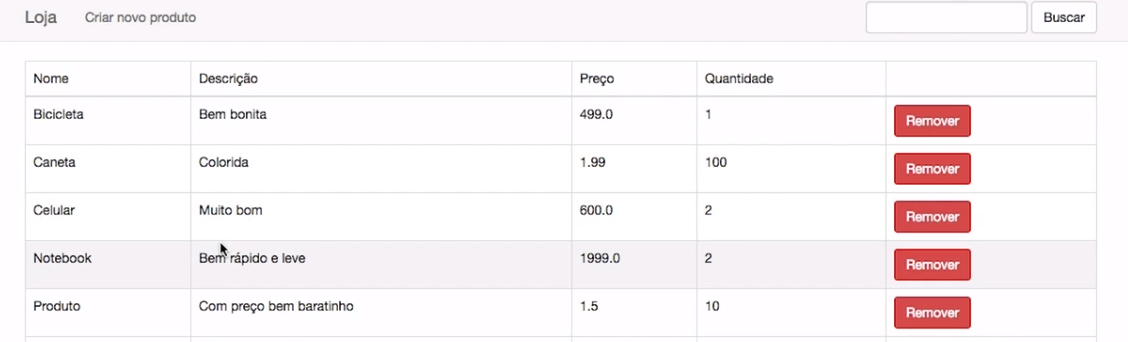
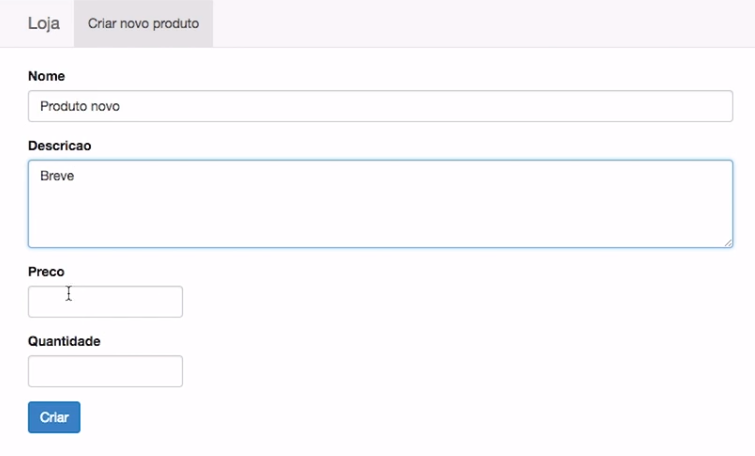
1. **Introdução**

**Transcrição**

Olá! Meu nome é Guilherme Lima, e neste curso de Rails 5 iremos lidar com a seguinte questão: temos uma uma loja on-line e queremos cadastrar todos os produtos disponíveis, de forma que o cliente possa visualizar quais são os itens disponíveis para venda. Teremos uma lista para mostrar todos os produtos, com a possibilidade de adicionar itens novos, bem como removê-los.



Na parte superior da tela, poderemos clicar sobre a opção "Criar Novo Produto", e seremos direcionados para um formulário que contém os campos "Nome", "Descrição", "Preço" e "Quantidade".



Ao criarmos um novo produto utilizando o formulário, ele estará visível na lista principal de produtos. Criaremos, ainda, uma categoria chamada "Produto com desconto", em que agruparemos os itens com menor preço. Como exemplo, criaremos um produto com o preço de 1.0, e ele será automaticamente agrupado na categoria de descontos, além de estar visível na lista geral de itens disponíveis.

Ao removermos esse produto novo, o navegador nos envia um pop-up perguntando se de fato queremos realizar a remoção. Ao clicarmos em "OK", outro produto com o preço menor em relação aos demais é agrupado na categoria "Produto com desconto", dessa forma teremos sempre em nossa loja algum item com desconto.

Criaremos também um campo de busca para que o usuário consiga encontrar itens com mais facilidade. Esse campo de busca ficará localizado na parte superior direita da tela. Uma vez que o cliente buscar um produto específico, serão exibidas as informações referentes a ele.

| **Nome** | **Descrição** | **Preço** | **Quantidade** |  |
| --- | --- | --- | --- | --- |
| Celular | Muito Bom | 600.0 | 2 | **Remover** |

Nesta primeira parte do curso, é essa estrutura que iremos montar. Vamos lá?

# 02 Preparando o ambiente

**Olá!**

Seja bem vindo ao treinamento de Ruby on Rails. Antes de iniciar nosso curso, vamos preparar o ambiente de desenvolvimento de acordo com seu sistema operacional.

**MAC**

* Homebrew nos permite instalar e compilar pacotes de software facilmente a partir da fonte. Para instalar, no terminal execute todo o comando com o link:

ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"

* Para controlar a versão do ruby que vamos utilizar, no terminal execute o comando:

brew install rbenv ruby-build

* Em seguida, vamos escolher a versão do ruby com os seguintes comandos (execute um de cada vez):

rbenv install 2.5.1

rbenv global 2.5.1

* Agora, vamos instalar o rails com o seguinte comando:

gem install rails -v 5.2.0

* Para podermos usar o executável Rails, precisamos informar ao rbenv:

rbenv rehash

* Vamos instalar o banco de dados padrão do Rails, o SQLite com o seguinte comando:

brew install sqlite3

**Windows**

* [Instale o ruby clicando neste link com o kit de desenvolvimento](https://github.com/oneclick/rubyinstaller2/releases/download/rubyinstaller-2.5.3-1/rubyinstaller-devkit-2.5.3-1-x64.exe).
* Após o download, execute o instalador, aceite o termo de uso e prossiga com a instalação.
* Execute o comando para instalar o rails, na versão 5.2.

gem install rails -v 5.2.0

* Execute o comando abaixo para instalar o bundler.

gem install bundler

* Agora vamos precisar do Nodejs. Caso você já tenha instalado, maravilha. Caso ainda não tenha, [acessando esse link você pode fazer o download do Nodejs](https://nodejs.org/en/).
* Para finalizar a instalação, instale o banco de dados padrão do rails, o SQLite com o seguinte comando:

gem install sqlite3

**Linux**

* No terminal do linux, atualize a lista de programas que podem ser baixados com o seguinte comando:

sudo apt-get update

* Vamos instalar o curl para transferir os dados:

sudo apt-get install curl

gpg --keyserver hkp://keys.gnupg.net --recv-keys 409B6B1796C275462A1703113804BB82D39DC0E

\curl -L https://get.rvm.io | bash -s stable --ruby

* Agora vamos instalar o Nodejs com o seguinte comando:

sudo apt-get install nodejs

* Algumas gems são necessárias para o funcionamento do rails:

sudo apt-get install build-essential patch ruby-dev zlib1g-dev liblzma-dev

* Para finalizar, vamos instalar o rails:

gem install rails --no-document

* Vamos instalar também o git:

sudo apt-get install git

# 03 Primeira rota

## Transcrição

Temos vários produtos diferentes, e a ideia é criarmos uma estrutura de e-commerce simples para que os usuários consigam acessar a página e visualizar todos os itens disponíveis para compra. Cada produto deve possuir um nome, descrição breve, preço e quantidade.

Criaremos, portanto, uma aplicação utilizando o Ruby on Rails. Na atividade desta aula, foi disponibilizado um passo a passo de como preparar o ambiente de trabalho que iremos utilizar. Para começar, acessaremos o terminal - que usaremos bastante ao desenvolver um projeto utilizando Rails.

Para iniciarmos o projeto, temos um comando específico que ativará o Rails, mas antes de utilizá-lo, queremos que o diretório do projeto fique salvo na área de trabalho, tornando mais fácil a manipulação dos arquivos. Para isso, usaremos o comando cd destktop.

Last login: Mon Sep 17 10:01:06 on ttys000

Alura-Verde:~ aluraverde$ cd desktop

Alura-Verde:desktop aluraverde$

Em seguida, usaremos o comando rails new, para que o Rails crie um novo projeto para nós. Assim feito, devemos nomear o projeto a ser criado, como loja.

Last login: Mon Sep 17 10:01:06 on ttys000

Alura-Verde:~ aluraverde$ cd desktop

Alura-Verde:desktop aluraverde$ rails new loja

Vários arquivos serão criados e durante o curso entenderemos melhor a utilidade de cada um deles. Não trabalharemos exclusivamente com o terminal, faremos uso também do Visual Studio Code, mas se você quiser utilizar outro editor similar, como Sublime ou Atom, não há problema algum.

Abriremos o Visual Studio e arrastaremos a pasta do projeto para dentro do editor. Feito isso, teremos acesso a uma série de conteúdos criados pelo Rails, e um desses arquivos é muito importante, o Gemfile, onde encontraremos todas as gems que estamos utilizando no projeto. Temos, por exemplo, a versão do Ruby, do Rails, qual banco de dados será usado, assim como o web server.

source 'https://rubygems.org'

git\_source(:github) {|repo| "https://github.com/#{repo}.git" }

ruby '2.5.1'

# Bundle adge Rails instead: gem 'reails', github: 'rails/reails'

gem 'reails', '~> 5.2.1'

# Use sqlite3 as the database for Active Record

gem 'sqlite3'

# Use Puma as the app server

gem 'puma', '~> 3.11'

# Use SCSS for stylesheets

gem 'sass-rails', '~> 5.0'

Nós podemos acessar os arquivos criados pelo Rails e visualizar o projeto no navegador, para isso usaremos o comando cd loja para entrarmos no diretório do projeto, em seguida acionaremos o rails server:

Alura-Verde:loja aluraverde$ rails server

=> Booting Puma

=> Rails 5.2.1 application starting in development

=> Run `rails server -h` for more startup options

Puma starting in single mode...

\* Version 3.12.0 (ruby 2.5.1-p57), codename: Llamas in Pajamas

\* Min threads: 5, max threads: 5

\* Environment: development

\* Listening on tcp://0.0.0.0:3000

Use Ctrl-C to stop

Vimos que é a Puma que disponibiliza a aplicação para o servidor, e o listening é ouvido na porta tcp://0.0.0.0:3000. E esse endereço que utilizaremos no navegador para visualizarmos o projeto. Teremos uma mensagem de boas-vindas, mas não é esse o conteúdo que deve ser exibido em uma loja web, devemos mostrar os produtos disponíveis e suas características.



De volta ao Visual Studio, clicaremos no diretório app - que armazenará todos os códigos referentes à aplicação - em seguida clicaremos com o botão direito sobre a pasta views e selecionaremos a opção "New File". Esse novo arquivo, por padrão, será chamado de index.html.

Criaremos um código HTML simples, construindo uma tabela que conterá as colunas de Nome, Descrição, Preço e Quantidade.

<html>

<body>

<table>

<thread>

<tr>

<td>Nome</td>

<td>Descrição</td>

<td>Preço</td>

<td>Quantidade</td>

</tr>

</thead>

</table>

</body>

</html>

Temos o cabeçalho pronto, resta criamos os produtos. O primeiro item será uma Camiseta, cuja descrição é Bem bonita, preço 30 e a quantidade, 10.

<html>

<body>

<table>

<thread>

<tr>

<td>Nome</td>

<td>Descrição</td>

<td>Preço</td>

<td>Quantidade</td>

</tr>

</thead>

<tbody>

<tr>

<td>Camiseta</td>

<td>Bem bonita</td>

<td>30</td>

<td>10</td>

</tr>

</tbody>

</table>

</body>

</html>

O segundo produto será uma Luva, e preencheremos as características como descrição, preço e quantidade.

<html>

<body>

<table>

<thread>

<tr>

<td>Nome</td>

<td>Descrição</td>

<td>Preço</td>

<td>Quantidade</td>

</tr>

</thead>

<tbody>

<tr>

<td>Camiseta</td>

<td>Bem bonita</td>

<td>30</td>

<td>10</td>

</tr>

<tr>

<td>Luva</td>

<td>Para os dias frios</td>

<td>39</td>

<td>5</td>

</tr>

</tbody>

</table>

</body>

</html>

Feito isso, poderemos acessar o navegador e visualizar os produtos na tabela que construímos. Digitaremos a URL localhost:3000/produtos. No entanto, teremos uma mensagem de erro do Rails:

Routing Error

No route matches [GET] "/produtos"

Rails.root: /Volumes/Alura Verde 2/aluraverde/Desktop/loja

Isso quer dizer que precisamos criar uma rota para conseguirmos acessar /produtos. Clicaremos sobre "Config > routes.rb", e nesse arquivo conseguiremos especificar quem atende a requisição enviada pelo navegador.

Rails.application.routes.draw do

# For details on the DSL avaible within this file, see

http://guides.rubyonrails.org/routing.html

end

Deletaremos os comentários sobre a documentação. Em seguida, especificaremos que quem deve atender o get "produtos" será produtos #index. O que estamos construindo é um **index de produtos**, para tornar nosso código mais organizado. Podemos construir um index de todas as categorias de produtos, cidades, inscrições, e assim por diante.

Rails.application.routes.draw do

get "produtos", to: "produtos#index"

end

Ao acessarmos a URL localhost:3000/produtos, teremos outra mensagem de erro:

Routing Error

uninitialized constant ProdutosController

Esse erro ocorreu porque toda a requisição necessita de um controle sobre o que deve ser exibido ou não, isto é, os dados que devem ser impressos na tela. Portanto, não basta apenas criarmos uma rota da requisição, precisamos também criar um elemento de controle. Esse erro ocorreu porque toda a requisição necessita de um controle sobre o que deve ser exibido ou não, isto é, os dados que devem ser impressos na tela. Portanto, não basta apenas criarmos uma rota da requisição, precisamos também de um **elemento de controle**.

O Rails nos ajudará nessa tarefa. No terminal, estamos executando o servidor. Ao acionarmos "Ctrl + C", o servidor será desligado e a página da loja não pode mais ser acessada no navegador. Para evitarmos esse problema, abriremos uma nova janela no terminal. Na primeira aba o servidor ficará no ar, e ele pode ser habilitado pelo comando rails s, e na outra janela escreveremos o código Ruby.

Primeiramente acessaremos nosso projeto, e em seguida usaremos o comando rails generate para gerar um controller de Produtos.

Alura-Verde:loja aluraverde$ rails generate controller Produtos

No Visual Studio, veremos que a nova pasta produtos é criada dentro de app. Essa nova pasta ainda está completamente vazia, e como havíamos criado um HTML para exibição de nossos produtos na pasta index.html, iremos movê-la para dentro de produtos. Ainda, será criada a classe produtos\_controller, localizada em "app > controllers". Essa classe herda algumas regras de ApplicationController que entenderemos posteriormente:

class ProdutosController < ApplicationController

end

De volta ao navegador, acessaremos o endereço localhost:3000/produtos. Não teremos mais a mensagem de erro, mas sim informações a respeito dos nossos dois produtos:

Nome Descrição Preço Quantidade Camiseta Bem bonita 30 10 Luva Para os dias frios 39 5

A ideia é cadastrarmos mais dois produtos. Já sabemos o procedimento: em index - agora armazenado na pasta produtos, criaremos mais tags <tr> e <td>. Trata-se de um processo bastante trabalhoso, e seria mais interessante que nossos produtos fossem cadastrados no banco de dados.

Vamos tornar essa ação possível nas próximas aulas.