



Plataformas de aplica  es Web

Aula 04 - Plataformas full-stack - Parte 1



Material Did tico do Instituto Metr pole Digital - IMD

Termo de uso

Os materiais did ticos aqui disponibilizados est o licenciados atrav s de Creative Commons **Atribui  o-SemDeriva  es-SemDerivados CC BY-NC-ND**. Voc  possui a permiss o para realizar o download e compartilhar, desde que atribua os cr ditos do autor. N o poder  alter -los e nem utiliza-los para fins comerciais.

Atribui  o-SemDeriva  es-SemDerivados

CC BY-NC-ND



<https://creativecommons.org/licenses/by-nc-nd/4.0/>

Apresentação

Plataformas full-stack, também chamadas de Frameworks full-stack são uma coleção de bibliotecas que fornecem facilidades para a criação de sistemas web em todos os níveis (daí o termo full-stack).

Isso significa que em uma plataforma full-stack você normalmente tem disponível funcionalidades de:

- Processamento de requisições pelo lado do servidor
- Roteamento
- Acesso a banco de dados
- Template para criação de páginas dinâmicas do lado do servidor
- Autenticação
- Log
- E várias outras facilidades

Normalmente essas ferramentas são maiores e demandam um maior tempo de estudo para que seja dominada. Iremos apresentar as características de algumas delas nesta disciplina para que você as conheça e, se desejar se aprofundar, saber por onde começar.

Alguns exemplos de plataformas full-stack são:

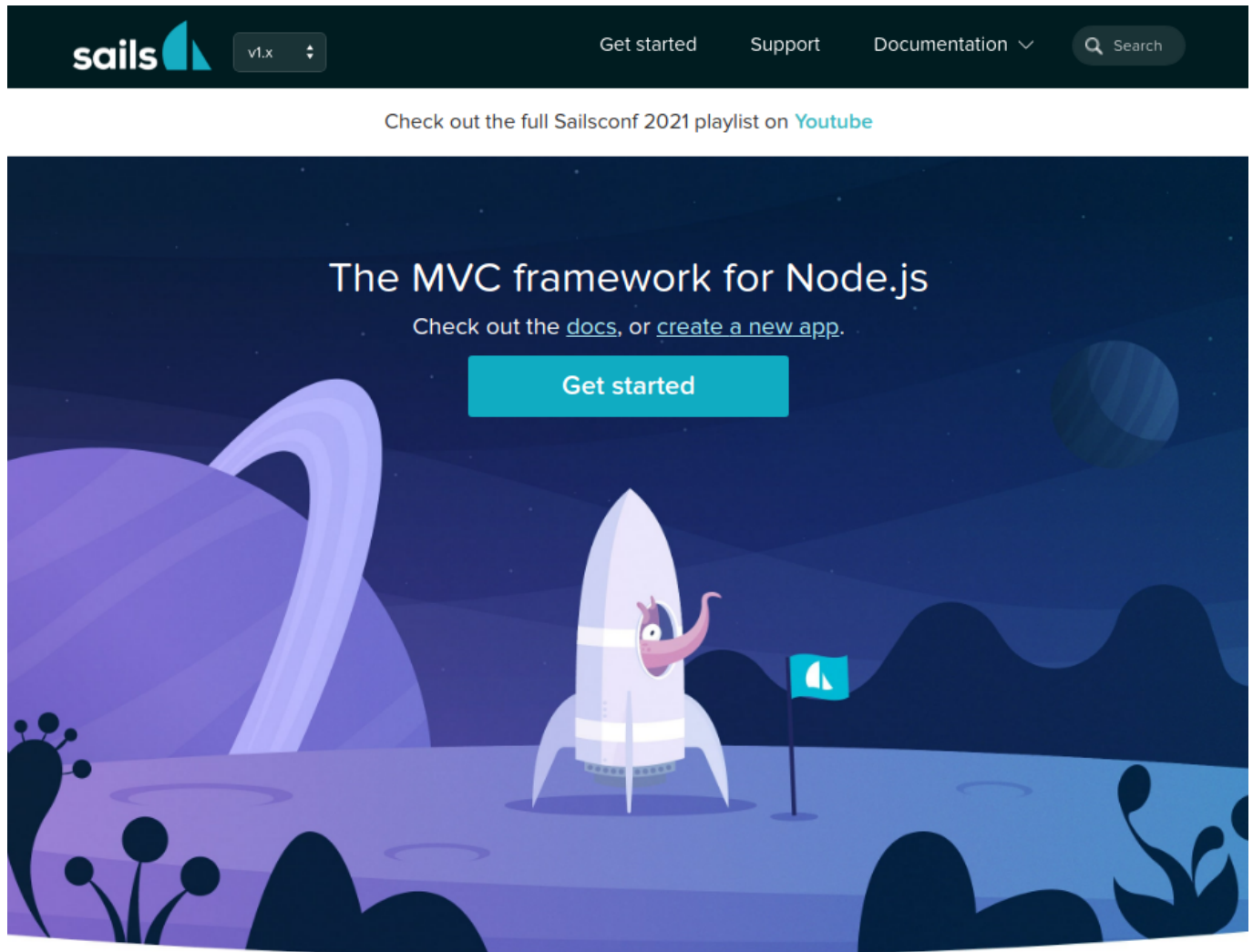
- SailsJS (<https://sailsjs.com/>)
- Ruby on Rails (<https://rubyonrails.org/>)
- Django (<https://www.djangoproject.com/>)
- Laravel (<https://laravel.com/>)

Cada uma das plataformas listadas acima utiliza linguagens de programação diferentes. O Sails, por exemplo, utiliza Javascript, O RubyOnRails Ruby, Django utiliza Python e o Laravel utiliza PHP.

Sails

Website: <https://sailsjs.com/>

O Sails é um framework full-stack muito popular que segue o padrão MVC (Model-View-Controller), assim como diversos outros, escrito em Javascript, é inspirado no popular framework Ruby on Rails, traz diversas funcionalidades muito úteis na criação de sistemas completos modernos e utiliza a linguagem Javascript.



Site inicial do Sails. Fonte: <https://sailsjs.com/>

Ele é totalmente feito em JavaScript, suporta vários banco de dados através do seu ORM (Object-Relational Mapper), vem com uma funcionalidade chamada blueprints, que ajuda a inicializar o back-end sem escrever código, possui um front-end compatível com qualquer framework front-end, integração facilitada com websocket, entre outras funcionalidades.

Utiliza fortemente a funcionalidade de geração automática de código para as rotas, controllers e views do seu sistema, porém você pode personalizar tudo de forma simples, mas é necessário conhecer as funcionalidades do framework para fazer isso.

Características do Sails

Totalmente em Javascript

Criar aplicações com o Sails significa que seu sistema será escrito 100% em Javascript, para quem gosta dessa linguagem de programação, essa é uma grande

vantagem.



100% JavaScript

O Javascript é uma linguagem muito popular no front-end e o Sails permite que você também utilize essa linguagem para a parte back-end da sua aplicação, tornando sua solução um sistema de linguagem única.

Qualquer banco de dados

O Sails vem junto com uma ferramenta ORM (Object-Relational Mapping) chamada Waterline, que fornece uma camada de acesso a banco de dados simples e poderosa e abstrai qual banco está sendo utilizado, requerendo somente uma simples configuração.



Any database

Suportando mais de 30 bancos de dados através de adaptadores criados pela comunidade, o Waterline pode ser utilizado em seu projeto de forma separada, mesmo que não seja um projeto Sails. O seu site é: <https://waterlinejs.org/> e para mais informações do uso dentro do Sails, veja:

<https://sailsjs.com/documentation/concepts/models-and-orm>

Geração automática de APIs REST

O Sails vem como uma ferramenta chamada blueprint que ajuda a iniciar o back-end da sua aplicação sem escrever muito código e de forma produtiva criar APIs REST com facilidade.



Auto-generated REST APIs

Os blueprints são compostos de duas funcionalidades principais: blueprint routes e blueprint actions. Elas são a base para a criação de APIs REST JSON no Sails todas as vezes que um model (classe que representa uma entidade na base de dados) e um controller (representam ações no seu sistema) é criado.

Por exemplo, você pode criar um model chamado Usuario.js, que é uma classe que representa um usuário no banco de dados. Através dos blueprints, isso automaticamente permite que você tenha disponível a URL `/usuario/create?nome=Maria`, criando um novo usuário no banco e também possa visitar o URL: `/usuarios` para obter a lista de usuário no formato JSON, que pode ser utilizado no seu front-end. Tudo isso sem escrever praticamente nenhum código.

Para mais informações: <https://sailsjs.com/documentation/concepts/blueprints>

Front-end Agnóstico

O Sails permite que você utilize o EJS no front-end por padrão, porém é possível utilizar qualquer outra ferramenta front-end que desejar.



Front-end agnostic

É possível, com o Sails, criar APIs e utilizá-las em frameworks web front-end como o Angular, React, Svelte, VueJS ou até em aplicativos desktop ou móveis Android e iOS.

Integração fácil com WebSocker

O uso de WebSockets em aplicações web é um tópico um pouco mais avançado

mas, em linhas gerais, com essa tecnologia você pode realizar interações através de mensagens de alta performance e baixa latência entre o front-end e back-end com uma conexão persistente, permitindo fluxos de dados contínuo entre cliente e servidor em tempo real.



Easy WebSocket integration

O Sails traz uma integração facilitada com WebSocket que não requer que você escreva um sistema extra para isso. O interpretador de requisições no Sails traduz as mensagens enviadas ao servidor para você, tornando-as compatíveis com cada rota do seu sistema.

Para mais informações sobre essa integração visite:

<https://sailsjs.com/documentation/reference/web-sockets/socket-client>

Criando um sistema com o Sails

Para se criar um sistema Sails, você precisa de um ambiente com o NodeJS instalado. Acesse o site <https://nodejs.org> e siga as instruções se ainda não tem esse ambiente na sua máquina.

Instale o Sails globalmente (opção -g do npm) no seu computador (necessita de poderes administrativos). Usa-se Linux para rodar o comando com o "sudo" na frente.

```
sudo npm install sails -g
```

```
[isaac@gru apps]$ sudo npm install sails -g
npm WARN deprecated uuid@3.0.1: Please upgrade to version 7 or higher. Older versions may
use Math.random() in certain circumstances, which is known to be problematic. See https://
/v8.dev/blog/math-random for details.

added 234 packages, and audited 235 packages in 4s

1 package is looking for funding
  run `npm fund` for details

3 low severity vulnerabilities

To address all issues, run:
  npm audit fix

Run `npm audit` for details.
[isaac@gru apps]$ |
```

Resultado da instalação do Sails no sistema operacional Linux. Fonte: Autor

Utilize o comando "sails" para criar uma nova aplicação, escolhendo a opção "Web App" (opção 1):

```
sails new lojavirtual
```

Escolha a opção 1 + ENTER

```
[isaac@gru apps]$ sails new lojavirtual
Choose a template for your new Sails app:
1. Web App    · Extensible project with auth, login, & password recovery
2. Empty      · An empty Sails app, yours to configure
(type "?" for help, or <CTRL+C> to cancel)
? 1
info: Installing dependencies...
Press CTRL+C to cancel.
(to skip this step in the future, use --fast)
info: Created a new Sails app `lojavirtual`!
[isaac@gru apps]$ |
```

Comando de criação de uma nova aplicação Sails. Fonte: Autor

Isso criará uma aplicação Sails novinha com o nome "lojavirtual" na pasta com o mesmo nome.

Entrando na pasta e verificando os arquivos existentes, note que se trata de uma aplicação NodeJS, porém com diversos arquivos e pastas já adicionadas.

```
[isaac@gru apps]$ cd lojavirtual/
[isaac@gru lojavirtual]$ ls
api      assets  Gruntfile.js  package.json      README.md  tasks
app.js   config  node_modules  package-lock.json  scripts    views
[isaac@gru lojavirtual]$ |
```

Nova aplicação Sails criada. Fonte: Autor

Sua aplicação foi criada. Vamos executá-la e acessá-la pelo navegador. Para executar o servidor com a aplicação, você precisa entrar na pasta da aplicação (se ainda não estiver lá) e executar o comando específico do sails para executar o servidor:

```
cd lojavirtual
sails lift
```

```
[isaac@gru apps]$ cd lojavirtual/
[isaac@gru lojavirtual]$ ls
api      assets  Gruntfile.js  package.json  README.md  tasks
app.js   config  node_modules  package-lock.json  scripts    views
[isaac@gru lojavirtual]$ sails lift

info: Starting app...

info: Initializing project hook... (`api/hooks/custom/`)
info: Initializing `apianalytics` hook... (requests to monitored routes will be logged!)
info: •• Auto-migrating... (alter)
info:   Hold tight, this could take a moment.
info: ✓ Auto-migration complete.

debug: Running v0 bootstrap script... (looks like this is the first time the bootstrap has run on this computer)
info:
info:   .-...-.
info:
info:   Sails      <|      .-...-.
info:   v1.4.4      \|
info:               /|. \
info:             /  ||  \
info:            /   ||   \
info:           /-==|/_--'
info:          /-----'
info:  -----
info:  -----
info:
info: Server lifted in `/home/isaac/Documents/imd/tecnico/2021.1/plataformas_de_aplicacoes_web_2021.1/apps/lojavirtual'
info: To shut down Sails, press <CTRL> + C at any time.
info: Read more at https://sailsjs.com/support.

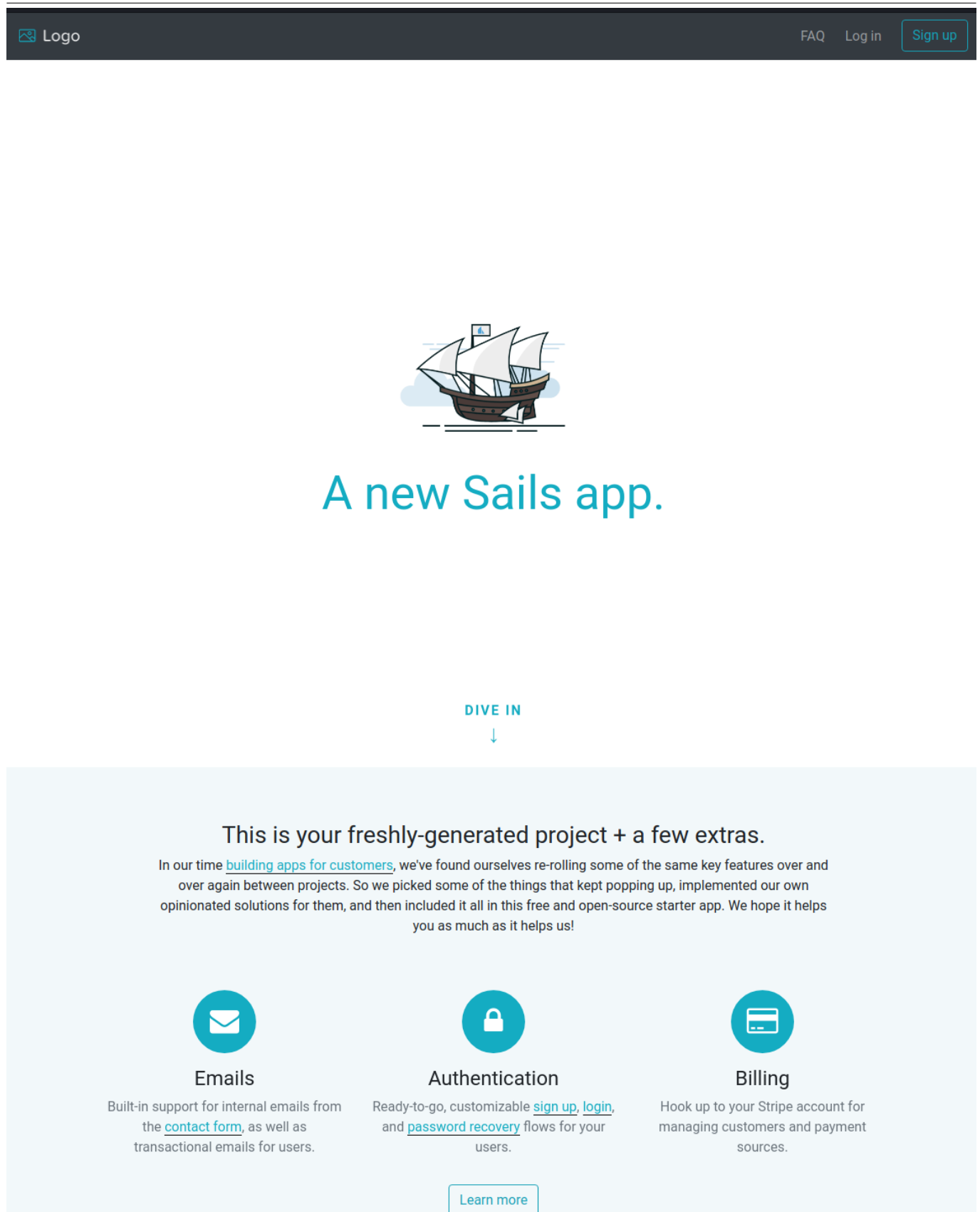
debug: -----
debug: :: Thu Jul 29 2021 20:02:43 GMT-0300 (Horário Padrão de Brasília)

debug: Environment : development
debug: Port        : 1337
debug: -----
```

Executando a aplicação criada com o Sails. Fonte: Autor

Veja que mensagens informando que você está executando uma aplicação Sails aparecerão no terminal, assim como a informação de que você está no ambiente development e a porta é a 1337 (padrão do Sails).

Abra um navegador e acesse <http://localhost:1337> para ver o resultado.



Nova aplicação Sails exibida no navegador. Fonte: Autor.

Repare que uma nova aplicação Sails criada com a opção "Web App", que foi a que escolhemos, já vem com bastante conteúdo e funcionalidades. Existe uma página pública iniciar com o conteúdo exibido na imagem acima, no menu você pode ver o espaço para uma logo, as opções FAQ, Login, Sign up são todas funcionais e exibem

uma página com um conteúdo estático (FAQ), Um página para login do usuário (Log in) e uma página de registro do usuário (Sign Up). Vamos clicar em "Sign up" e ver o que aparece:

Logo FAQ Log in Sign up

Create an account

Let's get started! Becoming a member is free and only takes a few minutes.

Full name

Isaac Franco Fernandes

Email address

isaacfranco@imd.ufrn.br

Choose a password

.....

Confirm password

.....

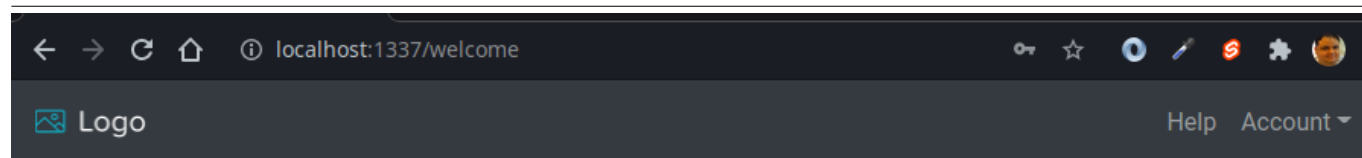
☐ I have read & agree to the [terms of service](#).

Create account

Have an account? [Sign in](#)

Página padrão de cadastro de usuário no Sails. Fonte: Autor.

Veja que já foi criada para você uma página de registro na aplicação com toda a lógica pronta. Vamos testar:



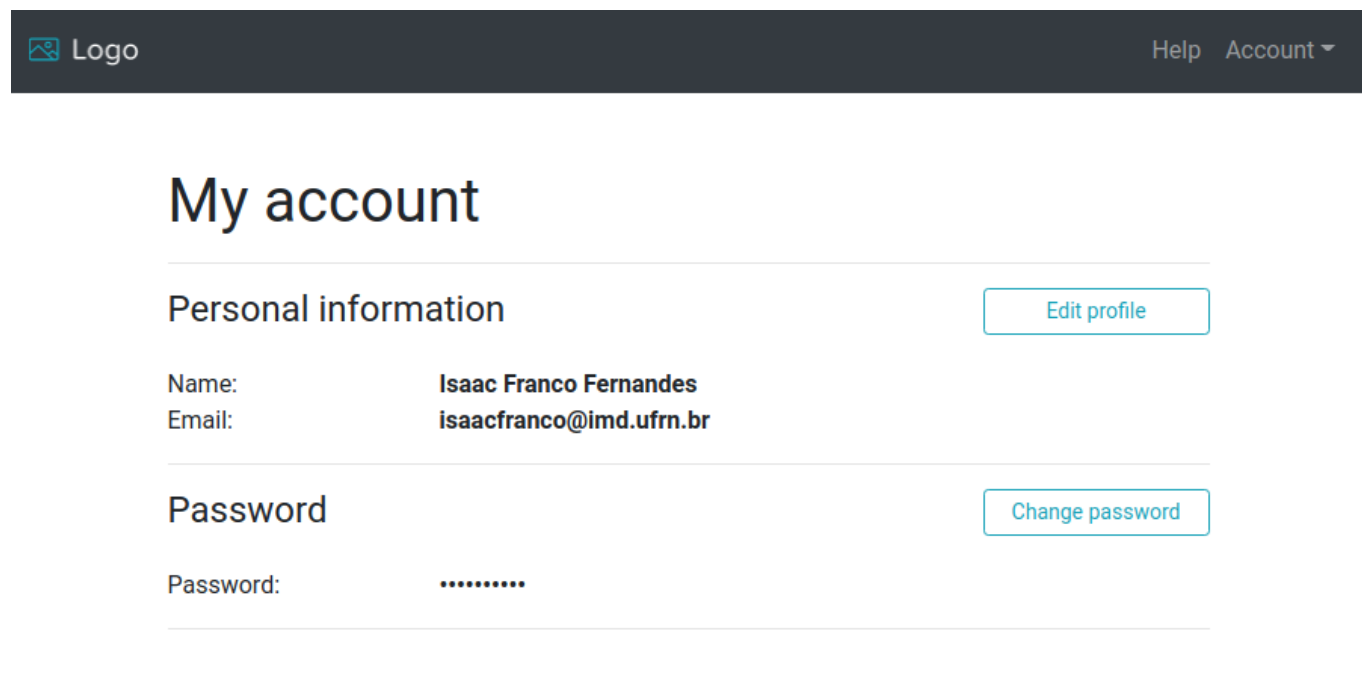
Welcome!

This is a page that only logged-in people can visit. Don't you feel special? Try clicking on a button below to do some things you can't do when you're logged out.

[Update my email](#)[Open a modal](#)

Página interna para usuários logados da aplicação Sails. Fonte: Autor

Criar usuário já funciona. Você, quando logado, pode acessar a opção no menu Account -> Settings e verá a seguinte página:

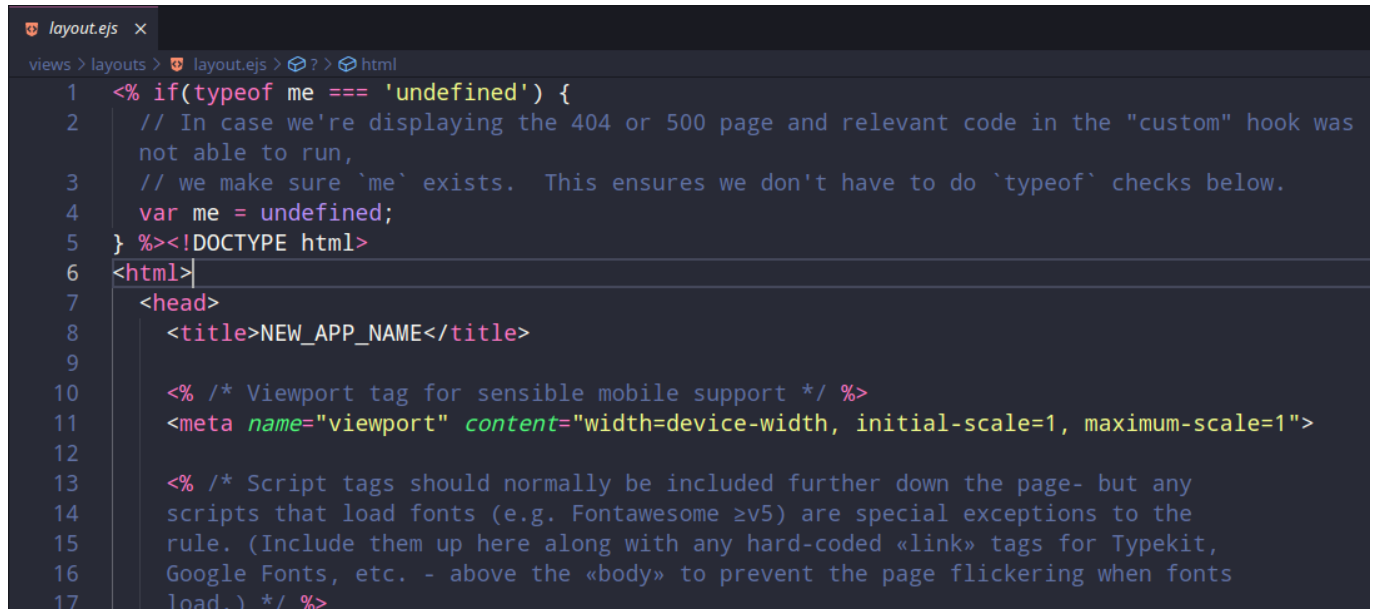


Página de configurações da conta no Sails. Fonte: Autor.

Nessa página você pode editar o seu perfil ("Edit profile"), trocando seu e-mail e nome, além de poder trocar a sua senha ("Change password").

Todas essas páginas e funcionalidades estão criadas na sua aplicação nova e você pode modificá-las livremente, assim como adicionar mais funcionalidades.

Como exercício, acesse a página EJS localizada em dentro da pasta lojavirtual em `views/layouts/layout.ejs` e verifique que se trata de um arquivo EJS praticamente padrão, que combina HTML, CSS e Javascript e ele é o layout principal que engloba todas as páginas internas. O título da janela da aplicação ("NEW_APP_NAME") pode ser alterado nesse arquivo, os textos dos links do menu também etc.



```
1 <% if(typeof me === 'undefined') {  
2   // In case we're displaying the 404 or 500 page and relevant code in the "custom" hook was  
3   // not able to run,  
4   // we make sure `me` exists. This ensures we don't have to do `typeof` checks below.  
5   var me = undefined;  
6 } %><!DOCTYPE html>  
7 <html>  
8   <head>  
9     <title>NEW_APP_NAME</title>  
10  
11     <% /* Viewport tag for sensible mobile support */ %>  
12     <meta name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1">  
13  
14     <% /* Script tags should normally be included further down the page- but any  
15     scripts that load fonts (e.g. Fontawesome ≥v5) are special exceptions to the  
16     rule. (Include them up here along with any hard-coded «link» tags for Typekit,  
17     Google Fonts, etc. - above the «body» to prevent the page flickering when fonts  
18     load.) */ %>
```

Início do arquivo de layout principal do Sails (views/layouts/layout.ejs). Fonte: Autor.

Veja no mesmo arquivo de layout que existe um trecho do código EJS que determina que itens de menu serão exibidos caso o usuário esteja logado (`<% if (me) %>`) ou caso contrário.

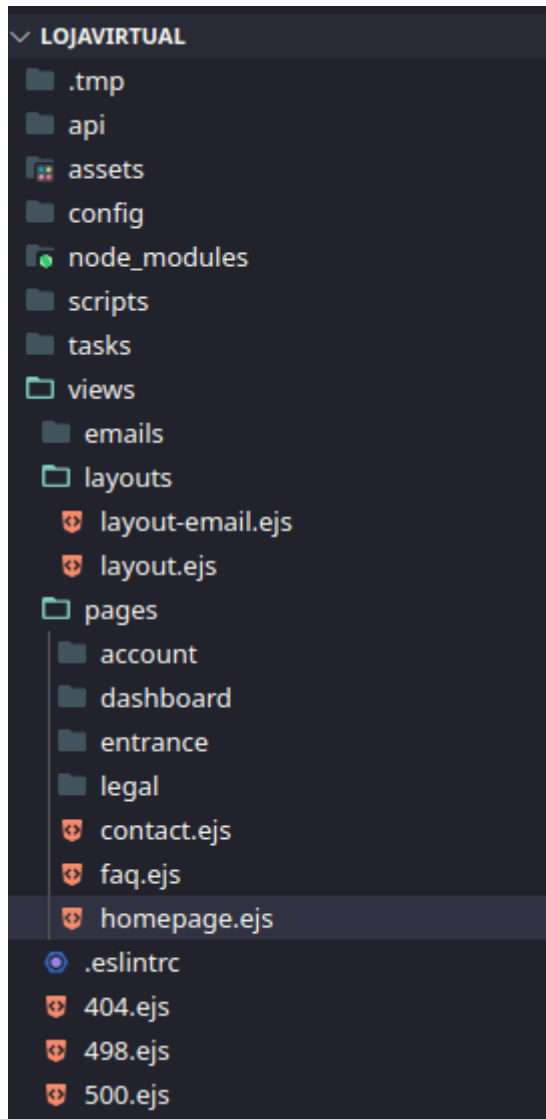
```

<!-- LOGGED-IN NAVIGATION -->
<% if(me) { %>
<a class="nav-item nav-link ml-2 ml-md-0 mr-2 mr-md-0" href="/contact">Help</a>
<!-- Only in desktop nav -->
<div class="nav-item dropdown d-none d-sm-block">
  <a class="nav-link dropdown-toggle" id="header-account-menu-link"
    data-toggle="dropdown" aria-haspopup="true" aria-expanded="false">Account</a>
  <div style="left: auto; right: 0;" class="dropdown-menu"
    aria-labelledby="header-account-menu-link">
    <a class="dropdown-item" href="/account">Settings</a>
    <a class="dropdown-item" href="/logout">Sign out</a>
  </div>
</div>
<!-- Only in mobile nav -->
<a class="nav-item nav-link ml-2 mr-2 d-block d-sm-none" href="/account">Account
Settings</a>
<a class="nav-item nav-link ml-2 mr-2 d-block d-sm-none" href="/logout">Sign out</a>
<% } else { %>
<!-- LOGGED-OUT NAVIGATION -->
<a class="nav-item nav-link ml-2 ml-md-0 mr-2" href="/faq">FAQ</a>
<a class="nav-item nav-link ml-2 ml-md-0 mr-2" href="/login">Log in</a>
<!-- Only in desktop nav -->
<div class="form-inline d-none ml-2 d-md-block" >
  <a class="btn btn-outline-info" href="/signup">Sign up</a>
</div>
<!-- Only in mobile nav -->
<a class="nav-item nav-link text-info ml-2 d-block d-md-none" href="/signup">Sign up</a>
<% } %>

```

Trecho do código EJS do layout principal com a definição de que itens de menu serão exibidos. Fonte: Autor.

O arquivo `views/pages/homepage.ejs` contém a página principal, que é exibida por padrão no centro do layout (substituindo o conteúdo de onde tem o trecho `<%-body %>`). Essa página vem com bastante conteúdo, mas a intenção é que você coloque o seu conteúdo aí.



Parte da estrutura de pastas de uma aplicação

Sails.

É possível também criar uma aplicação Sails sem todas essas funcionalidades já criadas para você, caso não precise ou deseje criar do zero. Basta escolher a opção 2 (Empty) quando for criar uma nova aplicação.

Acesse <https://sailsjs.com/documentation/concepts> para entender melhor os conceitos por trás do Sails e explore a documentação para aprender mais sobre o framework, a estrutura de pastas e arquivos, funcionalidades em geral e como trabalhar com ele de forma produtiva.

Ruby on Rails

Website: <https://rubyonrails.org/>

O Ruby on Rails é um framework full-stack escrito em na linguagem de programação Ruby que ganhou muita popularidade em meados dos anos 2000 por sua alta produtividade e hoje é utilizado em diversos sites e sistemas web como o Shopify, GitHub, Basecamp, Hulu, SlideShare, Groupon etc.

[Blog](#)[Guides](#)[API](#)[Forum](#)[Team](#)[Contribute](#)

Imagine what you could build if you learned Ruby on Rails...

Learning to build a modern web application is daunting. Ruby on Rails makes it much easier and more fun. It includes **everything you need** to build fantastic applications, and **you can learn it** with the support of **our large, friendly community**.



Latest version — Rails 6.1.4 released June 24, 2021

Web site do Ruby on Rails: Fonte: <https://rubyonrails.org/>

Também chamado somente de "Rails", esse framework utiliza o padrão MVC (Model-View-Controller) para estruturar sua aplicação, mas você não precisa ficar preso a suas convenções.

O Rails traz o conceito de DRY (Don't Repeat Yourself ou Não se repita, em português) como conceito central, trazendo um conjunto de convenções que se forem seguidas faz com que você precise configurar muito pouco para ter um sistema completamente funcional com muita produtividade.

Os criadores do Rails identificam o framework uma ferramenta com filosofias opostas ao minimalismo, o comparando analogamente com uma metrópole.

Trata-se de uma ferramenta grande, com muitas funcionalidades para aprender, mas se dominá-la serás muito produtivo(a).

Características do Ruby on Rails

Convenção sobre configuração

O Rails tem uma filosofia de que a maioria das configurações de um framework deve ser opcional e, caso não se configure nada, em uma determinada funcionalidade, ela deve se comportar de uma maneira bem determinada.

Por exemplo, no Rails é possível se criar models, que são classes que representam tabelas no banco de dados. Essas classes são escritas na linguagem de programação Ruby e devem herdar da classe `ApplicationModel`. Segue um exemplo (em Ruby):

```
class Produto < ApplicationRecord end
```

Nesse exemplo acima está se declarando uma classe chamada "Produto", herdada de `ApplicationModel`, classe interna do Ruby on Rails que tem funcionalidades de acesso a banco de dados. No exemplo, nenhuma outra configuração foi feita, nenhum construtor criado, nenhuma informação à qual tabela do banco essa classe está associada, nem em que máquina o banco de dados está. Tudo isso pode ser mudado, mas se você não informar nada, o Rail vai assumir algumas convenções, como, por exemplo que o nome da tabela será "produtos" (o nome da classe, em minúsculo e no plural), que o banco é o mesmo que está configurado por padrão na aplicação, que a tabela produtos tem uma chave primária chamada "id" que é um inteiro autoincremental, não é necessário informar quais são as outras colunas do banco de dados, pois quando a aplicação iniciar ele obtém essas informações do banco e já cria automaticamente os atributos na classe `Produto` com os mesmos nomes das colunas da tabela "produtos", e assim por diante. São convenções que podem sim ser mudadas, mas se você criar seu banco as seguindo vai ter um imenso ganho de produtividade.

Testes automatizados

O Ruby on Rails já disponibiliza um conjunto de ferramentas para testes automatizados da sua aplicação.

As ferramentas já disponíveis no framework são: MiniTest: Testes de unidade e avaliação de performance Capybara: Testes de sistema e interface Testes paralelos: Acelera os testes permitindo serem executados em paralelo.

Para mais informações sobre testes visite:

<https://guides.rubyonrails.org/testing.html>

Características do Ruby on Rails -

parte 2

Testes automatizados

O Ruby on Rails já disponibiliza um conjunto de ferramentas para testes automatizados da sua aplicação.

As ferramentas já disponíveis no framework são: MiniTest: Testes de unidade e avaliação de performance Capybara: Testes de sistema e interface Testes paralelos: acelera os testes permitindo serem executados em paralelo

Para mais informações sobre testes, visite:

<https://guides.rubyonrails.org/testing.html>

Internacionalização

O Rails tem um suporte forte para internacionalização de aplicações através da biblioteca Ruby I18n, permitindo que você crie facilmente sistemas que suportam múltiplas línguas de maneira organizada.

O processo de internacionalização significa que você pode criar um sistema que através de uma simples configuração pode ser executado com uma língua padrão diferente. Claro que é necessário que você informe quais são as traduções que você suporta, mas o Rails oferece uma forma organizada de você adicionar os textos do seu sistema em diferentes línguas em arquivos separados do HTML, o que torna o processo de tradução muito mais fácil.

Além da tradução, o Rails permite que as datas e formatos monetários sejam convertidos de maneira correta de acordo com as opções do usuário do sistema.

Por exemplo, o arquivo na sua aplicação config/locales/en.yml pode conter suas strings em inglês da seguinte forma:

```
en:
  welcome: Hello!
  bye: Goodbye
```

E outro arquivo chamado config/locales/pt-BR.yml pode conter as suas strings em português:

```
pt-BR:
  welcome: Olá!
  bye: Até logo
```

Para utilizar essas strings nas suas páginas ERB (algo similar ao EJS, porém utilizando o Ruby embutido no HTML), você deve executar o seguinte comando Ruby:

```
<%= I18n.translate 'welcome' %>
```

Ou

```
<%= I18n.translate 'bye' %>
```

E o Rails vai selecionar o texto final correto de acordo com a configuração da aplicação (que pode inclusive mudar de acordo com o usuário que está logado no sistema, por exemplo).

Existem muitas opções de internacionalização no Rails. Para descobrir mais, visite: <https://guides.rubyonrails.org/i18n.html>

Ruby on Rails - Bibliotecas

Algumas bibliotecas no Rails

Algumas bibliotecas que já estão presentes no Rails são:

- **Active Record:** ferramenta de ORM que permite acessar diversos bancos de dados diferentes, facilitando sua manipulação e consulta.
- **Action Mailer:** utilizado para enviar e-mails com conteúdo dinâmico para usuários
- **Action Cable:** utilizado para realizar mudanças em tempo real na sua página utilizando WebSocket

Outra biblioteca que não faz parte do framework Rails por padrão mas é muito popular é uma chamada "Devise".

O Devise foi criado pelo brasileiro José Valim e se tornou a solução mais popular de autenticação no Ruby on Rails (que não traz nenhuma robustez por padrão).

O devise é uma biblioteca completa, com 10 módulos de autenticação que podem ser utilizados no seu sistema Rails, são eles:

- **Database Authenticatable:** guarda senhas criptografadas no banco de dados para validar a autenticidade do usuário no login.
- **Omniauthable:** adiciona suporte à biblioteca OmniAuth(<https://github.com/omniauth/omniauth>), que permite login com plataformas como Facebook, Google etc.

- **Confirmable:** envia e-mail com instruções de confirmação para verificar se uma conta criada já existe e só é desbloqueada quando o usuário clica no link enviado por e-mail.
- **Recoverable:** reseta a senha do usuário com um link enviado por e-mail com instruções.
- **Registerable:** permite que o usuário se registre sozinho no sistema, com as opções de editar o perfil e remover a conta.
- **Rememberable:** permite ao usuário optar pela opção do navegador "se lembrar" que ele já logou no sistema anteriormente.
- **Trackable:** registra um contador de logins por conta, com a hora e o endereço IP de origem.
- **Timeoutable:** expira uma sessão que não está ativa por um determinado período de tempo.
- **Validatable:** provê validação de formato de e-mail e senha de acordo com as configurações do usuário.
- **Lockable:** trava automaticamente contas com múltiplas tentativas de login dentro de um período curto de tempo. Pode destravar a conta por e-mail ou depois de um determinado tempo automaticamente.

Como se pode observar, a utilização do Devise em uma aplicação Rails permite que se ganhe muita produtividade com todos esses recursos pronto, só precisando ser minimamente configurados em alguns minutos.

Para mais informações sobre o devise visite: <https://github.com/heartcombo/devise>

Conclusão

Foi apresentada na aula somente uma visão geral de algumas características de duas plataformas de desenvolvimento de aplicações full-stack. Essas ferramentas são geralmente muito grandes, com muitos recursos e seu aprendizado pode levar bastante tempo, porém tem um grande valor pois proporcionam um arsenal imenso na criação de sistemas complexos com muitos recursos que sua criação sem essas plataformas seria muito demorada.

Seria necessário um curso inteiro para exhibir mais a fundo as funcionalidades em detalhes de qualquer uma dessas plataformas.

Vale a pena estudar mais a fundo as plataformas citadas, e inclusive outras que desejar, fazer os tutoriais sugeridos nas suas páginas e tentar escolher uma delas para se aprofundar.