



# Plataformas de aplicações Web

## Aula 01 - Tipos de plataformas de aplicações web



Material Didático do Instituto Metrôpole Digital - IMD

### Termo de uso

Os materiais didáticos aqui disponibilizados estão licenciados através de Creative Commons **Atribuição-SemDerivações-SemDerivados CC BY-NC-ND**. Você possui a permissão para realizar o download e compartilhar, desde que atribua os créditos do autor. Não poderá alterá-los e nem utiliza-los para fins comerciais.

Atribuição-SemDerivações-SemDerivados

CC BY-NC-ND



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

# Apresentação

Como um desenvolvedor(a) de aplicações web, é muito importante se ter uma visão ampla das existentes plataformas de desenvolvimento, quais são suas características, que tipo de tecnologias elas utilizam, qual a sua arquitetura, aplicação, vantagens e desvantagens em diferentes cenários.

Ao longo dos anos, muitas opções como plataforma de aplicações web surgiram e tiveram altos e baixos em termos de popularidade. Nesta disciplina, vamos ter uma visão geral de algumas dessas plataformas, nos aprofundando mais à frente em algumas delas.

## Tipos de plataformas de aplicações web

É comum se categorizar de maneira mais simples as plataformas de aplicações web nos seguintes tipos:

- Bibliotecas de componentes UI
- Plataformas full-stack
- Plataformas back-end
- Plataformas front-end reativas

Nesta disciplina, veremos exemplos de diversas plataformas nessas categorias e suas características, entretanto é importante observar que uma aplicação web completa também contempla outros elementos como Bibliotecas de componentes UI, Bibliotecas de acesso a bancos de dados, Soluções de armazenamento de arquivos enviados para sua aplicação, etc.

Algumas plataformas back-end, por exemplo, não contemplam acesso a banco de dados e cabe a você programador(a) escolher uma. Outras plataformas de aplicações front-end reativas (Como o React, VueJS, Svelte etc.) te ajudam a tornar suas páginas mais dinâmicas, mas não trazem nenhum componente CSS para estilizar suas páginas e também cabe a você escolher algum ou então criar o seu próprio com CSS.

## Bibliotecas de componentes UI

Também chamadas chamadas (em inglês) de: UI Library, Web components library, Web UI, Libraries, Web UI Kits, UI Components Libraries, CSS Frameworks etc. as bibliotecas de componentes UI são normalmente uma coleção de recursos CSS e até Javascript que facilitam a criação de HTML estilizado com maior facilidade.

Podem ser utilizadas para se criar qualquer documento HTML e se integram com qualquer tipo de plataforma, seja ela full-stack, back-end ou front-end.

Cada um deles tem uma coleção de estilos diferentes para seus elementos, mas oferecem possibilidade de customização. Alguns exemplos são Bootstrap, Bulma e Materialize.

Email address

We'll never share your email with anyone else.

Password

☐ Check me out

Submit

Copy

```
<form>
  <div class="mb-3">
    <label for="exampleInputEmail1" class="form-label">Email address</label>
    <input type="email" class="form-control" id="exampleInputEmail1" aria-describedby="ema
    <div id="emailHelp" class="form-text">We'll never share your email with anyone else.</
  </div>
  <div class="mb-3">
    <label for="exampleInputPassword1" class="form-label">Password</label>
    <input type="password" class="form-control" id="exampleInputPassword1">
  </div>
  <div class="mb-3 form-check">
    <input type="checkbox" class="form-check-input" id="exampleCheck1">
    <label class="form-check-label" for="exampleCheck1">Check me out</label>
  </div>
  <button type="submit" class="btn btn-primary">Submit</button>
</form>
```

Formulário criado com HTML + Bootstrap. Fonte:

<https://getbootstrap.com/docs/5.0/forms/overview/>

```


<form action="#">
  <p>
    <label>
      <input type="checkbox" />
      <span>Red</span>
    </label>
  </p>
  <p>
    <label>
      <input type="checkbox" checked="checked" />
      <span>Yellow</span>
    </label>
  </p>
  <p>
    <label>
      <input type="checkbox" class="filled-in" checked="checked" />
      <span>Filled in</span>
    </label>
  </p>
  <p>
    <label>
      <input id="indeterminate-checkbox" type="checkbox" />
      <span>Indeterminate Style</span>
    </label>
  </p>
  <p>
    <label>
      <input type="checkbox" checked="checked" disabled="disabled" />
      <span>Green</span>
    </label>
  </p>
  <p>
    <label>
      <input type="checkbox" disabled="disabled" />
      <span>Brown</span>
    </label>
  </p>
</form>


```



- ☐ Red
- ☒ Yellow
- ☒ Filled in
- ☐ Indeterminate Style
- ☒ Green
- ☐ Brown

Componentes de formulário do tipo checkbox criados com o Materialize. Fonte: <https://materializecss.com/checkboxes.html>

 Email 

 Password

Login

```
HTML
<div class="field">
  <p class="control has-icons-left has-icons-right">
    <input class="input" type="email" placeholder="Email">
    <span class="icon is-small is-left">
      <i class="fas fa-envelope"></i>
    </span>
    <span class="icon is-small is-right">
      <i class="fas fa-check"></i>
    </span>
  </p>
</div>
<div class="field">
  <p class="control has-icons-left">
    <input class="input" type="password" placeholder="Password">
    <span class="icon is-small is-left">
      <i class="fas fa-lock"></i>
    </span>
  </p>
</div>
<div class="field">
  <p class="control">
    <button class="button is-success">
      Login
    </button>
  </p>
</div>
```

For  
mulário de login com o Bulma. Fonte: <https://bulma.io/documentation/form/general/>

As bibliotecas de componentes UI visam reduzir o trabalho de quem está desenvolvendo interfaces HTML, mas é importante conhecê-las bem para que sejam utilizadas de forma que realmente se obtenha o resultado final desejado e, principalmente, para que se tenha produtividade no processo. Veremos mais à frente outros detalhes sobre algumas dessas bibliotecas.

# Plataformas full-stack


As plataformas de aplicações consideradas full-stack normalmente são uma grande coleção de tecnologias combinadas em um único framework, com soluções já escolhidas por padrão para que você consiga criar sistemas completos sem precisar se preocupar em decidir que tecnologias utilizar para cada parte da aplicação.

elas normalmente são criadas ao redor de uma ou mais linguagens de programação específica e te oferecem uma forma de criar uma aplicação "esqueleto" já configurada para se utilizar um servidor web específico para desenvolvimento, normalmente vêm com alguma biblioteca de acesso a banco de dados, sistema de template para criação de HTML dinâmico do lado do servidor (e algumas do lado do cliente também), podem já ter soluções avançadas para envio de e-mail, websocket, alertas, etc.

A vantagem de utilizar uma plataforma full-stack é o fato de as tecnologias escolhidas na plataforma serem muito bem integradas e a quantidade de configuração necessária para se ter um sistema funcional ser menor.

As maiores desvantagens são o fato de seu sistema ficar normalmente maior que o necessário e ser necessário um maior tempo de aprendizado para se conhecer todo o funcionamento da plataforma.

Alguns exemplos são o Sails(Javascript), AdonisJS(Typescript), RubyOnRails(Ruby), Laravel(PHP), dentre outras.

 Logo

[FAQ](#) [Log in](#) [Sign up](#)

# Sign in to your account

☒ Remember me

Sign in

[Forgot your password?](#)

[Contact us](#) [Terms of use](#) [Privacy policy](#)

Copyright © 2021 [NEW\\_APP\\_COMPANY\\_NAME](#). All rights reserved.

Sistema base gerado com o SailsJS já com autenticação implementada. Fonte: Autor

The diagram illustrates the process of adding a new product. On the left, a form titled 'New Product' contains the following fields: 'Nome' (Name) with the value 'Brinquedo para cachorro', 'Descricao' (Description) with the value 'Crinquedo bem bacana que pode ser', 'Disponivel' (Available) checked, and 'Preco' (Price) with the value '25,30'. A 'Create Product' button is at the bottom. An arrow points from this form to a confirmation page on the right. The confirmation page displays a green message: 'Product was successfully created.' Below this, the product details are listed: 'Nome: Brinquedo para cachorro', 'Descricao: Crinquedo bem bacana que pode ser usado de forma', 'Disponivel: true', and 'Preco: 25.0'. At the bottom of the confirmation page are links for 'Edit' and 'Back'.

*Telas de adicionar e visualizar um produto em um sistema gerado com o RubyOnRails e utilizando o recurso de scaffold para gerar automaticamente as páginas de criar/listar/detalhar/editar/remover produtos.*

As plataformas full-stack, mesmo com uma série de configurações e escolhas de tecnologias prontas, podem ser altamente configuráveis e você pode alterar o comportamento delas, as deixando com a sua cara e utilizando suas tecnologias favoritas.

## Plataformas back-end

Na disciplina, estamos chamando de plataformas back-end aquelas que oferecem somente as funcionalidades mínimas para que uma aplicação web seja criada no lado do servidor e que deixa a princípio o programador(a) mais livre para adicionar recursos de acesso a banco de dados, geração de HTML dinâmico etc.

Lembre que todas as plataformas full-stack são também plataformas back-end, mas nem toda plataforma back-end é full-stack.

Também existem ao redor de uma linguagem de programação específica e cabe ao programador(a) que deseja utilizá-la, conhecer essa linguagem minimamente para que possa utilizar.

Alguns exemplos são o ExpressJS (Javascript), Sinatra(Ruby), Spring Boot(Java), Flask(Python) e Slim(PHP).

Algumas dessas plataformas também se denominam micro-frameworks back-end, ou seja, frameworks mínimos para que você comece a desenvolver sua solução, somente fornecendo uma forma de se redirecionar requisições HTTP para ações, que as processam e retornam um resultado.



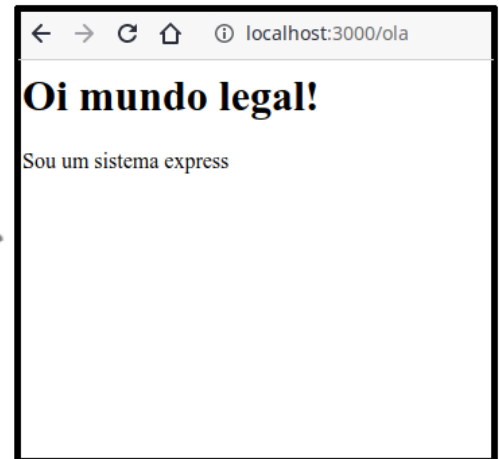
Elas têm como vantagem o fato de serem mais enxutas que as full-stack, mas como desvantagem podemos citar a necessidade do programador(a) de adicionar mais bibliotecas externas para que o sistema fique completo.

Na primeira imagem abaixo, você pode ver a única tela de um sistema criado com o ExpressJS com uma rota que responde a "/ola" e retorna um HTML específico. Já na segunda imagem, você tem o mesmo sistema, porém criado na linguagem Ruby utilizando a plataforma back-end Sinatra.

```
const express = require('express');
const app = express();

app.get('/ola', (req, res) => {
  let html = "<html><body>";
  html += "<h1>Oi mundo legal!</h1>";
  html += "<p>Sou um sistema express</p>";
  html += "</body></html>";
  res.send(html);
});

app.listen(3000, () => {
  console.log('Sistema iniciou na porta 3000');
});
```



Sistema mínimo criado com o ExpressJS. Fonte: Autor

```
require 'sinatra'

get '/ola' do
  html = "<html><body>" \
    "<h1>Oi mundo legal!</h1>" \
    "<p>Sou um sistema sinatra</p>" \
    "</body></html>"
  return html
end
```



Sistema mínimo criado com o Sinatra. Fonte: Autor

É possível gerar um sistema inicial um pouco mais completo com o ExpressJS já com log, EJS, etc, mas também é possível criar facilmente um sistema mínimo como esse, porém, quando o sistema vai crescendo geralmente é necessário se adicionar mais bibliotecas com outras funcionalidades. Da mesma forma, com o Sinatra, é possível se adicionar bibliotecas de acesso a banco de dados, processamento de imagem etc., para se criar um sistema realmente completo, porém isso cabe a você programador(a).