

## Código-fonte

O Sass foi definido como método de estruturação do código fonte para o projeto. Esta é uma linguagem pré-processador de alta produtividade baseada em CSS que pode ser compilada, resultando no CSS que será interpretado pelos navegadores web.

## Automatizando tarefas

O desenvolvimento do projeto aqui proposto é baseado na automatização de tarefas, possibilitando um desenvolvimento ágil e viabilizando a manutenção do código para futuros ajustes ou atualizações. As tarefas automatizadas são: codificação, compilação e compressão de CSS; remoção das marcações de comentários no código fonte quando compilado; criação de arquivos em versões minificadas do código fonte quando compilado; e sincronização de navegadores em diversos dispositivos para testes em tempo real.

## Implementação da automatização de tarefas

Em princípio é utilizado o *Javascript* para implementar o *Node Package Manager (NPM)* a fim de automatizar todo o processo de desenvolvimento. O NPM é um gerenciador de pacotes e utilitário de linha de comando que consiste em um repositório online para publicação de projetos de código aberto para *Node.js*.

Esses pacotes são um conjunto de códigos reutilizáveis que resolvem problemas específicos em diferentes tipos aplicação para web. Dentre todos os pacotes disponíveis existem os que podem ser utilizados no *front-end*. No desenvolvimento do projeto aqui proposto são utilizados os seguintes pacotes: *Gulp*, *Gulp sass*, *Gulp rename* e *Gulp strip css comments*.

## Pacotes do NPM utilizados

**Gulp:** O *Gulp* é um conjunto de ferramentas que auxiliam a automatização de tarefas que são recorrentes ou demoradas no fluxo de trabalho do desenvolvimento. Neste projeto é utilizado em sua versão 3.9.1.

**Gulp sass:** O *Gulp sass* é o pacote do *Gulp* responsável pela compilação do Sass e compressão do CSS. Neste projeto é utilizado em sua versão 3.1.0.

**Gulp rename:** É o pacote responsável por renomear arquivos. Neste projeto é utilizado em sua versão 1.2.2.

**Gulp strip css comments:** É o pacote do *Gulp* responsável por remover as marcações de comentários dos arquivos de CSS. Neste projeto é utilizado em sua versão 1.2.0

## Estruturas de arquivos

Ao obter ferramenta aqui proposta através de download em <https://goo.gl/brhyfn>, teremos como pasta principal **Editorial Framework 1.0.0** que comporta os arquivos de configuração, os arquivos Sass e, por fim, o arquivo de CSS compilado e pronto para ser utilizado em novos projetos, como segue na figura 1.

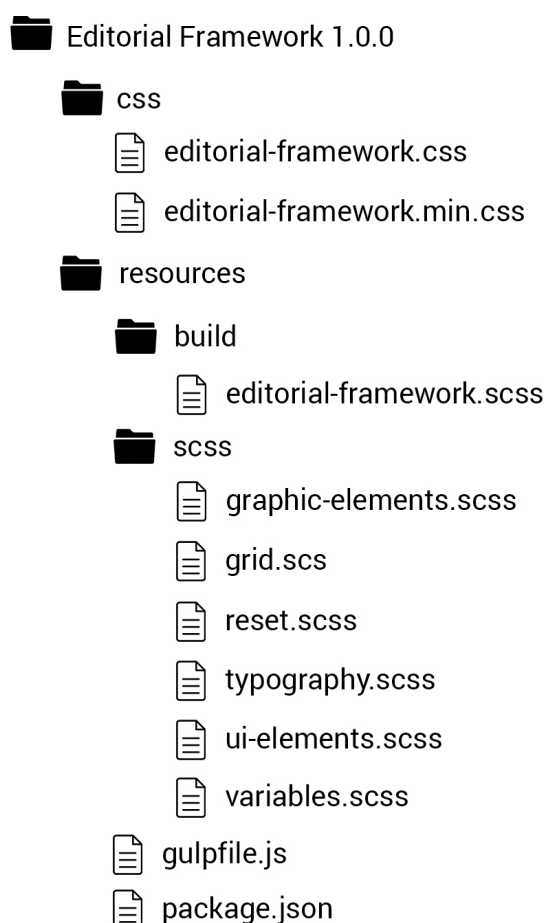


Figura 1: Estrutura de arquivos

Na pasta **css** temos o arquivo *editorial-framework.css*, esse é o resultado final que é compilado e o arquivo que deve ser utilizado no desenvolvimento de novos

websites, a pasta ainda contém o arquivo em sua versão minificada, *editorial-framework.min.css*.

Em **resources** se encontram os recursos necessários que o projeto utiliza para compilar o arquivo *editorial-framework.css*, os recursos estão organizados nas pastas **scss**, **build** e os arquivos de configuração *gulpfile.js* e *package.json*.

Em *gulpfile.js* são criadas as tarefas de automatização a serem executadas e a partir desse arquivo é feito a compilação do código fonte. Abaixo segue as tarefas criadas para atender as necessidades desse projeto.

```
1
2 // Variáveis
3 const gulp = require('gulp'),
4       sass = require('gulp-sass'),
5       rename = require('gulp-rename'),
6       stripCssComments = require('gulp-strip-css-comments');
7
8 const src = {
9   sass : ['build/**/*.scss'],
10 };
11
12 // Tarefa que inicia e executa as demais tarefas
13 gulp.task('default',function(){
14   gulp.start(['build-compressed-css','build-css']);
15 });
16
17 // Tarefa que processa o Sass e remove os comentários do código,
18 // gerando o arquivo editorial-framework.css
19 gulp.task('build-css', function() {
20   gulp.src(src.sass)
21     .pipe(sass.sync().on('error', sass.logError))
22     .pipe(stripCssComments())
23     .pipe(gulp.dest('../css'));
24 });
25
26 // Tarefa que processa o Sass, remove os comentários do código
27 // e comprime o código, gerando o arquivo editorial-framework.min.css
28 gulp.task('build-compressed-css', function() {
29   gulp.src(src.sass)
30     .pipe(sass.sync().on('error', sass.logError))
31     .pipe(sass({outputStyle: 'compressed'}))
32     .pipe(rename(function (path) {
33       path.basename += ".min";
34     }))
35     .pipe(stripCssComments())
36     .pipe(gulp.dest('../css'));
37 });
38
```

Figura 2: Arquivo *gulpfile.js*

No arquivo **package.json** temos uma listagem dos pacotes do NPM necessário para a execução de cada tarefa. Também contém informações do projeto como nome, descrição, versão, autor e indicação do arquivo *gulpfile.scss*.

```
1
2 {
3   "name": "EditorialFramework",
4   "version": "1.0.0",
5   "description": "Front-end framework for development
6     fast and intuitive way of publishing sites.",
7   "main": "gulpfile.js",
8   "scripts": {
9     "gulp": "gulp"
10  },
11  "author": "Leandro Hora",
12  "license": "ISC",
13  "devDependencies": {
14    "gulp": "^3.9.1",
15    "gulp-rename": "^1.2.2",
16    "gulp-sass": "^3.1.0",
17    "gulp-strip-css-comments": "^1.2.0"
18  }
19 }
```

Figura 3: Arquivo package.json

Na pasta **scss** se encontram os arquivos referentes a cada tipo de elemento disponível pelo framework:

*graphic-elements.scss*: contém os estilos de css para os elementos gráficos como títulos, parágrafo, indentação de parágrafo, capitular, olho, alinhamento de texto, corpo e tabelas.

*grid.scss*: contém os estilos de configuração do sistema de grid como sua largura máxima, quantidade de colunas e espaço entre colunas.

*typography.scss*: contém os estilos que definem as configurações padrão da tipografia como corpo, peso, entrelinha e cor.

*ui-elements.scss*: contém os estilos referentes aos elementos de interface do usuário como botão, área de texto, seletor, rótulo, legenda, caixa de seleção, input de e-mail, número, pesquisa, texto, telefone, url, senha.

*variables.scss*: neste arquivo se concentram as variáveis com os valores padrões utilizados para cada tipo de elemento. Essas variáveis são utilizadas pelos arquivos *graphic-elements.scss*, *grid.scss*, *typography.scss*, *ui-elements.scss*. Desta forma a manutenção do código é viabilizada.

*reset.scss*: este arquivo contém os estilos que garantem a consistência e compatibilidade dos elementos do framework com todos os navegadores.

A pasta **build** contém um único arquivo, o *editorial-framework.scss*, sua função é reunir todos os arquivos contidos na pasta **scss** para a partir dele ocorrer a compilação final, resultando no arquivo *editorial-framework.css* e *editorial framework.min.css*.

## Componentes do Editorial Framework

O projeto aqui proposto disponibiliza um conjunto de componentes úteis para estruturação de layouts de sites editoriais, contando com um sistema de grid responsivo, tipografia, elementos gráficos e elementos de interface gráfica do usuário.

### Sistema de grid

O grid é formado por 12 colunas com largura máxima padrão de 960px que será reajustado de acordo com o tamanho do navegador em desktops ou dispositivos menores. A largura máxima pode ser reajustada com as necessidades de cada projeto. As colunas são combinadas para gerar blocos, possibilitando as mais variadas composições de blocos. A combinação de blocos devem conter sempre 12 colunas. Seus estilos de configuração como calha e margens, se encontram no arquivo *grid.scss*.

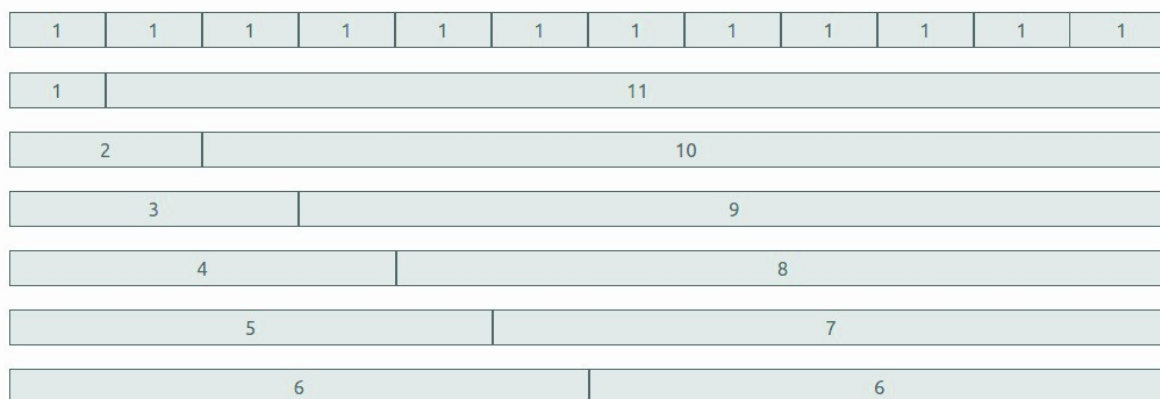


Figura 4: Grid com 12 colunas

O sistema é composto por três grids que reajustam a largura e a quantidade de blocos de acordo com medidas específicas definidas pelo Framework. O reajuste posiciona os blocos em sobreposição e define largura de 100% para todos os blocos independentes de sua quantidade de colunas. Os sistemas foram definidos como: *Large*, *Medium*, *Small*.

O reajuste do grid é feito apenas quando a largura máxima do navegador for menor que 768px. Caso contrário a forma do grid se mantém, independente do sistema de utilizado.

No sistema *Large* o reajuste do grid é feito quando a largura máxima do navegador for menor que 768px. No sistema *Medium* o reajuste do grid é feito quando a largura máxima do navegador for menor que 640px. No sistema *Small* o reajuste do grid é feito quando a largura máxima do navegador for menor que 480px.

A estrutura básica para a criação e combinação dos sistemas de grid pode ser obtida com as seguintes marcações de texto HTML.

```
<div class="row">
  <div class="col-1"> ... </div>
  <div class="col-11"> ... </div>
</div>
```

Figura 5: Marcação HTML para grid padrão

```
<div class="row-l">
  <div class="col-1"> ... </div>
  <div class="col-11"> ... </div>
</div>
```

Figura 6: Marcação HTML para grid large

```
<div class="row-s">
  <div class="col-1"> ... </div>
  <div class="col-11"> ... </div>
</div>
```

Figura 7: Marcação HTML para grid small

```
<div class="row-m">
  <div class="col-1"> ... </div>
  <div class="col-11"> ... </div>
</div>
```

Figura 8: Marcação HTML para grid medium

## Tipografia

Como padrão assumimos estilos com valores genéricos para a fonte que se aplicam em todos elementos de texto contidos na página, os estilos definem o corpo de texto, a entrelinha, o peso e a cor. Esses valores podem ser alterados no arquivo *typography.scss* de acordo com a necessidade da fonte escolhida para o projeto que utiliza a ferramenta aqui proposta.

## Elementos gráficos

Os elementos gráficos são componentes disponíveis para estruturar layout, organizar e estilizar conteúdos na página.

## Títulos

Os estilos de CSS para os títulos são aplicados nas *tags h1 a h6*, podendo ser aplicados também com as classes *.h1 a .h2* em qualquer *tag* que comporte conteúdo de texto. Os valores da estilização podem ser alterados em *graphic-elements.scss*. Abaixo seguem exemplos de sua aplicação em marcação *HTML*.

## Parágrafo

Os estilos de margem superior e inferior dos elementos de paragrafo foram redefinidos facilitar o espaçamento e construção de layout.

### Indentação de parágrafo

A indentação é aplicada nos elementos que comportam conteúdo de texto, utilizando a classe *.rec*, referente ao recuo do texto em relação à margem. Seus estilos se encontram no arquivo *graphic-elements.scss*.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Etiam pellentesque ex in augue egestas sagittis. Proin id imperdiet nunc.

```
<p class="rec">Lorem ipsum dolor  
sit amet,consectetur adipiscing elit.  
Etiam pellentesque ex in augue egestas  
sagittis...</p>
```

Figura 9: Marcação HTML para indentação de parágrafo

### Capitular

O capitular também é aplicado nos elementos que comportam conteúdo de texto, utilizando a classe *.cap*. Seus estilos se encontram no arquivo *graphic-elements.scss*. Abaixo, na figura 14, segue sua aplicação no elemento de parágrafo.

**L**orem ipsum dolor sit amet, consectetur adipiscing elit. Etiam pellentesque ex in augue egestas sagittis. Proin id imperdiet nunc.

```
<p class="cap">Lorem ipsum dolor  
sit amet,consectetur adipiscing elit.  
Etiam pellentesque ex in augue egestas  
sagittis...</p>
```

Figura 10: Marcação HTML para capitular



## Olho

O olho é aplicado especificamente no elemento de parágrafo, utilizando a classe `.eye`. Seus estilos se encontram no arquivo `graphic-elements.scss`. Abaixo, na figura 15, segue sua aplicação.

“ Lorem ipsum dolor sit amet,  
consectetur adipiscing elit.  
Nam eget velit cursus. ”

```
<p class="eye"> Lorem ipsum dolor sit  
amet, consectetur adipiscing elit.  
Nam eget velit cursus.</p>
```

Figura 11: Marcação HTML para olho

## Alinhamento de texto

O alinhamento de texto pode ser aplicado em qualquer elemento que comporta conteúdo de texto com a classe `.text-left` alinha o texto à esquerda; a classe `.text-right` alinha o texto à direita; a classe `.text-center` centraliza o texto e a classe `.text-justify` mantém o texto justificado. Na figura que segue temos exemplos de sua aplicação.

```
<p class="text-left"> ... </p>  
<p class="text-right"> ... </p>  
<p class="text-center"> ... </p>  
<p class="text-justify"> ... </p>
```

Figura 12: Marcação HTML para alinhamento de texto

## Peso

O peso é utilizado em qualquer elemento que comporta conteúdo de texto com a classe `.bolder`, aplicando o peso *bold*; a classe `.normal`, aplicando o peso padrão e a classe `.lighter`, aplicando o peso light. Abaixo, na figura 17, segue sua aplicação.

```
<span class="bolder"> bolder </span>  
<span class="normal"> normal </span>  
<span class="lighter"> lighter </span>
```

Figura 13: Marcação HTML para peso de texto



## Tabelas

A tabela, além do estilo padrão, apresenta quatro variações de estilo que podem ser aplicadas de acordo com as necessidades de cada projeto. As variações são: tabela listrada, tabela com *hover*, tabela com borda e tabela condensada. O estilo padrão é aplicado no elemento de tabela com a classe `.table`. Abaixo segue a figura da tabela padrão.

Nome	Idade	Email
Lucas	23	lucas@email.com
Marcos	18	marcos@email.com
Julia	27	julia@email.com

Figura 14: Tabela padrão

Os estilos das variações são aplicadas adicionando suas classes. As tabelas podem conter mais de um estilo de variação. Podemos visualizar as variações a seguir.

Nome	Idade	Email
Lucas	23	lucas@email.com
Marcos	18	marcos@email.com
Julia	27	julia@email.com

Figura 15: Tabela listrada

Nome	Idade	Email
Lucas	23	lucas@email.com
Marcos	18	marcos@email.com
Julia	27	julia@email.com

Figura 16: Tabela com hover

Nome	Idade	Email
Lucas	23	lucas@email.com
Marcos	18	marcos@email.com
Julia	27	julia@email.com

Figura 17: Tabela com borda

Nome	Idade	Email
Lucas	23	lucas@email.com
Marcos	18	marcos@email.com
Julia	27	julia@email.com

Figura 18: Tabala condensada

Abaixo temos as marcações *HTML* com os elementos de tabela e a aplicação das classes de variação.

```
<table class="table">
  <thead>
    <tr>
      <th> Nome </th>
      <th> Idade </th>
      <th> Email </th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td> Lucas </td>
      <td> 23 </td>
      <td> lucas@email.com </td>
    </tr>
    [...]
  </tbody>
</table>
```

Figura 20: Marcação HTML para tabela padrão

```
<table class="table table-striped">
  <thead>
    <tr>
      <th> Nome </th>
      <th> Idade </th>
      <th> Email </th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td> Lucas </td>
      <td> 23 </td>
      <td> lucas@email.com </td>
    </tr>
    [...]
  </tbody>
</table>
```

Figura 19: Marcação HTML para tabela listrada

```
<table class="table table-hover">
  <thead>
    <tr>
      <th> Nome </th>
      <th> Idade </th>
      <th> Email </th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td> Lucas </td>
      <td> 23 </td>
      <td> lucas@email.com </td>
    </tr>
    [...]
  </tbody>
</table>
```

Figura 22: Marcação HTML para tabela com hover

```
<table class="table table-border">
  <thead>
    <tr>
      <th> Nome </th>
      <th> Idade </th>
      <th> Email </th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td> Lucas </td>
      <td> 23 </td>
      <td> lucas@email.com </td>
    </tr>
    [...]
  </tbody>
</table>
```

Figura 21: Marcação HTML para tabela com borda

```

<table class="table table-condensed">
  <thead>
    <tr>
      <th> Nome </th>
      <th> Idade </th>
      <th> Email </th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td> Lucas </td>
      <td> 23 </td>
      <td> lucas@email.com </td>
    </tr>
    [...]
  </tbody>
</table>

```

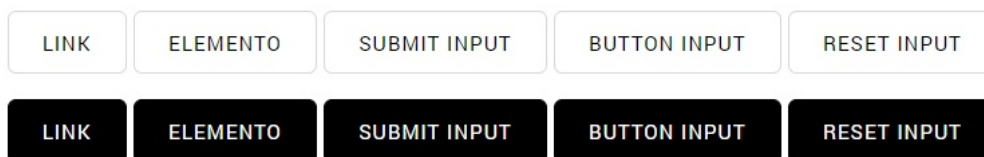
Figura 23: Marcação HTML para tabela condensada

## Elementos de interface gráfica do usuário

O projeto aqui apresentado disponibiliza como elementos da interface do usuário botões e campos de formulário. Os estilos dos elementos se encontram no arquivo *ui-elements.scss*.

### Botões

Os botões apresentam um estilo padrão e uma variação. O estilo padrão é aplicado pela classe *.button* e também diretamente nos elementos de *input* e *button*. A variação é aplicada com a classe *.button-primary*.



```

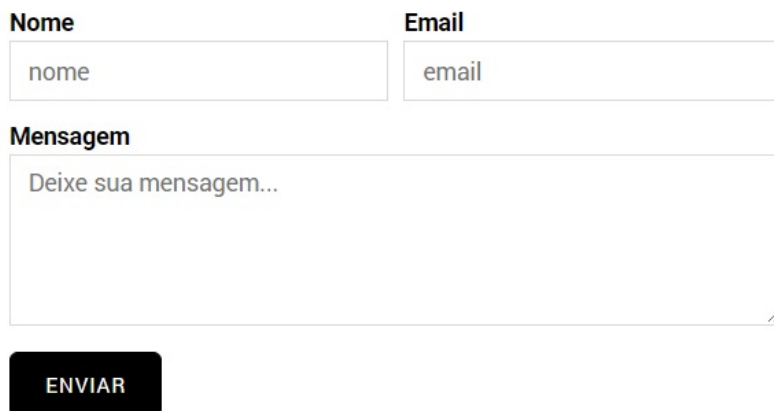
<a class="button" href="#"> Link </a>
<button> Elemento </button>
<input type="submit" value="submit input">
<input type="button" value="button input">
<input type="reset" value="reset input">

```

Figura 24: Marcação HTML para tabela botões

## Formulários

Os estilos para formulário são aplicados nos elementos de *input* (elementos de entrada de dados) como email, número, pesquisa, texto, telefone, link e senha. Os estilos dos elementos se encontram no arquivo *ui-elements.scss*.



The visual representation of the form shows two input fields side-by-side. The first is labeled 'Nome' and contains the placeholder text 'nome'. The second is labeled 'Email' and contains the placeholder text 'email'. Below these is a larger text area labeled 'Mensagem' with the placeholder text 'Deixe sua mensagem...'. At the bottom is a black button with the white text 'ENVIAR'.

```
<form>
  <div class="row-s">
    <div class="col-6">
      <label> Nome </label>
      <input type="text" name="nome" placeholder="nome" />
    </div>
    <div class="col-6">
      <label> Email </label>
      <input type="email" name="email" placeholder="email" />
    </div>
  </div>
  <div class="row-s">
    <div class="col-12">
      <label> Mensagem </label>
      <textarea> </textarea>
      <input class="button-primary" type="submit" value="Enviar"/>
    </div>
  </div>
</form>
```

Figura 25: Marcação HTML para formulário