Documento de Arquitetura

Eleições 2018Template de sistemas

AlmgANTT

[Antes de utilizar este documento **LEIA AS INSTRUÇÕES** a seguir:

Para garantir a padronização dos documentos utilizados, são utilizados dois recursos especiais do Microsoft Word: campos automáticos (com fundo cinza) e estilos de fontes pré-definidos.

Para atualizar o valor dos campos automáticos, realize os seguintes passos:

Selecione todo o documento (Pressione Ctrl + T)

Pressione F9 e informe os valores conforme solicitado.

Para os cabeçalhos e rodapé é necessário repetir o mesmo processo, porém é necessário estar no modo de edição de cabeçalho e rodapé.

Os estilos de fontes pré-definidos são os seguintes:

Dica: Usado para criar dicas de preenchimento.

Normal: Usado para textos descritivos nas seções do documento

Título 1: Seções de nível 1

Título 2: Subseções de nível 2

Título 3: Subseções de nível 3

TituloDocumento: Nome do artefato na capa do documento

NomeCliente: Nome do cliente na capa do documento

NomeProjeto: Nome do projeto na capa do documento

Cabeçalho: Fonte utilizada nos cabeçalhos do documento

Importante: Após o preenchimento do documento e a leitura de todas as dicas, é necessário ocultar todos os textos de dica (que possuem o mesmo formato que você está lendo). Na guia Início do Menu, identifique o grupo Estilo e selecione o botão "Mais" (). A janela Aplicar Estilos será exibida. Escreva “Dica” no campo “Nome do Estilo” e selecione o botão “Modificar...”. A janela Modificar Estilos aparecerá. No canto inferior esquerdo, selecione o botão “Formatar” e em seguida selecione “Fonte...”. Marque o checkbox “Oculto” e selecione “OK”. Selecione “OK” novamente e o texto de dica será ocultado.

Para exibir a dica novamente, clique com o botão direito nesta página -> Estilos -> Atualizar Dica para Corresponder a Seleção.

]

**Revisões do Documento**

|  |  |  |  |
| --- | --- | --- | --- |
| **Revisão** | **Data** | **Lista de Autores** | **Observações** |
| 1.0 | 18/10/2018 | Luís Gustavo | Versão inicial |
|  |  |  |  |

**Sumário**

[1. Objetivo 4](#_Toc530470276)

[2. Insumos básicos 4](#_Toc530470277)

[3. Compatibilidade 4](#_Toc530470278)

[4. Protótipo HTML 5](#_Toc530470279)

[5. Master 5](#_Toc530470280)

[6. Calendário de tarefas 5](#_Toc530470281)

[6.1. Lista de tarefas 5](#_Toc530470282)

[6.2. Lista de tarefas internas 6](#_Toc530470283)

[7. Componentes 6](#_Toc530470284)

[7.1. Alternar Classes 6](#_Toc530470285)

[7.2. Modal 7](#_Toc530470286)

[7.2.1. Modal de confirmação 7](#_Toc530470287)

[7.3. Calendário 7](#_Toc530470288)

[7.4. Alternar visibilidade 7](#_Toc530470289)

[7.5. Select 8](#_Toc530470290)

[7.6. Gráfico 8](#_Toc530470291)

[7.7. Paginação 8](#_Toc530470292)

[7.8. Mensagens de aviso 8](#_Toc530470293)

[7.9. Validação de formulários 9](#_Toc530470294)

[7.10. Máscara 9](#_Toc530470295)

[7.11. Duplicar conteúdo 9](#_Toc530470296)

[7.12. Adicionar conteúdo 9](#_Toc530470297)

[7.13. Abas 10](#_Toc530470298)

[7.13.1. Botões Anterior / Próximo 10](#_Toc530470299)

[8. Funcionalidades HTML 10](#_Toc530470300)

[9. Versão utilizável 11](#_Toc530470301)

[10. Acessibilidade 11](#_Toc530470302)

# Objetivo

O objetivo deste documento é descrever como foi feita a implementação dos requisitos no protótipo HTML do template de sistemas da ANTT. O protótipo foi desenvolvido de forma desacoplada de qualquer linguagem back-end para que se possa utilizá-la como base de forma independente de qualquer linguagem de desenvolvimento.

# Insumos básicos

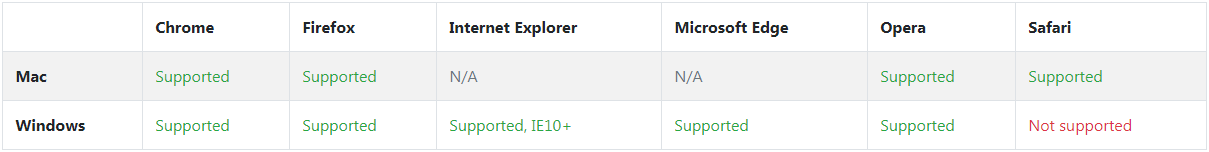
Para construção do protótipo foi utilizado HTML, CSS e JS. Para a renderização do CSS foi utilizado o pré-processador Sass e para HTML e JS utilizados dos seguintes insumos:

|  |  |  |
| --- | --- | --- |
| **NOME** | **TIPO** | **DESCRIÇÃO** |
| Bootstrap v4.1.3 | Framework front-end | Utilizado como base estrutural para o projeto. Disponível em: https://getbootstrap.com/ |
| jQuery v3.3.1 | Biblioteca Javascript | Utilizada como auxiliar para ações em JS |
| Popper | Auxiliar de plug-ins | Utilizado como auxiliar para plug-ins do framework Bootstrap |

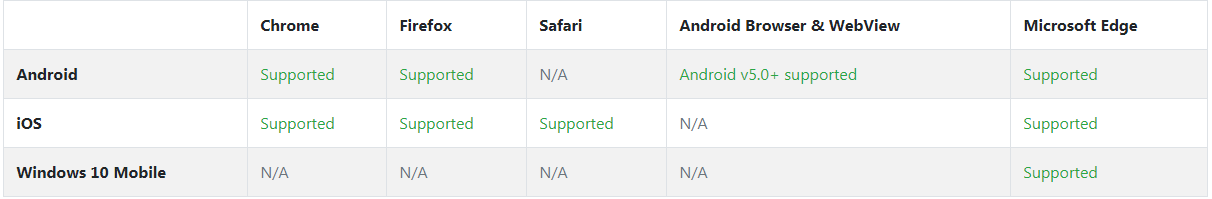
# Compatibilidade

A compatibilidade do protótipo desenvolvido é herdada do framework e demostrada nas figuras abaixo:

* Suporte para navegadores desktop:



* Suporte para navegadores mobile:



# Protótipo HTML

Visto que o protótipo pode servir como base para várias equipes de desenvolvimento, foram criadas algumas interações em JavaScript para que ele fique o mais funcional possível, porém, essas interações podem (e devem) ser alteradas de acordo com cada implementação.

# Master

A fim de reaproveitar blocos de códigos comuns a todas as páginas do sistema, foi criada um HTML chamado ‘**master.html**’ que contém esses blocos de código. Como estratégia para incluir esses blocos nas demais páginas foi utilizado a função ‘load’ do Jquery no arquivo ‘**global.js**’, encontrada entre os blocos de comentário ‘[START INCLUDE CODE]’ e ‘[END INCLUDE CODE]’. Essa estratégia deve ser alterada de acordo com o paradigma mais adequado de cada linguagem de desenvolvimento.

Alguns outros trechos de código também foram replicados e adicionados na pasta ‘**include**’ do protótipo.

# Calendário de tarefas

Alguns sistemas terão informações laterais sobre as tarefas relacionadas. Para importar o Calendário de Tarefas no sistema é necessário, além da marcação do HTML, importar o arquivo ‘**aside.js**’. Ao utilizar o Calendário de tarefas, também é necessário se atentar a implementação do plug-in de calendário (ver item **7.3**).

Ao selecionar a data, alguns componentes da tela devem se atualizar, para isso foi adicionado um log no console, presente no arquivo ‘**aside.js’** indicando o momento de se realizar a ação.

## Lista de tarefas

A cada comando de manter a lista de tarefas deve-se realizar uma requisição AJAX para atualizar o estado do componente na base de dados usada.

As informações não visuais das tarefas estão em uma div com a classe ‘**item-info**’ presente no HTML em cada linha de dados. O script utiliza das classes presentes em cada input para popular a modal de edição:

* **ref-data\_inicial** : Informação da data Inicial da tarefa
* **ref-data\_final**: Informação da data final da tarefa
* **ref-desc** : Informação da descrição da tarefa
* **ref-id** : Informação do ID da tarefa (Se a tarefa está sendo criada o ID da tarefa será vazio)
* **ref-title** : Informação do título da tarefa

## Lista de tarefas internas

Para importar a lista de tarefas nas páginas internas do sistema é necessário implementar o código do arquivo ‘**task\_fixed**’ na pasta ‘**include**’.

# Componentes

Visto que o protótipo pode servir como base para várias equipes de desenvolvimento, foram criadas algumas interações em JavaScript para que ele fique o mais funcional possível, porém, essas interações podem (e devem) ser alteradas de acordo com cada implementação.

Alguns componentes utilizam plug-ins externos que precisam ser importados, segue a lista dos plug-ins e sua função no sistema:

**O**

|  |  |  |
| --- | --- | --- |
| **NOME** | **FUNÇÃO** | **DOCUMENTAÇÃO** |
| Tempusdominus v5.1.2 | Calendário | <https://tempusdominus.github.io/bootstrap-4/> |
| Moment v2.22.2 | Manipulador de datas | <http://momentjs.com/docs/> |
| Chart js v2.7.3 | Gráfico | <http://www.chartjs.org/docs/latest/> |
| Select2 4.0.6 | Selectbox | <https://select2.org/getting-started/basic-usage> |
| Validation v1.17.0 | Validação de formulário | <https://jqueryvalidation.org/documentation/> |
| Toastr v2.1.4 | Mensagens de aviso | <https://codeseven.github.io/toastr/demo.html> |
| Mask v1.14.15 | Máscara | <https://github.com/igorescobar/jQuery-Mask-Plugin> |

## Alternar Classes

Script criado para trocar a classe ‘active’ de um determinado alvo. Na marcação do

elemento trigger deve ter os seguintes atributos:

* **data- toggle="class"** : atributo informando que o elemento é um trigger do script de classe
* **data-target** : atributo com informação sobre o elemento alvo. Se for um ID deve vir acompanhado de ‘#’ antes do valor.

Para utilizar essa funcionalidade é necessário importar o arquivo ‘**apply\_toggle.js**’

## Modal

A modal do sistema apresenta marcação semelhante ao descrito pela documentação do framework (<https://getbootstrap.com/docs/4.1/components/modal/>), porém em alguns casos, além da marcação padrão, existem alguns atributos opcionais que foram adicionados no botão de trigger, descritos a seguir:

* **data-title** : atributo responsável por informar qual é o título da modal.
* **data-desc** : atributo responsável por informar qual é o conteúdo do corpo da modal.

Para utilizar essa funcionalidade é necessário importar o arquivo ‘**apply\_modal.js’**

### Modal de confirmação

Foi utilizada apenas uma estrutura de modal para as confirmações do sistema. Para diferenciar um contexto do outro é necessário passar no elemento trigger dois atributos:

* **data-action** : ação que será executada caso haja confirmação da modal
* **data-ref** : referência do elemento em que a ação irá ser executada. Caso não seja declarada, o valor padrão é **‘sistema’.**

## Calendário

Para incluir o calendário é necessário importar os arquivos ‘**tempusdominus-bootstrap-4.min.js’** e ‘**apply\_date.js**’, além da folha de estilo para o componente: ‘**tempusdominus-bootstrap-4.min.css’**. Esse plug-in de calendário trabalha em conjunto com um script auxiliar de manipulação de datas, portanto também é necessário importar o arquivo ‘**moment.min.js**’ e sua tradução para português ‘**moment\_pt-br.js’**

Além da implementação padrão declarada pela documentação, foi implementado um script para criação de range entre duas datas (data inicial até data final). Para implementar essa funcionalidade é necessário criar uma estrutura HMTL conforme mostra a página ‘**master.html**’ na sessão da modal de tarefas.

## Alternar visibilidade

Além do comportamento padrão disponibilizado pelo framework (<https://getbootstrap.com/docs/4.1/components/collapse/>), foi criado um script que alterna a visibilidade do elemento imediatamente posterior ao trigger. A marcação no trigger deve conter o atributo ‘ **data-toggle="collapse-next"** ’

Também é possível alterar a visibilidade de elementos, controlados por um selectbox. É necessário colocar no selectbox os seguintes atributos:

* Classe ‘**changeContent’**
* **data-container**: referência para o ID do container onde estarão os conteúdos
* **data-content (na tag option)**: referencia para o ID do conteúdo

Para utilizar essa funcionalidade é necessário importar o arquivo ‘**apply\_toggle.js’**

## Select

Para estilização e controle do elemento ‘select’ foi utilizado o plugin ‘select2’. É necessário adicionar a classe ‘**select2**’ no elemento.

Para utilizar essa funcionalidade é necessário importar os arquivos ‘**select2.min.js**’, ‘**apply\_select.js**’ e ‘**select2.min.css**’

## Gráfico

Os dados do gráfico estão sendo carregados via JSON no formato exigido pela documentação do plugin (<http://www.chartjs.org/docs/latest/getting-started/usage.html>). Na marcação do elemento trigger deve ser ter alguns atributos necessários pra montar o gráfico:

* **data-type** : tipo de gráfico que deve ser montado. Os valores possíveis são disponibilizados pelo plugin.
* **data-path** : url para o arquivo com as informações do gráfico

É importante ressaltar que alguns gráficos apresentam uma relação de equivalência entre os valores. Como por exemplo, o valor mostrado equivale a 30% do total. Nesses casos é necessário adicionar na marcação do elemento uma classe ‘**relative**’ que indicará ao JS que o primeiro valor do JSON é relativo ao segundo valor do JSON. Foi adicionado os arquivos ‘**demand.json**’ e ‘**demand-final.json**’ na pasta ‘json’ do protótipo, para exemplificar a estrutura requerida.

Para o tipo ‘doughnut’ é possível gerar um gráfico mais fino, para isso é necessário adicionar a classe ‘**small-chart**’ no elemento do gráfico e também um gráfico intermediário, com a classe ‘**medium-chart**’.

Para utilizar essa funcionalidade é necessário importar os arquivos ‘**Chart.min.js**’ e ‘**apply\_chart.js**’.

## Paginação

A marcação da paginação é dada por uma classe ‘**pagination-content**’ que através da importação do arquivo ‘**apply\_pagination.js**’ importa a estrutura ‘**pagination.html**’ presente na pasta include do protótipo. A cada click da paginação é gerado um log no console informando ao desenvolvedor a necessidade de implementação do código AJAX para atualizar a lista. Também é atualizado o input hidden ‘**searchPage**’ presente no formulário contendo a informação de qual página foi solicitada

## Mensagens de aviso

As mensagens de aviso são disparadas via javascript seguindo o indicado pela documentação do plugin, não tendo uma marcação HTML para tal.

Para utilizar essa funcionalidade é necessário importar os arquivos ‘**toastr.min.js’** e ‘**apply\_toastr.js’**

## Validação de formulários

Para validar formulários é necessário adicionar a classe ‘**validateForm**’no elemento form. Foi adicionado logs no console nos pontos onde deve-se ter uma intervenção do desenvolvedor no arquivo ‘**apply\_pagination.js**’.

Prevendo o possível uso do plugin para formulários que necessitam de um fluxo assíncrono, o script faz o uso da classe ‘**submit\_form**’ no elemento do formulário para o caso de seguir o fluxo padrão do submit. Caso a classe for omitida, o script segue o fluxo para a implementação assíncrona (logs no console foram criados para mostrar para o desenvolvedor onde implementar o código).

Para utilizar essa funcionalidade é necessário importar os arquivos ‘**jquery.validate.min.js**’ e ‘**apply\_pagination.js**’. Essa funcionalidade tem uma dependência com os arquivos explicitados no item **7.8** para que (se for o caso) possa mostrar as mensagens de feedback.

## Máscara

No protótipo desenvolvido foi implementado as seguintes máscaras, com suas respectivas classes de controle:

* **mask-tel** : máscara de telefone (9º dígito opcional).
* **mask-number**: apenas números válidos.
* **mask-date**: máscara de data.
* **mask-dateTime**: máscara de data e hora.

Para utilizar essa funcionalidade é necessário importar os arquivos ‘**jquery.mask.min.js’** e ‘**apply\_mask.js**’.

## Duplicar conteúdo

Para duplicar o conteúdo é necessário que a estrutura ‘pai’ tenha a classe ‘duplicate-content’ e a existência dos ícones de controle para adicionar ou remover o conteúdo, assim como demonstrado na página ‘parlamentar\_cadastrar.html’.

Os ‘ids’ e ‘names’ dos inputs serão gerados dinamicamente caso eles tenham a atributo ‘**data-id**’ neles. Caso o ‘name’ do input não deva ser alterado, é necessário adicionar o atributo ‘**data-ignoreName**’ no elemento.

Para utilizar essa funcionalidade é necessário importar o arquivo ‘**duplicate\_content.js**’.

## Adicionar conteúdo

Para adicionar conteúdos dinamicamente de acordo com o preenchimento de alguns campos do formulário é necessário a adição da classe ‘add\_content’ no elemento trigger e os seguintes atributos

* **data-form** : Marcação do ID da sessão do formulário que contém os campos relevantes para gerar os blocos visuais.
* **data-target** : Marcação do container que irá armazenar os blocos gerados.
* **data-content:** Marcação dos ID’s (separados por vírgula) dos inputs que irão formar o bloco.

No container que irá armazenar os blocos gerados, é necessário adicionar também, a classe ‘**add\_content-container**’. É possível observar que além da parte visual dos blocos gerados, existem também alguns inputs hidden que armazenam os valores dos campos da sessão do formulário. Eles mantêm o valor do input no momento de inclusão do bloco e seu name é formado pelo name original do input procedido do texto ‘\_saved’.

Para utilizar essa funcionalidade, é necessário importar os arquivos ‘**add\_content.js**’. Essa funcionalidade tem uma dependência com os arquivos explicitados no item **7.8** para que o sistema possa mostrar as mensagens de feedback.

## Abas

Além da marcação padrão provida pelo framework (<https://getbootstrap.com/docs/4.1/components/navs/#javascript-behavior>), o protótipo desenvolvido disponibiliza uma visualização de abas compatível com dispositivos móveis. Para utilizar essa funcionalidade é necessário um container com uma classe ‘**responsive-tab’**.

Para utilizar essa funcionalidade é necessário importar o arquivo ‘**apply\_tab.js**’.

### Botões Anterior / Próximo

Para adicionar a funcionalidade de avançar e voltar abas através de botões de controle, é necessário adicionar um container com a classe ‘**btn-tabContent**’, e dois elementos descendentes com a classe ‘**next-tab**’ para avançar e a classe ‘**prev-tab**’ para voltar.

Considerando que para avançar, exista uma interação com o servidor, existe um log no console indicando para o desenvolvedor o momento de realizar a requisição AJAX.

# Funcionalidades HTML

No protótipo, existem algumas marcações de elementos que devem ser tratados programaticamente.

Na **lista de resultados**, funcionalidade como ordenação de resultados, número de resultados e resultados não encontrados, devem ser tratados conforme os requisitos de cada sistema.

# Versão utilizável

A versão utilizável do protótipo desenvolvido está disponibilizada no GitLab, na URL (<http://gitlab.squadra.com.br/luis.batista/template_sistemas_antt-2.0.git>). Os arquivos Sass utilizados podem ser importados para versão final do sistema como forma de controle e visualização via console de regras CSS, mas sua importação é opcional, visto que os arquivos já foram compilados em CSS antes da entrega.

# Acessibilidade

O código gerado foi validado pelas regras do w3c pelo site <https://validator.w3.org/> e pelas regras WCAG 2.0 validados pelo software Total Validator.

Os relatórios de validação estão presentes em docs/wcag e docs/w3c.

Nos relatórios de validação WCAG, o erro “*Nest headings properly (H1 > H2 > H3)*” acontece por hierarquia de escrita dos títulos. O ‘h1’ sempre se refere ao título principal do portal que é inserida posteriormente pela função ‘load’ do jquery (conforme dito no item **5. Master**), não sendo visível pelo validador, que considera apenas o HTML escrito inicialmente e desconsidera o HTML gerado depois da página carregada. Em ambiente de desenvolvimento, este erro não deverá ocorrer, pois a estratégia de ‘include’ é outra.