

O que aprenderemos?

- Criar tabela no banco de dados para salvar os dados do calendário;
- Criar factory para gerar e armazenar dados falsos para popular o banco de dados;
- Adicionar, editar, deletar e atualizar eventos no calendário com ajax através de uma janela *modal*, sem precisar recarregar a página;
- Utilizar elementos estilizados para datas, horários e cores;

Sumário

1.Criar novo projeto.....	2
2.Adicionar providers e aliases.....	2
3.Adicionar os arquivos do Fullcalendar ao projeto.....	3
4.Criar banco de dados.....	3
5.Criar <i>migration</i> e <i>model</i>	4
6.Criar uma <i>factory</i> para criar dados falsos para popular a tabela automaticamente.....	5
7.Criar <i>controller</i> e rota.....	6
8.Adicionar os códigos do calendário, incluir javascripts e stylesheets.....	6
9.Criar formulário em uma janela <i>modal</i> para adicionar evento via Ajax.....	8

1. Criar novo projeto

Rodar o comando:

```
$ composer create-project --prefer-dist laravel/laravel calendar_laravel
```

2. Adicionar providers e aliases

Esses são os pacotes que permitem o manejo de formulários e assets.

Disponível em: <https://laravelcollective.com/docs/master/html#installation>

Rodar o comando:

```
$ composer require laravelcollective/html
```

Acessar o arquivo `config/app.php` e adicionar as seguintes linhas dentro dos pacotes indicados.

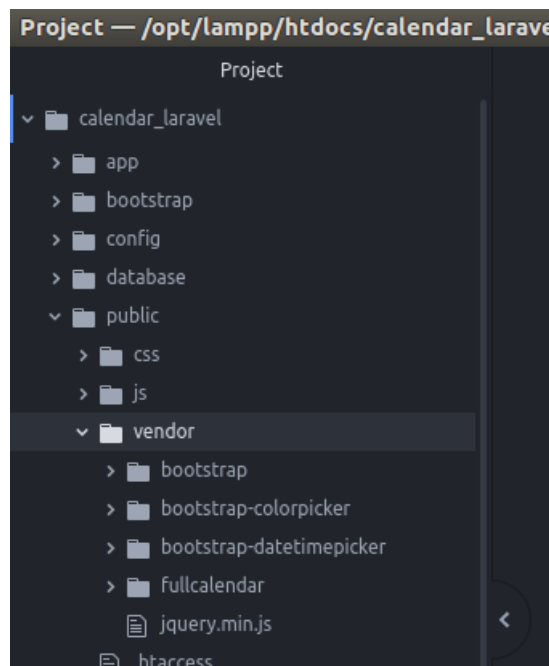
```
'providers' => [  
    // ...  
    Collective\Html\HtmlServiceProvider::class,  
    // ...  
],  
'aliases' => [  
    // ...  
    'Form' => Collective\Html\FormFacade::class,  
    'Html' => Collective\Html\HtmlFacade::class,  
    // ...  
],
```

3. Adicionar os arquivos do Fullcalendar ao projeto

Disponíveis para download em:

https://drive.google.com/file/d/1Z8y241p46jgzd4rf8WSSHW1M_7vyYUlb/view?usp=sharing

No projeto ir na pasta public, criar pasta com o nome “vendor” e colar os arquivos dentro da pasta “vendor”



4. Criar banco de dados

Criar banco de dados com o nome “calendario”

Exemplo de criação no banco de dados mysql por linha de comando

Passo 1: Iniciar servidor artisan do laravel. Digitar o comando dentro da pasta do projeto:

```
$ php artisan serve
```

Abrir outro terminal dentro da pasta do projeto e digitar o comando:

```
$ mysql -u root -p
```

Entrar com a senha definida para o seu banco de dados.

Comando para criar o banco de dados:

```
mysql> create database bd_calendario;
```

Retorno esperado = Query OK, 1 row affected (0,00 sec)

Para sair da linha de comando do mysql, digite:
mysql> exit

Configurar os dados do banco de dados no arquivo .env, localizado na raiz do projeto.

Preencher os campos com os dados do seu banco de dados:

```
...
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=bd_calendario
DB_USERNAME=root
DB_PASSWORD=masterkey
...
```

5. Criar *migration* e *model*

Executar o seguinte comando para criarmos o *model* Event e a migration para a criação da tabela no banco de dados:

```
$ php artisan make:model Event -m
```

Resultado esperado =
Model created successfully.
Created Migration: ...(data e horario)_create_events_table

Na migração criada , colocar os comandos para criar as colunas na tabela.

Ir em database/migrations/...nome_da_migracao_criada

Dentro na função *up* e dentro de Schema::create('events', function (Blueprint \$table)
{... adicionar as seguintes linhas para a criação das colunas:

```
$table->string('title');
$table->datetime('start');
$table->datetime('end');
$table->string('color', 7);
```

Adicionar no model -> Event model as características da tabela events

Em app/Event.php adicionar as seguintes linhas:

```
protected $table = 'events';
```

```
protected $fillable = [  
    'title','start','end','color'  
];
```

6. Criar uma *factory* para criar dados falsos para popular a tabela automaticamente

Acessar o arquivo em `database/factories/UserFactory.php` e definir nova *factory*.

A *factory* ficará assim:

```
use Carbon\Carbon;  
$factory->define(App\Event::class, function (Faker $faker) {  
    $date_start = $faker->dateTimeThisYear();  
    $date_end = new Carbon($date_start->format('r'));  
    return [  
        'title' => $faker->sentence(4),  
        'start' => $date_start,  
        'end' => $date_end->addHours($faker->numberBetween(1,35)),  
        'color' => $faker->hexColor  
    ];  
});
```

O *carbon* é uma *API PHP* para dados *DateTime*, ele possui todas as funções herdadas da classe base *DateTime* e isso permite que você acesse a funcionalidade básica em *DateTime*. Documentação disponível em: <http://carbon.nesbot.com/docs/>

Criar eventos falsos

Digite o comando:

```
$ php artisan make:seeder EventsTableSeeder
```

Acrir aquivo em `database/seeds/EventsTableSeeder.php`

Adicionar dentro da função *run* a seguinte linha para a criação da *factory*:

```
factory(App\Event::class)->times(100)->create();
```

Ativar para realizar a migração dos dados falsos

No arquivo `database/seeds/DatabaseSeeder.php`, substituir a linha comentado por:

```
$this->call(EventsTableSeeder::class);
```

Executar o comando para criar a tabela e inserir os dados falsos:

```
$ php artisan migrate --seed
```

Para listar os dados cadastrados no mysql console:

```
$ mysql -u root -p
mysql> use db_calendario;
mysql> select * from events;
```

7. Criar *controller* e rota

Gerar o *controller* que irá agrupar a lógica das rotas relacionadas a classe.

Execute o comando:

```
$ php artisan make:controller EventsController --resource
```

Acessar o arquivo criado em app/Http/Controllers/EventsController.php

Adicionar referência no topo do arquivo

```
...
use App\Event;
```

Adicionar as seguintes linhas na função index(){...

```
$data = Event::get(['id','title','start','end','color']);
return Response()->json($data, 200, array(), JSON_PRETTY_PRINT);
```

Adicionar rota em routes/web.php

```
Route::resource('events','EventsController');
```

Reiniciar o server , Ctrl+C no terminal onde o server foi iniciado e execute novamente o comando (caso o server tenha sido iniciado antes de ter feito a alterações dos dados no arquivo de configuração do banco de dados .env)

```
$ php artisan serve
```

Navegar para a url /events e deverá ver a lista dos dados cadastrados em eventos no formato *json*

8. Adicionar os códigos do calendário, incluir javascripts e stylesheets

Adicionar linhas para inclusão dos javascript, com os caminhos para os arquivos do bootstrap e fullcalendar adicionados anteriormente, referentes as funções necessárias para as ações do calendário.

Links úteis:

```
# modal
http://getbootstrap.com/docs/4.0/getting-started/download/
# colorpicker
https://farbelous.github.io/bootstrap-colorpicker/
# datepicker
https://github.com/T00rk/bootstrap-material-datetimepicker
```

Sugestão: inserir os javascripts entre o final da tag `</body>` e o final da tag `</html>`

```
{!! Html::script('vendor/jquery.min.js') !!}
{!! Html::script('vendor/fullcalendar/lib/moment.min.js') !!}
{!! Html::script('vendor/fullcalendar/fullcalendar.min.js') !!}
{!! Html::script('vendor/fullcalendar/locale/pt-br.js') !!}
{!! Html::script('vendor/bootstrap/assets/js/vendor/popper.min.js') !!}
{!! Html::script('vendor/bootstrap/dist/js/bootstrap.min.js') !!}
{!! Html::script('vendor/bootstrap-datetimepicker/js/bootstrap-material-datetimepicker.js') !!}
{!! Html::script('vendor/bootstrap-colorpicker/dist/js/bootstrap-colorpicker.min.js') !!}
```

Logo abaixo, inserir esses códigos para a inicialização e algumas configurações do calendário.

```
<script type="text/javascript">
var BASEURL = "{{ url('/') }}";
$(document).ready(function() {
    $('#calendar').fullCalendar({
        header: {
            left: 'prev,next today',
            center: 'title',
            right: 'month,basicWeek,basicDay'
        },
        navLinks: true, //para ativar a navegacao nos links para listar dias/semanas
        editable: true,
        selectable: true, // para permitir selecionar cada evento
        selectHelper: true, // permite adicionar novo evento no calendario
        select: function(start){
            start = moment(start.format());
            $('#date_start').val(start.format('DD-MM-YYYY'));
            $('#responsive-model').modal('show');
        },
        events: BASEURL + '/events'
    });
});
</script>
```

Incluir stylesheets para a definição dos estilos da página e do calendário.

Adicionar dentro da tag <header>

```
{!! Html::style('css/app.css') !!}  
{!! Html::style('vendor/fullcalendar/fullcalendar.min.css') !!}  
{!! Html::style('vendor/bootstrap/dist/css/bootstrap.min.css') !!}  
{!! Html::style('vendor/bootstrap-datetimepicker/css/bootstrap-material-datetimepicker.css') !!}  
{!! Html::style('vendor/bootstrap-colorpicker/dist/css/bootstrap-colorpicker.min.css') !!}
```

Dentro da tag <body> adicionar o as tags <div> e o id="calendar" responsável por receber os estilos e configurações do calendário que serão mostrados na página.

```
<div class="container">  
  <div id="calendar"></div>  
</div>
```

Navegar para a url <http://localhost:8000/> e veja o calendário que será mostrado.

9. Criar formulário em uma janela *modal* para adicionar evento via Ajax

Informação/curiosidade:

Ajax (JavaScript assíncrono e XML) é um método de criação de aplicativos interativos para a Web que processam solicitações de usuários imediatamente. O Ajax combina várias ferramentas de programação, incluindo JavaScript, HTML dinâmico, Extensible Markup Language (XML), folhas de estilo em cascata (CSS), o Document Object Model (DOM) e o objeto Microsoft XMLHttpRequest.

Para criar o formulário *modal* adicione esses códigos acima da tag <div id="calendar"></div>

Exemplo:

[código aqui]

```
<div id="calendar"></div>
```

```
<div id="responsive-model" class="modal fade" tabindex="-1" data-backdrop="static">  
  <div class="modal-dialog">  
    <div class="modal-content">  
      <div class="modal-header">  
        <h4>REGISTRAR NOVO EVENTO</h4>
```



```

</div>
<div class="modal-body">
  <div class="form-group">
    {{ Form::label('title', 'TÍTULO') }}
    {{ Form::text('title', old('title'), ['class' => 'form-control']) }}
  </div>
  <div class="form-group">
    {{ Form::label('date_start', 'INÍCIO EVENTO') }}
    {{ Form::text('date_start', old('date_start'), ['class' => 'form-control', 'readonly' => 'true']) }}
  </div>
  <div class="form-group">
    {{ Form::label('time_start', 'HORA INÍCIO') }}
    {{ Form::text('time_start', old('time_start'), ['class' => 'form-control']) }}
  </div>
  <div class="form-group">
    {{ Form::label('date_end', 'FIM DO EVENTO') }}
    {{ Form::text('date_end', old('date_end'), ['class' => 'form-control']) }}
  </div>
  <div class="form-group">
    {{ Form::label('color', 'COR') }}
    <div id="cp10" class="input-group colorpicker colorpicker-component colorpicker-element">
      {{ Form::text('color', old('color'), ['class' => 'form-control']) }}
      <span class="input-group-addon"><i style="background-color: rgb(75, 0, 110);"></i></span>
    </div>
  </div>
</div>
<div class="modal-footer">
  <button type="button" class="btn btn-default" data-dismiss="modal">CANCELAR</button>
  <a id="save" type="button" class="btn btn-success">SALVAR</a>
</div>
</div>
</div>
</div>

```

Adicione também, abaixo da tag `<div id="calendar"></div>`, o seguinte código referente a janela modal que será utilizada para editar um evento.

```
<div id="modal-event" class="modal fade" tabindex="-1" data-backdrop="static">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h4>DETALHES DO EVENTO</h4>
      </div>
      <div class="modal-body">
        <div class="form-group">
          {{ Form::label('title_edit', 'TÍTULO') }} {{ Form::text('title_edit', old('title_edit'), ['class' => 'form-control']) }}
        </div>
        <div class="form-group">
          {{ Form::label('date_start_edit', 'INÍCIO EVENTO') }} {{ Form::text('date_start_edit', old('date_start_edit'), ['class' =>
'form-control', 'readonly' => 'true']) }}
        </div>
        <div class="form-group">
          {{ Form::label('time_start_edit', 'HORA INÍCIO') }} {{ Form::text('time_start_edit', old('time_start_edit'), ['class' =>
'form-control']) }}
        </div>
        <div class="form-group">
          {{ Form::label('date_end_edit', 'FIM DO EVENTO') }} {{ Form::text('date_end_edit', old('date_end_edit'), ['class' =>
'form-control']) }}
        </div>
        <div class="form-group">
          {{ Form::label('color_edit', 'COR') }}

          <div id="cp10" class="input-group colorpicker colorpicker-component colorpicker-element">
            {{ Form::text('color_edit', old('color_edit'), ['class' => 'form-control']) }}
            <span class="input-group-addon"><i style="background-color: rgb(75, 0, 110);"></i></span>
          </div>
        </div>
      </div>
      <div class="modal-footer">

        <meta name="csrf-token" content="{{ csrf_token() }}">
        <a id="delete" data-href="{{ url('events') }}" data-id="" class="btn btn-danger"> DELETAR </a>
        <button type="button" class="btn btn-default" data-dismiss="modal"> CANCELAR </button>
        <button id="edit" type="button" class="btn btn-success" data-dismiss="modal"> ATUALIZAR </button>
      </div>
    </div>
  </div>
</div>
```

Para inicializar o *colorpicker* e o *bootstrapMaterialDatePicker*, adicione os seguintes códigos após o final da função javascript `$('#calendar').fullCalendar({...});`

```
$('.colorpicker').colorpicker();
$('#time_start, #time_start_edit').bootstrapMaterialDatePicker({
  date: false,
  shortTime: false,
  format: 'HH:mm:ss'
});
$('#date_end, #date_end_edit').bootstrapMaterialDatePicker({
  date: true,
  shortTime: false,
  format: 'DD-MM-YYYY HH:mm'
});
```

Adicione esses estilos dentro da tag <head> para mostrar a caixa de cor na janela modal.

```
<style>
  .input-group-addon {
    padding: .375rem 1.75rem !important;
  }
  .colorpicker-element .add-on i,
  .colorpicker-element .input-group-addon i {
    position: absolute;
    right: 14px;
  }
  #calendar {
    margin-top: 20px;
    margin-bottom: 30px;
  }
  .fc-event {
    font-size: 1.1em !important;
  }
  .fc-time,
  .fc-title {
    color: #000000 !important;
  }
</style>
```

Agora copie o próximo código e cole no javascript após o final da função \$(document).ready(function() { ... });

Essa função irá ser chamada ao clicar no botão “salvar” da janela modal e fará o envio os dados por Ajax para o controller, onde serão capturados e armazenados no banco de dados.

```
$("#save").on('click', function(e) {
  e.preventDefault();
  var title = $('#title').val();
  var date_start = $('#date_start').val();
  var date_start = dateToUsFormat(date_start);
  var time_start = $('#time_start').val();
  var date_end = $('#date_end').val();
  var date_end = onlyDateToUsFormat(date_end) + ' ' + onlyTime(date_end);
  var color = $('#color').val();
  $.ajaxSetup({
    headers: {
      'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')
    }
  });
  $.ajax({
    url: 'criarEvento',
    data: 'title=' + title + '&start=' + date_start + ' ' + time_start +
      '&end=' + date_end + '&color=' + color,
    type: "POST",
    success: function(json) {
      $('#responsive-model').modal('hide');
      console.log("Criado com sucesso");
      $('#calendar').fullCalendar('refetchEvents');
      clearInputModalSave();
    },
    error: function(json) {
      console.log("Erro ao criar");
    }
  });
});
```

```
});
});
```

Logo abaixo, adicione também essas funções javascript que farão alguns tratamentos necessários para as datas, horários e limpeza dos inputs após o envio dos dados.

```
function clearInputModalSave() {
    $('#title').val("");
    $('#date_start').val("");
    $('#time_start').val("");
    $('#date_end').val("");
    $('#color').val("");
}
function dateToUsFormat(dateStr) {
    var parts = dateStr.split("-");
    var new_date = new Date(parts[2], parts[1] - 1, parts[0]);
    var us_format = moment(new_date).format('YYYY-MM-DD');
    return us_format;
}
function onlyDateToUsFormat(dateStr) {
    var only_date = dateStr.split(' ')[0];
    var parts = only_date.split("-");
    var new_date = new Date(parts[2], parts[1] - 1, parts[0]);
    var us_format = moment(new_date).format('YYYY-MM-DD');
    return us_format;
}
function onlyTime(dateStr) {
    var only_time = dateStr.split(' ')[1];
    return only_time;
}
```

Adicione a seguinte rota no arquivo routes/web.php

```
Route::post('criarEvento', 'EventsController@create');
```

E certifique-se que a rota para events está do seguinte modo: (se não estiver, altere para ficar assim)

```
Route::resource('events', 'EventsController');
```

Acesso o arquivo app/Http/Controllers/EventsController.php e coloque esses código dentro da função create(){...}

```
//Valores recibidos via ajax
$title = $_POST['title'];
$start = $_POST['start'];
$end = $_POST['end'];
$color = $_POST['color'];
$evento= new Event();
$evento->title = $title;
$evento->start = $start;
$evento->end = $end;
$evento->color = $color;
$evento->save();
```

Recarregue a página do calendário, clique no espaço vazio de alguma data e cadastre um novo evento, colocando a data final após a data inicial (pois não tratamos a validação para o cadastro de datas no intervalo inválido), e clique em “salvar”.

10. Editar evento em janela *modal* via Ajax

Para editar um cadastro, adicione a função *eventClick* que será responsável por pegar a instância do evento clicado.

Então, após a linha *events: BASEURL + '/events'* na função javascript, adicione uma “,” (vírgula) e adicione o seguinte código:

```
eventClick: function(event, jsEvent, view) {
    $('#modal-event #delete').attr('data-id', event.id);
    $('#modal-event #edit').attr('data-id', event.id);
    var date_start = moment(event.start).format('DD-MM-YYYY');
    var time_start = moment(event.start).format('hh:mm:ss');
    var date_end = moment(event.end).format('DD-MM-YYYY hh:mm:ss');
    $('#modal-event #title_edit').val(event.title);
    $('#modal-event #date_start_edit').val(date_start);
    $('#modal-event #time_start_edit').val(time_start);
    $('#modal-event #date_end_edit').val(date_end);
    $('#modal-event #color_edit').val(event.color);
    $('#modal-event').modal('show');
}
```

Para responder a ação do botão “Atualizar”, iremos colocar essa função *javascript* que fará o envio dos dados da janela modal para que sejam atualizados. Então, logo abaixo da função de salvar, adicione os seguintes códigos:

```
$("#edit").on('click', function() {
    var x = $(this);
    var id_event = x.attr('data-id');
    var title_edit = $('#title_edit').val();
    var date_start_edit = $('#date_start_edit').val();
    var date_start_edit = dateToUsFormat(date_start_edit);
    var time_start_edit = $('#time_start_edit').val();
    var date_end_edit = $('#date_end_edit').val();
    var date_end_edit = onlyDateToUsFormat(date_end_edit) + ' ' + onlyTime(date_end_edit);
    var color_edit = $('#color_edit').val();
    $.ajaxSetup({
        headers: {
            'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')
        }
    });
    $.ajax({
        url: 'atualizaEvento',
        data: '&id=' + id_event + '&title=' + title_edit + '&start=' + date_start_edit + ' ' + time_start_edit +
            '&end=' + date_end_edit + '&color=' + color_edit,
        type: "POST",
        success: function(json) {
            $('#modal-event').modal('hide');
            console.log("Atualizado com sucesso");
        }
    });
});
```

```

        $('#calendar').fullCalendar('refetchEvents');
    },
    error: function(json) {
        console.log("Erro ao atualizar");
    }
});
});
});

```

Adicione no arquivo de rotas `routes/web.php` a nova rota para a função `update`

```
Route::post('atualizaEvento', 'EventsController@update');
```

No controlador de *Events* em `app/Http/Controllers/EventsController.php` adicione os códigos dentro da função `function update() { ... }`, remova os parâmetros `"Request $request, $id"` se houver, para que os dados sejam capturados e enviados para a atualização no banco de dados.

```

//Valores recibidos via ajax
$id = $_POST['id'];
$title = $_POST['title'];
$start = $_POST['start'];
$end = $_POST['end'];
$color = $_POST['color'];
$evento = Event::find($id);
$evento -> title = $title;
$evento -> start = $start;
$evento -> end = $end;
$evento -> color = $color;
$evento -> save();

```

Agora pode testar.

- Clique em algum evento cadastrado, deverá aparecer uma janela modal com os inputs preenchidos com os dados do evento clicado.
- Altere os dados para testar e clique em "Atualizar".

Obs.: Em alguns casos pode acontecer dos dados não serem enviados pelo ajax no primeiro momento, e, para contornar esse erro que você tratará depois, faça um novo cadastro de teste e logo após tente editar qualquer evento, e deverá funcionar. Caso não ocorra esse erro, ignore essa observação.

11. Deletar evento

O botão "Deletar" já está incluso no janela *modal* do formulário de edição e o que faremos agora é adicionar uma nova *modal* que será para capturar a confirmação ou o cancelamento do evento Deletar.

Então, adicione esses códigos no topo da página, logo após ao início da tag `<div class="container"> ...`

```

<div class="modal fade" id="modalDelete" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true" style="z-index: 1051 !important;">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModalLabel">Deletar</h5>
        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
    </div>
  </div>
</div>

```

```

        </button>
    </div>
    <div class="modal-body">
        Tem certeza que deseja deletar ?
    </div>
    <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-dismiss="modal">Cancelar</button>
        <button id="yes_delete" type="button" class="btn btn-danger">Deletar</button>
    </div>
</div>
</div>
</div>

```

Agora vamos adicionar a função *javascript* que fará o envio dos dados via *Ajax* para que sejam deletados do banco de dados. Então, adicione logo abaixo do final da função `$(document).ready(function() {...})`; os seguintes códigos com a função que captura a ação do *click* no botão “Deletar”.

```

$('#delete').on('click', function() {
    var x = $(this);
    var id_event = x.attr('data-id');
    var delete_url = x.attr('data-href') + '/' + x.attr('data-id');
    $.ajaxSetup({
        headers: {
            'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')
        }
    });
    $('#modalDelete').modal('show');
    $('#yes_delete').on('click', function(e) {
        $('#modalDelete').modal('hide');
        e.preventDefault();
        $.ajax({
            url: delete_url,
            type: 'DELETE',
            success: function(result) {
                $('#modal-event').modal('hide');
                $('#calendar').fullCalendar('removeEvents', id_event);
            },
            error: function(result) {
                $('#modal-event').modal('hide');
            }
        });
    });
});

```

No controller de Events, adicione os seguintes códigos dentro da função `destroy($id)` :

```

$event = Event::find($id);
if ($event == null) {
    return Response() -> json([
        'message' => 'error delete.'
    ]);
}
$event -> delete();
return Response() -> json([
    'message' => 'success delete.'
]);

```

Agora pode testar.

- Clique em algum evento cadastrado;
- Na janela *modal* clique em “Deletar” e deverá aparecer outra janela *modal* por cima dessa, para que seja confirmada ou cancelada a ação;
- Clique em “Deletar” e as janelas *modal* deverão desaparecer e o evento deletado deverá ter sido excluído do calendário.

12. Atualizar posição do evento ao ser arrastado e solto em nova data

A seguinte função fará a captura de quando um evento é arrastado e solto em outra data. Faremos a programação do envio dos dados para atualizar a nova posição do evento. Para isso, adicione uma vírgula após o final da função *eventClick*: *function ...* e adicione os seguintes códigos:

```
eventDrop: function(event, delta) {
  var id_event = event.id;
  var title_edit = event.title;
  var date_start_edit = moment(event.start).format('YYYY-MM-DD');
  var time_start_edit = moment(event.start).format('hh:mm:ss');
  var date_end_edit = moment(event.end).format('YYYY-MM-DD hh:mm:ss');
  var color_edit = event.color;
  $.ajaxSetup({
    headers: {
      'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')
    }
  });
  $.ajax({
    url: 'atualizaEvento',
    data: '&id=' + id_event + '&title=' + title_edit + '&start=' + date_start_edit + ' ' + time_start_edit +
      '&end=' + date_end_edit + '&color=' + color_edit,
    type: "POST",
    success: function(json) {
      $('#modal-event').modal('hide');
      console.log("Atualizado com sucesso");
      $('#calendar').fullCalendar('refetchEvents');
    },
    error: function(json) {
      console.log("Erro ao atualizar");
    }
  });
}
```


Pronto, agora faça o teste:

- Clique, segure e arraste algum evento para uma nova data e solte;
- Atualize a página pressionando F5;
- O evento alterado deverá permanecer na mesma posição onde foi deixado e com o mesmo intervalo de datas cadastrados anteriormente.

MUITO OBRIGADO !

BONS ESTUDOS.

Naiguel Corrêa dos Santos

UPF - Universidade de Passo Fundo

ANÁLISE E DESENVOLVIMENTO DE SISTEMAS - 6º Semestre

TÓPICOS ESPECIAIS EM DESENVOLVIMENTO DE SISTEMAS II

Professor: GUILHERME AFONSO MADALOZZO