Curso PHP

Do XLSX ao CMS (aula 20: final)

Calendário
FullCalendar

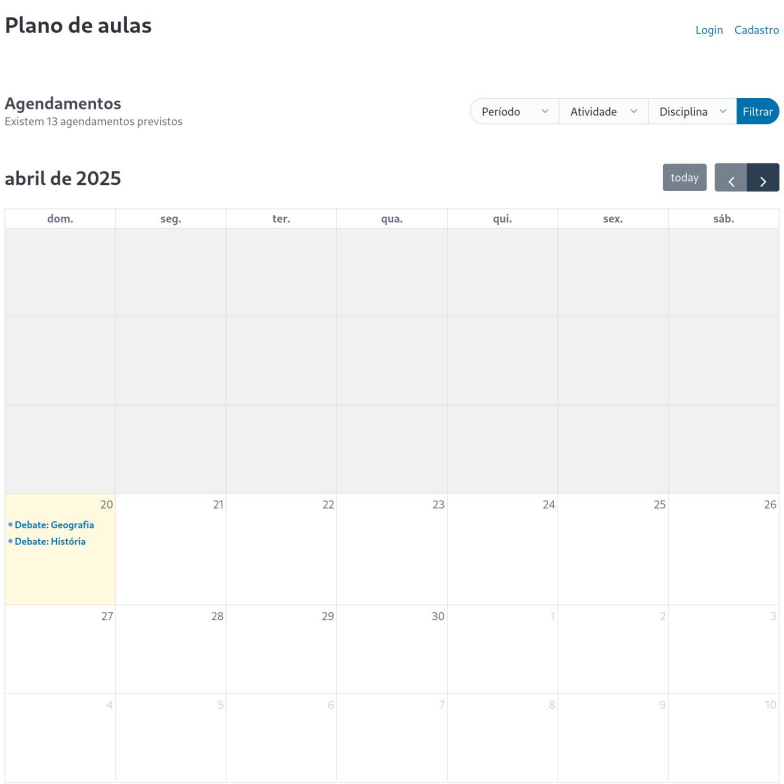
Full Calendar

A biblioteca FullCalendar é, como o nome sugere, um biblioteca para manipular calendários.

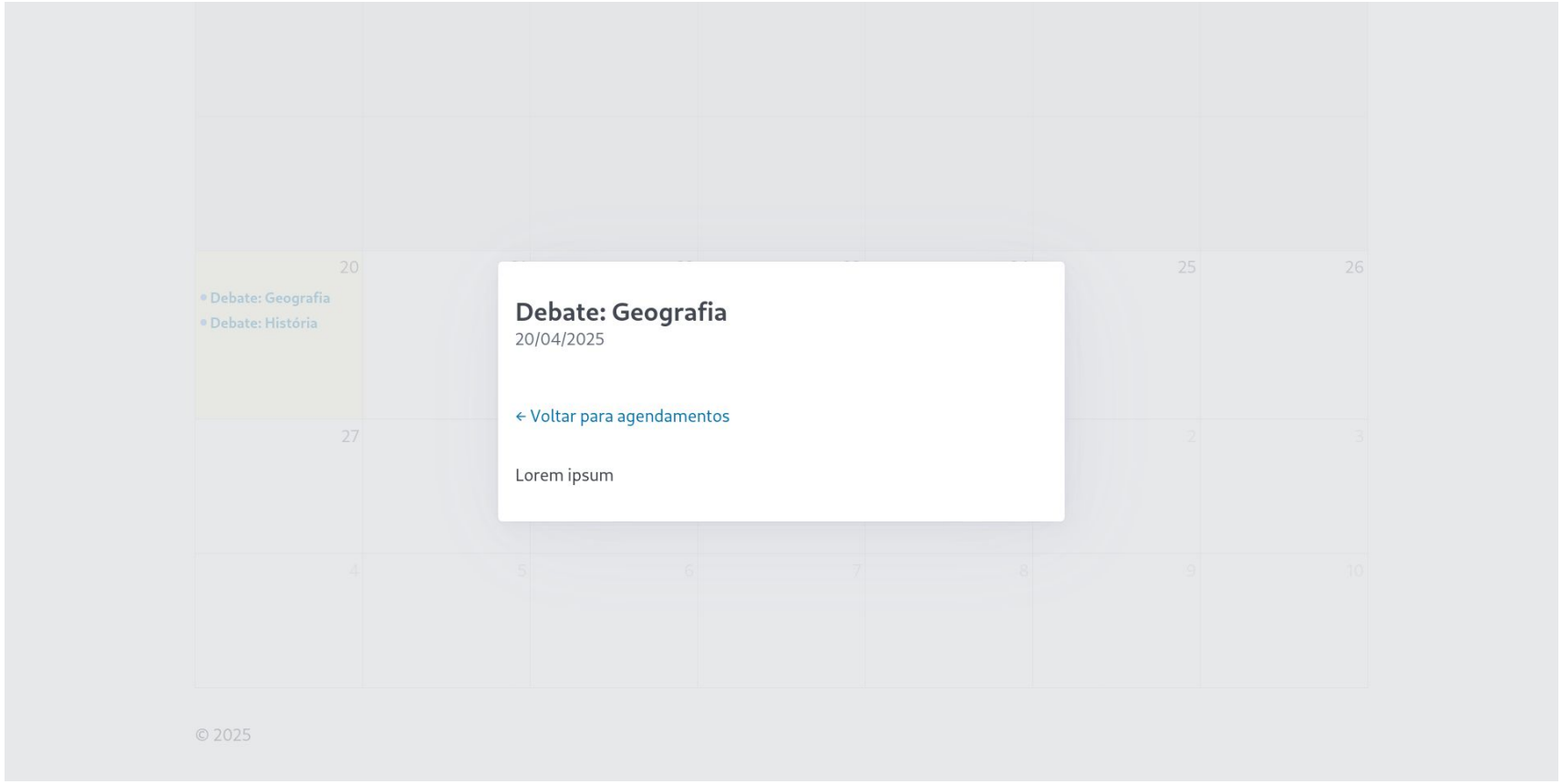
Muito simplificada, contém funções que são comuns na implementação desse tipo de serviço. Além disso, permite usar plugins, que pode estender as funcionalidades.

```
// Cria uma nova instância do calendário com as configurações desejadas
const calendar = new FullCalendar.Calendar(container, {
    // Passa os eventos para o calendário
    events,

    // Define a visualização inicial como um calendário mensal em grade
    initialView: 'dayGridMonth',

    // Define o idioma do calendário
    locale: this.env.app_locale,

    // Define o fuso horário do calendário
    timeZone: this.env.app_timezone,

    // Oculta o fuso-horário (timezone) nos eventos exibidos
    displayEventTime: false,

    // Define o intervalo de datas válidas para navegação no calendário
    validRange: {
        start: minDate,
        end: maxDate
    },

    // Define o comportamento ao clicar em um evento
    async eventClick({ event, jsEvent }) {
        // ...
    }
})

// Move o calendário para a menor data (início dos eventos)
calendar.gotoDate(minDate)

// Renderiza o calendário na tela
calendar.render()
```

Configuração básica da biblioteca FullCalendar



Exibindo agendamentos usando FullCalendar



Ao clicar em um agendamento

Anexos
Uploading

Anexos

A anexação de arquivos é algo relativamente trivial em sistemas. É um campo onde o usuário seleciona um arquivo, clica em enviar e este arquivo é conectado ao recurso indicado, porém, a implementação requer certos cuidados, como validação de tipos e extensões aceitas.

É importante frisar que um arquivo com a extensão .JPG não é sempre uma imagem, para isto usamos um *MIME type*, contudo, não nos limitamos a somente isso. De mesma forma, um arquivo .TXT pode não ser um arquivo de texto plano, pois, dependendo de como é interpretado, pode ser um executável.

Leia mais em: https://developer.mozilla.org/docs/Web/HTTP/Guides/MIME_types

Tipos Suportados
Mime-type

Definindo tipos suportados

Nosso arquivo **config/uploads.php** contém a chave **mimes**, que mapeia as extensões aceitas pelo MIME.

Isto é importante, pois alguns MIME suportam mais que uma extensão, especialmente imagens.

```php
<?php

return [
    'directory' => PROJECT_ROOT.'/uploads',
    'mimes' => [
        // documentos
        'application/msword' => ['doc'],
        'application/pdf' => ['pdf'],
        'application/rtf' => ['rtf'],
        'application/vnd.ms-excel' => ['xls'],
        'application/vnd.ms-powerpoint' => ['ppt'],
        'application/vnd.oasis.opendocument.text' => ['odt'],
        'application/vnd.openxmlformats-officedocument.presentationml.presentation' => ['pptx'],
        'application/vnd.openxmlformats-officedocument.spreadsheetml.sheet' => ['xlsx'],
        'application/vnd.openxmlformats-officedocument.wordprocessingml.document' => ['docx'],
        'text/plain' => ['txt'],

        // imagens
        'image/bmp' => ['bmp'],
        'image/gif' => ['gif'],
        'image/jpeg' => ['jpg', 'jpeg'],
        'image/png' => ['png'],
        'image/svg+xml' => ['svg'],
        'image/tiff' => ['tif', 'tiff'],
        'image/webp' => ['webp'],
    ]
];
```

Anexos

Anexos: permissão

O processo de anexar é, como dito antes, algo que requer cuidados especiais, pois está permitindo que o usuário envie arquivos para o servidor, que irão ser interpretados pelo sistema, abrindo brechas para arquivos maliciosos.

```php
public function salvar(Request $request): RedirectResponse
{
    if (! $request->validate(Agendamento::rules())) {
        return redirectRoute('cadastrar_agendamento');
    }

    $agendamento = Agendamento::create($request->validated);

    if (hasPermission(Permission::INSERIR_ANEXOS)) {
        $agendamento->inserirAnexos($request->validated['anexos']);
    }

    notifyMany(Usuario::pluck('id')->toArray(), AgendamentoCadastrado::class, [
        'agendamento' => $agendamento->id,
    ]);

    return redirectRoute('agendamentos');
}
```

Anexos: regras

Nosso sistema espera que os anexos sejam enviados como um array.

Depois, um a um, verificamos se o MIME é aceito e se a extensão é aceita pelo MIME. Além disso, verificamos se o tamanho do arquivo é aceito pelas configurações do PHP.

Este é o ponto de tratamento dos uploads.

```php
public static function rules(): array
{
    return [
        'atividade_id' => Validator::intVal()->callback(Atividade::exists(...)),
        'disciplina_id' => Validator::intVal()->callback(Disciplina::exists(...)),
        'conteudo' => Validator::notEmpty()->length(1, 512),
        'data' => Validator::date('Y-m-d'),
        'anexos:files' => Validator::nullable(
            Validator::arrayType()->each(
                Validator::file()->callback(function (UploadedFile $file) {
                    $mime = $file->getMimeType();
                    $ext = strtolower($file->getClientOriginalExtension());
                    $allowedTypes = Anexo::allowedMimes();

                    return $file->isValid()
                        && isset($allowedTypes[$mime])
                        && in_array($ext, $allowedTypes[$mime], true)
                        && $file->getSize() <= UploadedFile::getMaxFilesize();
                })
            )
        ),
    ];
}
```

Anexos: inserir

Observe que existe o diretório **outros**. Ele existe para que seja dado suporte a arquivos em outros formatos, ampliando o suporte, porém nada impede de criar diretórios correspondentes.

Por fim, registramos os metadados no banco de dados, enquanto o arquivo é enviado para o diretório.

```php
/**
 * @param array<\Symfony\Component\HttpFoundation\File\UploadedFile> $uploads
 */
public function inserirAnexos(array $uploads): void
{
    $anexos = array_map(function (UploadedFile $anexo) {
        $tipo = $anexo->getMimeType();
        $diretorio = match (true) {
            str_starts_with($tipo, 'application/') ||
            str_starts_with($tipo, 'text/') => 'documentos',
            str_starts_with($tipo, 'image/') => 'imagens',
            str_starts_with($tipo, 'video/') => 'videos',
            default => 'outros'
        };
        $nome_original = $anexo->getClientOriginalName();
        $extensao = $anexo->getClientOriginalExtension();
        $nome = uniqid(more_entropy: true).".{$extensao}";
        $caminho = "{$diretorio}/{$nome}";
        $anexo = $anexo->move(Anexo::uploadsDir($diretorio), $nome);
        $tamanho = number_format($anexo->getSize() / 1024 / 1024, 2).' MB';

        return compact('nome_original', 'caminho', 'tipo', 'extensao', 'tamanho');
    }, $uploads);

    $this->anexos()->upsert($anexos, 'id');
}
```

Anexos: ver

Uma vez que o arquivos está salvo, podemos o enviar ao navegador através da **BinaryFileResponse**, que é a leitura e o envio total do arquivo.

Isto permite aplicar *middlewares* à requisição e forçar comportamentos do usuários. Além disso, o usuário não tem acesso direto ao arquivo.

```php
class AnexoController
{
    public function ver(Anexo $anexo): BinaryFileResponse|Response
    {
        $file = Anexo::uploadsDir($anexo->caminho);
        $filename = $anexo->nome_original;
        $contentType = $anexo->tipo;

        if (preg_match('/(application|image|text)\//', $contentType)) {
            $response = new BinaryFileResponse($file);

            $response->headers->set('content-type', $contentType);
            $response->setContentDisposition(ResponseHeaderBag::DISPOSITION_INLINE, $filename);

            return $response;
        }

        return responseError(Response::HTTP_NO_CONTENT);
    }
}
```

Anexos: listar

Em nosso sistema, os anexos são os materiais de aula.

Permitir que o usuário liste todos, sem navegar pelo calendário, torna a busca por um material mais fácil, pois, aplicando filtros à pesquisa, podemos ir “direto ao ponto”.

Plano de aulas

admin@example.com

Agendamentos **Anexos** Períodos Disciplinas Atividades Usuários Logs Jobs

Anexos

Todo o material de aula

Período | Disciplin: | Filtrar

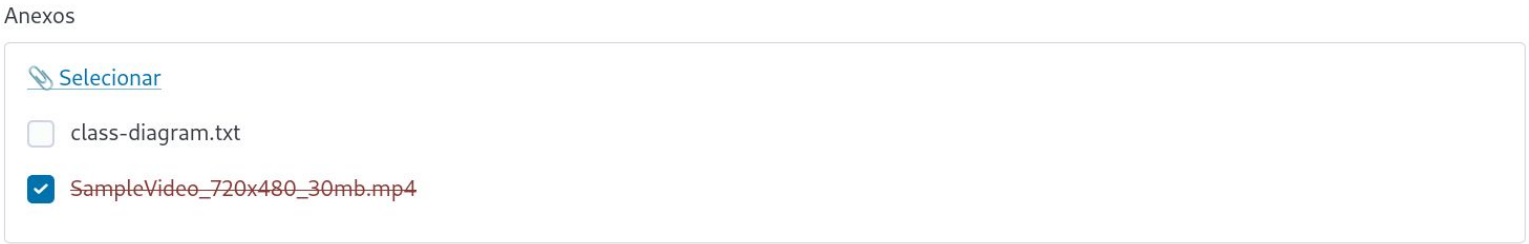
Anexo
aula.txt Geografia (2025.1) — 0.06 MB
class-diagram.txt História (2025.1) — 0.00 MB
SampleVideo_720x480_30mb.mp4 História (2025.1) — 30.09 MB

← Anterior 1 Próxima →

Listagem de anexos

Anexos: excluir

Para excluir um anexo, basta marcar a caixa dele na listagem de anexos do agendamento e salvar.



Enviar por email
Attachments

Anexos por email

Uma característica dos anexos é permitir o acesso a eles.

Pensando em facilidade, podemos enviar os anexos de um agendamento por email durante o processo de notificação.

Nossa regra de negócios fica estabelecida que:

- Somente serão enviados anexos durante a criação.
- Não serão enviados anexos durante a atualização.

```php
class AgendamentoCadastrado
{
    use Notifiable;

    public function __invoke(Usuario $recipient, Agendamento $agendamento): void
    {
        $atividade = ucfirst($agendamento->atividade->nome);
        $disciplina = ucwords($agendamento->disciplina->nome);
        $dm = $agendamento->data->format('d/m');
        $id = $agendamento->id;
        $anexos = $agendamento->anexos()->get()->map(fn (Anexo $anexo) => [
            'name' => $anexo->nome_original,
            'path' => Anexo::uploadsDir($anexo->caminho),
            'contentType' => $anexo->tipo,
        ])->toArray();

        $data = [
            'recipient' => $recipient,
            'subject' => "Agendamento #{$id}: {$atividade} de {$disciplina} para {$dm}",
            'cta' => Link::to(route('ver_agendamento', $agendamento->id), 'Ver agendamento'),
            'atividade' => $atividade,
            'disciplina' => $disciplina,
            'dm' => $dm,
            'attachments' => $anexos,
        ];

        $this->viaEmail('notifications/email/agendamento.twig', $data);
    }
}
```

Agendamento cadastrado

```php
trait Notifiable
{
    private function viaEmail(string $view, array $data): void
    {
        $transport = Transport::fromDsn(env('MAILER_DSN'));
        $mailer = new Mailer($transport);
        $message = new Email;
        $body = View::render($view, $data);
        $attachments = $data['attachments'] ?? [];

        $message->from(env('MAILER_FROM'));
        $message->to($data['recipient']->email);
        $message->subject($data['subject']);
        $message->html($body);

        foreach ($attachments as $attachment) {
            $message->attachFromPath(
                $attachment['path'],
                $attachment['name'],
                $attachment['contentType']
            );
        }

        $mailer->send($message);
    }
}
```

Adicionando anexos à mensagem

Streaming
Conteúdo parcial

Streaming

Transmitir os dados com um cabeçalho de conteúdo parcial é, como o nome sugere, transmitir os dados em partes. isto permite o navegador receber arquivos longos, como áudio e vídeo sem sobrecarregar a rede nem o servidor.

```php
class AnexoController
{
    public function ver(Request $request, Anexo $anexo): BinaryFileResponse|Response
    {
        $file = Anexo::uploadsDir($anexo->caminho);
        $filename = $anexo->nome_original;
        $contentType = $anexo->tipo;

        if (preg_match('/(application|image|text)\//', $contentType)) {
            // ...
        }

        if (preg_match('/(audio|video)\//', $contentType)) {
            // ...

            $response = new StreamedResponse(function () use ($file, $start, $length) {
                // ...
            }, Response::HTTP_PARTIAL_CONTENT);

            $response->headers->set('Content-Disposition', "inline; filename=\"{$filename}\"");
            $response->headers->set('Content-Type', $contentType);
            $response->headers->set('Content-Length', $length);
            $response->headers->set('Content-Range', "bytes {$start}-{$end}/{$size}");
            $response->headers->set('Accept-Ranges', 'bytes');

            return $response;
        }

        return responseError(Response::HTTP_NO_CONTENT);
    }
}
```

Aplicando streaming à exibição dos anexos

Próximos passos

Aplicar aprimoramentos nas regras de negócio.

Se você acompanhou este curso até aqui, aprendeu sobre os comportamentos básicos de um CMS, que podem ter as regras de negócio aprimoradas para se elevar um um ERP, CRM, SAP e tantas outras siglas que existem para gestão de conteúdo e materiais.

Grandes sistemas são feitos de pequenas partes, em que as partes são conceitos e uso de lógica. Unir tudo isso em um lugar, é o desafio que a programação faz superar.

end-of-file