### **Sajic Fullstack: Uma Aplicação para Gerenciamento de Eventos com Laravel e InertiaJS**

**Autores:** João Otávio, Natan Barbosa, Marcelo, Lorenzo, Gabriel Amaral,

**Professora:** Bruna G. Ribeiro¹

¹Centro Universitário Senac RS (UniSenac) Rua Gonçalves Chaves, 602 – Centro – Pelotas – 96015-560 – RS - Brasil   
[*joaootavio192@gmail.com*](mailto:joaootavio192@gmail.com)*, amaraldesouza9@gmail.com, natanpederzolli@gmail.com, lorenzodequadrosgoncalves@gmail.com, marcelo.desenvolvedor.contato@gmail.com, brgribeiro@senacrs.com.br*

**Resumo.** Este artigo detalha o desenvolvimento do Sajic Fullstack, uma aplicação web para gerenciamento de inscrições, controle de presença e certificados de palestras. O sistema utiliza Laravel no back-end e uma arquitetura de Single Page Application (SPA) com InertiaJS e React no front-end. A solução oferece funcionalidades distintas para usuários e administradores, automatizando processos-chave como o check-in e o envio de certificados. A arquitetura contempla ainda a renderização do lado do servidor (SSR) de forma opcional para otimização de SEO.

### **1. Introdução**

O gerenciamento de eventos acadêmicos, como o Sajic, envolve desafios logísticos complexos, incluindo o controle de inscrições em palestras, o registro de presença e a emissão de certificados. Para otimizar esses processos, foi desenvolvida a aplicação Sajic Fullstack, um sistema web completo que automatiza todo o fluxo de participação no evento. O objetivo central do projeto é fornecer uma plataforma robusta e intuitiva para usuários e administradores, simplificando a gestão e melhorando a experiência dos participantes.

O fluxo principal do sistema foi desenhado para ser eficiente: o usuário se inscreve nas palestras de seu interesse; no dia do evento, um administrador realiza o check-in dos presentes; e, logo após a confirmação de presença, o certificado correspondente é gerado e enviado automaticamente por e-mail, ficando também disponível no perfil do usuário. A plataforma oferece flexibilidade, permitindo que os participantes cancelem e se reinscrevam em atividades, desde que haja vagas.

O sistema opera com dois perfis de acesso:

* **Usuário Comum:** Pode se inscrever e cancelar inscrições em palestras, além de acessar seus certificados a qualquer momento.
* **Admin:** Possui controle total sobre o evento, podendo criar, editar e remover palestras e palestrantes, realizar o check-in de participantes, reabrir palestras que atingiram a lotação máxima e visualizar uma lista pública de presenças para fins de controle institucional.

for bottom margin and 3.0 cm for lateral margins, without headers or footers. The main font must be Times, 12 point nominal size, with 6 points of space before each paragraph. Page numbers must be suppressed (XAVIER, 2021).

### **2. Metodologia**

O projeto foi desenvolvido utilizando uma stack de tecnologias modernas, visando performance, escalabilidade e uma boa experiência de desenvolvimento. A metodologia de implementação seguiu as melhores práticas para aplicações web fullstack, separando as responsabilidades entre o back-end e o front-end, conectados pela ponte do InertiaJS.

As tecnologias utilizadas foram:

* **Backend:** Framework Laravel (PHP), para a lógica de negócio, APIs e autenticação.
* **SPA Bridge:** InertiaJS, para criar uma aplicação de página única (SPA) totalmente reativa sem a complexidade de uma API tradicional.
* **Frontend:** Biblioteca React com TypeScript, para a construção de interfaces de usuário dinâmicas e tipadas.
* **Build Tools:** Vite e Node.js, para compilação e otimização dos assets do front-end.
* **Banco de Dados:** Neon (PostgreSQL), um serviço de banco de dados serverless e gerenciado.
* **Serviços Auxiliares:** AWS Simple Email Service (SES) para o envio de e-mails, Ziggy para o uso de rotas nomeadas do Laravel no JavaScript, e uma biblioteca para geração de PDFs sob demanda.
* **Infraestrutura:** A aplicação é hospedada na AWS EC2 (Amazon Linux), com Nginx e FastCGI para servir a aplicação PHP e Cloudflare para gerenciamento de DNS e proxy reverso.

O processo de desenvolvimento foi estruturado com base na instalação de dependências via Composer (PHP) e NPM (JS), configuração de ambiente através de um arquivo .env, execução de migrações de banco de dados com php artisan migrate e compilação de assets com npm run dev. Para a validação da qualidade e integridade do código, foram utilizados testes automatizados com PHPUnit, executados através do comando php artisan test.  
  
Para organização de tarefas foi utilizado Jira com metodologia kanban e algumas sprints para entregas continuas durante o semestre

### **3. Arquitetura e Implementação do Sistema**

O sistema foi arquitetado de forma modular, aproveitando os recursos do framework Laravel e a flexibilidade do React. A seguir, detalha-se a estrutura de implementação.

**3.1. Estrutura do Back-End**

O back-end é organizado seguindo o padrão do Laravel. Na pasta app/, os Models/ (Eloquent) representam as entidades do banco de dados (usuários, palestras, inscrições). A lógica de negócio está nos Http/Controllers/, como o LectureController (gerencia inscrições, check-in) e o UserController (gerencia certificados e cancelamentos). O Http/Middleware/ contém as regras de negócio transversais, como o HandleInertiaRequests, que compartilha dados globais com o front-end, e os middlewares de autenticação, que protegem as rotas. As rotas são segmentadas em arquivos (web.php, admin.php, auth.php, settings.php) para melhor organização.

**3.2. Estrutura do Front-End**

O front-end é uma Single Page Application (SPA) construída em resources/js/. O ponto de entrada é o arquivo app.tsx, que inicializa o React e o Inertia. A pasta pages/ contém os componentes React que representam cada página da aplicação, mapeados diretamente a partir das rotas do Laravel. Componentes reutilizáveis, como botões, modais e cards, ficam em components/, e os layouts globais da aplicação (ex: cabeçalho e rodapé) são definidos em layouts/. Para otimização de SEO, foi implementado um Server-Side Rendering (SSR) opcional, cujo ponto de entrada é ssr.tsx e o build é feito com npm run build:ssr.

**3.3. Integração e Fluxo de Dados**

A integração entre back-end e front-end é o pilar do sistema, realizada pelo InertiaJS. Os controllers do Laravel, em vez de retornarem JSON, utilizam a função Inertia::render('NomeDaPagina', ['dados' => $dados]). Isso renderiza o componente React correspondente em resources/js/pages/NomeDaPagina.tsx, passando os dados do back-end como *props*.

Dados globais, como o usuário autenticado (auth.user), mensagens de feedback (flash) e as rotas nomeadas (via Ziggy), são compartilhados com todas as páginas através do middleware HandleInertiaRequests.php. No front-end, esses dados são facilmente acessados em qualquer componente através do hook usePage() do Inertia, garantindo um estado consistente e reativo em toda a aplicação. A autenticação segue o padrão do Laravel, utilizando sessions e cookies, com rotas protegidas por middleware.

### **4. Conclusão e Trabalhos Futuros**

A aplicação Sajic Fullstack demonstrou ser uma solução eficaz e completa para o gerenciamento do evento, automatizando tarefas críticas e fornecendo uma interface clara tanto para os participantes quanto para a administração. A escolha da stack tecnológica, com Laravel e Inertia/React, permitiu um desenvolvimento rápido e uma aplicação final performática e de fácil manutenção. O sistema cumpriu com sucesso todos os requisitos definidos, desde a inscrição online até a entrega automatizada de certificados.

Como trabalhos futuros, vislumbra-se a implementação de novas funcionalidades para enriquecer ainda mais a plataforma. Entre as possibilidades, destacam-se:

* Um sistema de notificações em tempo real (utilizando Laravel Reverb ou WebSockets) para avisar os usuários sobre o início das palestras.
* A integração com gateways de pagamento para gerenciar eventos com custos de inscrição.
* Um dashboard administrativo com gráficos e estatísticas sobre as inscrições e a participação no evento.
* O desenvolvimento de uma Progressive Web App (PWA) para melhorar a experiência em dispositivos móveis.

**5. Referências**

Sajic 2024