Sistema de Gerenciamento de Certificados Baseado em Microsserviços: Foco no Cliente Web Responsivo

Lucas N. Bertoldi, Alexandro S. Silva, Pablo F. Matos

Instituto Federal de Educação, Ciência e Tecnologia da Bahia (IFBA)

Av. Avenida Sérgio Vieira Melo, 3.150, Zabelê – 45.078-900
Vitória da Conquista – BA – Brasil

lucasn.bertoldi@gmail.com, {alexandrossilva, pablofmatos}@ifba.edu.br

Abstract. *One of the main goals of events is to spread knowledge. In this context, it is necessary to generate and make secure certificates available to attest the participation of participants in the activities of these events. There are some web event management platforms that assist event coordinators in performing this task. However, the vast majority of the platforms are not specialized in generating, validating and providing certificates. Some specialized platforms have few options for configuring the certificate template, or were built with technologies that directly affect the user's usability, the deployment and the maintenance of the system. The purpose of this work is to refactor and improve the user interface of the platform to generate and validate certificates of the Instituto Federal da Bahia (IFBA), Campus Vitória da Conquista/BA. Thus, the microservices architecture was used, with the purpose to facilitate the separation of the Web client from the back-end of the application. This client was built using the Single Page Application concept, with the help of the Next.JS framework and the React library. A microservice called API Gateway was also developed, which centralizes the client's requests to other microservices. The new user interface was built using design and placement patterns to provide usability improvements. Furthermore, problems with the previous application were solved, and new features were developed. The Web Client, API Gateway and the other microservices responsible for running the application are available in a public repository will be kept under an open source license.*

Resumo. *Um dos principais objetivos dos eventos é difundir conhecimento. Nestes casos, são necessárias a emissão e a disponibilização de certificados seguros para atestar a participação dos participantes nas atividades que ocorrem nos eventos. Para auxiliar os coordenadores de eventos na execução dessa tarefa, existem algumas plataformas Web de gerenciamento de eventos. Porém, a grande maioria não é especializada na geração, validação e disponibilização de certificados. Algumas plataformas especializadas possuem poucas opções para configuração do modelo do certificado ou foram construídas com tecnologias que afetam diretamente a usabilidade do usuário, a implantação e a manutenção do sistema. A proposta deste trabalho é reconstruir e aprimorar a interface da plataforma de emissão e validação de certificados do IFBA Campus Vitória da Conquista. Para isto, foi utilizada a arquitetura de microsserviços a fim de facilitar a separação do cliente Web do restante da aplicação. Este cliente foi construído utilizando o conceito de Aplicativo de Página Única, com o auxílio do framework Next.JS e da*

biblioteca React. Também foi desenvolvido um microsserviço chamado API Gateway que centraliza o consumo das requisições do cliente com os outros microsserviços. A nova interface foi construída utilizando padrões de design e posicionamento para fornecer melhorias de usabilidade. Além disso, foram resolvidos problemas da aplicação anterior, e desenvolvidas novas funcionalidades. O Cliente Web, a API Gateway e os outros microsserviços responsáveis pelo funcionamento da aplicação estão disponíveis em um repositório público e são open source.

1. Introdução

Instituições promovem regularmente atividades em eventos para difundir conhecimento. Professores, alunos ou organizadores precisam de uma forma segura de atestar a participação nas atividades que ocorrem nesses eventos. Cerificados geralmente são gerados em eventos para resolver esse problema. Entretanto, fazer essa gerência de certificados e disponibilizá-los aos participantes podem não ser tarefas tão simples e podem custar muito tempo. Existem plataformas online que têm o papel de auxiliar nessa função. Porém, a sua grande maioria é focada na inscrição, na divulgação e no pagamento do evento [Doity 2021], [Even3 2021] e [Sympla 2021]. A falta de especialização no gerenciamento, disponibilização e validação dos certificados, além de trazer uma possível dificuldade para novos usuários, pode fazer com que haja poucas opções para configuração da arte gráfica e texto dessa geração. Também existem limitações de usabilidade em relação à disponibilização dos certificados, a exemplo da plataforma Gerar Certificado (2021).

Uma das opções especialistas em gerenciamento de certificados é o sistema do Instituto Federal de Educação, Ciência e Tecnologia da Bahia (IFBA) *Campus* Vitória da Conquista. Ramos *et al.* (2018) definem os objetivos desse projeto como sendo um *software open source* para organizadores de eventos criar, editar e gerenciar a emissão e validação de certificados de forma rápida e prática. Porém, essa plataforma foi desenvolvida com tecnologias que não satisfazem as demandas dos usuários atuais da Internet. Uns dos principais problemas é a utilização de um pré-processador PHP que torna a navegação lenta em alguns momentos. Segundo Mesquita (2013), há uma boa usabilidade de acordo com a Norma ISO 9241-11 quando usuários atingem os seus objetivos com eficácia, eficiência e satisfação. Em relação a esses conceitos, o sistema também possui problemas. Além disso, com o passar do tempo, novos requisitos importantes foram identificados. A sua arquitetura monolítica faz com que ocorram algumas adversidades, como inconsistência em adição ou atualização de dependência, necessidade de reinicialização após implantação de pequenas mudanças, desperdício de recursos por necessidades diferentes entre módulos, limitação de escalabilidade e vinculação à única linguagem [Dragoni *et al.* 2017]. Esses fatores afetam a manutenção do projeto, dificultando o desenvolvimento por diferentes equipes.

Um dos objetivos deste trabalho é construir um novo cliente Web responsivo utilizando o *framework* Next.JS (2021). Assim, é possível criar um *Aplicativo de Página Única*, tornando, assim, a navegação mais dinâmica e rápida. A biblioteca React (2021), que é a base desse *framework*, contribui para a reutilização de código e torna a tarefa de construir uma interface dinâmica mais fácil. Deste modo, há a possibilidade de reconstruir o projeto, desenvolver novos requisitos e implementar conceitos de usabilidade de uma maneira mais ágil. Outra importante mudança foi o desenvolvimento da arquitetura baseada em microsserviços, que acarretou na

necessidade de construção de uma API Gateway, a qual é responsável por centralizar a comunicação entre os microsserviços e o Cliente Web.

Nesse sentido, este artigo está organizado como segue. Na Seção 2 está descrita a fundamentação teórica que auxiliou no desenvolvimento deste trabalho. Na Seção 3 é apresentada a plataforma de Certificados do IFBA *Campus* Vitória da Conquista na versão monolítica, outras plataformas, e o comparativo entre elas. Os materiais e métodos são evidenciados na Seção 4. Os resultados e as considerações finais estão descritos, respectivamente, nas Seções 5 e 6.

2. Fundamentação Teórica

Nesta seção é apresentada a fundamentação teórica necessária para a compreensão do desenvolvimento deste trabalho. Nas Seções 2.1 e 2.2 são descritas as arquiteturas monolítica e de microsserviços, enquanto na Seção 2.3 é demonstrado o papel e os benefícios de uma API Gateway. As tecnologias empregadas em páginas Web são evidenciadas na Seção 2.4 e um exemplo de conceito de construção em páginas únicas é explicado na Seção 2.5. Por fim, na Seção 2.6 são definidos usabilidade e *design* responsivo.

2.1. Arquitetura Monolítica

Os módulos de um *software* são abstrações que têm o objetivo de reduzir sua complexidade. Uma aplicação é determinada como monolítica quando esses módulos são construídos em um único artefato de execução e este não tem capacidade de ser executado independentemente [Dragoni *et al.* 2017]. Segundo Penna (2021), uma aplicação monolítica é mais simples de desenvolver, testar, implantar e escalar. Entretanto, essa arquitetura pode causar alguns problemas, que podem ser evidenciados quando a aplicação se tornar mais extensa e complexa. Inconsistência em adição ou atualização de dependência, necessidade de reinicialização após pequenas mudanças, desperdício de recursos por necessidades diferentes entre módulos, limitação de escalabilidade e vinculação à única linguagem são algumas adversidades desse padrão de desenvolvimento. Uma opção para resolver esses problemas é a arquitetura de microsserviços.

2.2. Arquitetura de Microsserviços

A arquitetura de microsserviços, por sua vez, propõe um conceito diferente do modelo monolítico, que é o desmembramento dos módulos em componentes mínimos e independentes. Seus princípios auxiliam gerentes de projetos e desenvolvedores a focarem na implementação e teste de poucas funcionalidades coesas. Isto resolve algumas das dificuldades enfrentadas pela arquitetura monolítica. Por exemplo, por ter uma quantidade limitada de funcionalidades, estas podem ser testadas e corrigidas de forma mais ágil e isolada. Além disso, é possível efetuar mudanças graduais de novas versões, o que descarta a possibilidade de reiniciar o sistema por completo. Também fornecem uma maior liberdade para a alocação de recursos e conveniência de implantar e dispor de instâncias com base na carga.

Os microsserviços oferecem a autonomia de escolha de diferentes linguagens, *frameworks*, bancos de dados, dentre outros. Na Figura 1 é mostrado um comparativo de um cenário de exemplo, referente à alocação de banco de dados para uma aplicação com arquitetura monolítica e uma outra de microsserviços. Em uma rede de microsserviços, a única exigência é a definição da tecnologia de comunicação [Dragoni

et al. 2017]. Um exemplo de tecnologia que pode ser utilizada para efetuar essa comunicação é o protocolo HTTP.

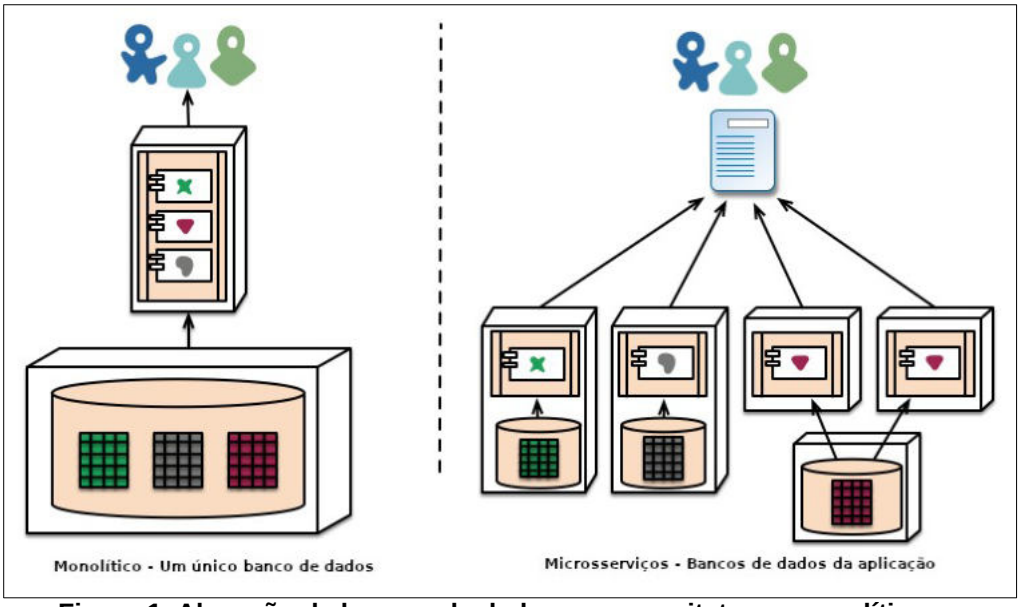

Figura 1. Alocação de bancos de dados para arquiteturas monolíticas e baseadas em microsserviços. Fonte: Adaptado de Lewis (2014)

2.3. API Gateway

A Interface de Programação de Aplicações (*Application Programming Interface* - API) é um conceito que define padrões e normas para o acesso a uma determinada aplicação. Uma API Gateway é uma espécie de filtro de entrada que encaminha as chamadas para um endereço mais apropriado [Negri 2021]. O seu uso facilita a possibilidade de limitação, bloqueio de acesso e monitoramento. Além disso, essa centralização de endereços torna mais simples a alteração de alguns desses serviços. Por exemplo, caso algum serviço seja descontinuado, ou tenha seu endereço modificado, essa alteração não precisa ser feita em todos os clientes, sendo apenas necessária na API.

2.4. Páginas Web

Páginas Web são construídas a partir de três tecnologias fundamentais, HTML (ou XHTML), CSS e JavaScript, que são incorporadas por padrão nos principais navegadores existentes. A primeira é responsável por definir a estrutura da página. Por exemplo, onde iniciam e terminam textos, parágrafos, cabeçalhos, entre outros. A HTML também define *links* e locais que as imagens devem aparecer. O CSS, por sua vez, controla fontes, cores, tamanhos ou qualquer outro aspecto visual de informações presentes nas páginas HTML.

Já o JavaScript é uma linguagem de programação responsável por fornecer interatividade às páginas por meio de *scripts* [Duckett 2010]. Apesar de ser muito importante em fornecer funcionalidades dinâmicas nas páginas clientes, o JavaScript também pode ser executado no servidor. O Node.js é um exemplo de tecnologia que executa programas no JavaScript no lado do servidor. O Node.js utiliza vínculos de baixo nível para manipulação de arquivos, processos, fluxos, soquetes, dentre outros e utiliza o interpretador V8 do Google, que é uma máquina virtual que permite que isso seja possível [Flanagan 2011]. Uma forma de construir páginas mais dinâmicas é utilizar o conceito de Aplicativo de Página Única.

2.5. Aplicativo de Página Única

As páginas Web tradicionais geralmente eram construídas utilizando *frameworks* MVC que focavam em servir páginas estáticas, considerando que todos os clientes eram "burros" e isso causava mais lentidão na navegação. Cada clique em um novo *link* fazia com que fosse exibido uma tela branca de carregamento e todo o conteúdo, como a navegação, anúncios, textos, e qualquer outra informação da página, fosse carregado novamente. Isto causava uma experiência inaceitável para consumidores da Internet mais sofisticados [Mikowski e Powel 2013]. Para resolver esse problema, surgiu uma nova abordagem para construir páginas Web, as Aplicações de Página Única (*Single Page Application* - SPA). O propósito do SPA é entregar uma aplicação *desktop* para o navegador, no qual o usuário consegue efetuar a navegação sem que seja necessário o carregamento da página por completo.

2.6. Usabilidade e Design Responsivo

A área de conhecimento que estuda a relação dos utilizadores com as ferramentas disponíveis e o seu uso de forma simples e fácil é a usabilidade, sendo ela a principal premissa para o sucesso na Internet. Se um site é difícil de usar, as pessoas simplesmente saem [Mesquita 2013]. Existem diferentes métodos para avaliar a usabilidade, sendo uma delas a norma ISO 9241-11. Ela define diretrizes com o propósito de que usuários atinjam os seus objetivos com eficácia, eficiência e satisfação. Um conceito que aprimora a usabilidade em uma página web é a responsividade.

Segundo Jiang *et al. (*2014), o Design Responsivo da Web integra as habilidades necessárias para utilizar mecanismos que façam a interface ser exibida de uma forma adequada para qualquer dispositivo. Alguns exemplos destes mecanismos são os recursos de *Media Queries* e *Flow Layout* do CSS e bibliotecas como o Bootstrap. Uma página web responsiva, além de ter uma interface amigável (*user-friendly*), também reduz o custo de manutenção, já que não são necessárias diferentes aplicações e domínios para dispositivos de variados tamanhos de telas.

3. Trabalhos Correlatos

3.1. Sistema de Certificados - Versão Monolítica

O Sistema de Certificados é uma solução Web de gerenciamento de emissão e validação de certificados implantado e utilizado no IFBA *Campus* Vitória da Conquista desde 2017 e o seu acesso está disponível em http://certificados.conquista.ifba.edu.br [Ramos *et al.* 2018]. Esta versão foi desenvolvida com o padrão de arquitetura monolítica e com a linguagem de programação PHP que define o pré-processamento das páginas HTML, possibilitando assim a montagem da interface com a qual o usuário interagirá com o sistema. Foi utilizado o sistema gerenciador de banco de dados MySQL para criar, editar, armazenar e organizar os dados da aplicação. O Sistema de Certificados é *open source*, o que facilita a implantação e o aprimoramento por qualquer outra instituição que tenha a intenção de usá-lo. O código fonte está disponível em https://github.com/joabepinheiro/certificados.

Segundo Ramos *et al.* (2018), o principal objetivo do sistema de Certificados é reduzir a quantidade de esforço e insumos gastos nesses processos de gerenciamento e disponibilização e trazer uma maior segurança, fornecendo a possibilidade de consultar a legitimidade do certificado. Para isso, o sistema possui o gerenciamento de eventos e suas atividades. Tais atividades podem contar com participantes para os quais serão

produzidos certificados com códigos únicos para a validação, com o objetivo de comprovar a participação dos participantes do evento. Os usuários que acessam o sistema são divididos em três categorias, cada um com o seu perfil de privilégio de acesso. O primeiro é o administrador, que pode cadastrar eventos e coordenadores, e gerenciar configurações para o funcionamento da aplicação. O outro perfil é o de coordenador, que tem a possibilidade de adicionar atividades e participantes aos eventos que ele coordena e definir o leiaute e o texto do certificado que essas participações gerarão. O último perfil é o de participante, que pode acessar o sistema utilizando o CPF e a data de nascimento, e assim, baixar os certificados de todas as atividades das quais participou na instituição. Também é disponibilizado um acesso público, que permite que qualquer pessoa verifique a validade de um certificado por meio de seu código.

3.2. Outras Soluções

Houve também uma pesquisa sobre o funcionamento de soluções similares para disponibilização de certificados para participantes de eventos, com o objetivo de identificar a necessidade de construção de uma nova plataforma, e também indicar possíveis melhorias que possam ser desenvolvidas. Dentre as que foram avaliadas, estão Doity (2021), Even3 (2021), Sympla (2021) e Gerar Certificado (2021). Com essa análise foi constatado que as três primeiras são plataformas focadas em gerenciar inscrições de eventos, fornecendo ferramentas robustas de divulgação, gerenciamento de pagamento, e controle de participantes. Elas são muito úteis para eventos que ainda não foram realizados. Entretanto, caso a intenção seja apenas gerar certificados para um evento que já foi realizado, alguns usuários podem enfrentar dificuldades, por conta do foco desalinhado entre o propósito da plataforma e o que precisa ser feito.

O Gerar Certificado (2021) é uma opção mais especialista e torna a experiência mais simplificada nesse aspecto. Porém, não há muitas opções em relação a customização do leiaute do certificado, se comparado com o Even3 (2021) e Sympla (2021). Ademais, todas precisam que o usuário crie uma conta na plataforma ou utilize um e-mail para efetuar o *download* do certificado, diferente da proposta do Sistema de Certificados IFBA (2021), que apenas exige a inserção do CPF e da data de nascimento. Na Tabela 1 é apresentada de uma forma mais detalhada a comparação entre as soluções disponíveis em comparação com a proposta deste trabalho.

Tabela 1. Comparação detalhada entre as plataformas disponíveis e o Sistema de Certificados do IFBA. Fonte: Próprio autor.

Sistema/ Característica	Foco	Participantes	Leiaute do Certificado	Tipo de Certificado	*Download* de Certificado
Doity (2021)	Gerência, divulgação, inscrição e pagamento do evento	Cadastro via plataforma ou via site de divulgação	Pouca personalização e não permite inserir verso	Participantes, organizadores e palestrantes	E-mail e cadastro na plataforma
Even3 (2021)		Cadastro via plataforma, planilha ou site de divulgação	Total personalização e permite inserir verso	Possibilidade de gerar para grupos ou todos	
Sympla (2021)		Cadastro via site de divulgação	Total personalização e não permite inserir verso	Um certificado por evento	
Gerar Certificado (2021)	Gerar e validar certificados	Cadastro via plataforma ou planilha	Pouca personalização e permite inserir verso	Um para cada geração	
Certificados IFBA (2021)	Gerar, disponibilizar e validar certificados		Total personalização e permite inserir verso	Gera de acordo com função e atividade do evento	CPF e data de nascimento (não é necessário o cadastro)

4. Materiais e Métodos

A arquitetura escolhida para a construção da nova plataforma foi a de microsserviços, principalmente porque ela proporciona uma definição clara de domínio, facilitando assim a divisão do desenvolvimento entre diferentes equipes. Portanto, neste trabalho será detalhado o serviço Web, que é responsável pela interface que o usuário interage com o sistema, bem como a API Gateway, que consome outros microsserviços e fornece um endereço central para as requisições HTTP que são enviadas pelos diferentes clientes da aplicação, incluindo a interface Web. Entretanto, estes não são suficientes para que o sistema funcione. Os outros microsserviços, que têm o papel de gerenciar permissões, realizar leitura, escrita e armazenamento de dados e demais funções, foram desenvolvidos por outra equipe. Na Figura 2 está localizado o contexto deste trabalho perante toda a aplicação.

O Docker foi utilizado com o objetivo de facilitar o desenvolvimento, o empacotamento e a execução dos microsserviços. Essa plataforma aberta fornece a possibilidade de empacotar e rodar uma aplicação em um ambiente pouco acoplado, determinado como contêiner [Docker 2021]. Esses contêineres utilizam *templates*, como imagens, para criar o ambiente de execução de cada aplicação. Também foi utilizado uma ferramenta para facilitar a criação dos contêineres e imagens para os microsserviços, o *Docker Compose*. Assim, é possível determinar em um único arquivo uma coleção de serviços que podem ser executados em contêineres de forma simultânea. Os contêineres do projeto foram construídos inicialmente para serem executados em um único ambiente computacional, entretanto, por utilizar a arquitetura com a tecnologia *Docker*, é possível realocar esses microsserviços sem muitos esforços.

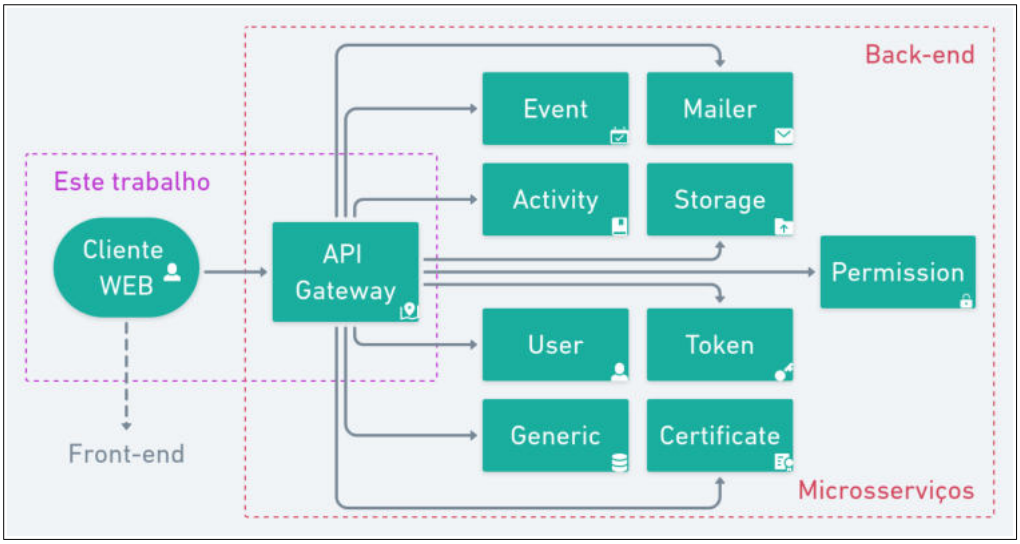

Figura 2. Disposição dos elementos da aplicação e o contexto deste trabalho. Fonte: Próprio autor.

4.1. Microsserviços Back-end

A API Gateway e os outros microsserviços consumidos por ele são categorizados pelo termo Back-end. Ao todo são 9 microsserviços (Figura 2): o *user* é responsável por gerenciar e armazenar usuários, enquanto que o *permission* fornece informações sobre o que eles têm de privilégio de acesso, e o *token* leva em consideração as permissões para gerar *tokens* no formato JWT para autorização nos demais microsserviços. Além destes, o *event*, *a activity* e *o certificate* armazenam e gerenciam os dados dos eventos, atividades e certificados, sendo o último também responsável por gerar o arquivo PDF do certificado de participação de acordo com o modelo que é representado por uma imagem gráfica no formato JPEG. O *storage* trata do armazenamento de arquivos (a exemplo de imagens dos modelos de certificados) e o *mailer* responde pelo envio de e-mails quando for necessário após a disponibilização do certificado. Existe ainda um microsserviço chamado *generic* que gerencia configurações que possuem apenas uma descrição e são usados como listas. Os tipos das atividades e as funções disponíveis para seleção estão armazenados nele. Apesar da arquitetura de microsserviços permitir o uso de múltiplos sistemas de gerenciamento de banco de dados, todos os microsserviços utilizaram a mesma instância do sistema de gerenciamento de banco de dados, o MongoDB. Na próxima subseção é explicado a motivação para a criação do microsserviço API Gateway e seus benefícios.

4.2. Microsserviço: API Gateway

Como a arquitetura escolhida para a construção do sistema foi a de microsserviços, o Cliente Web precisa se comunicar com todos eles para realizar as funções do sistema. Os 9 microsserviços criados poderiam estar localizados em diferentes endereços. Inicialmente não seria um grande problema configurar esses endereços no único cliente da aplicação. Porém, com o passar do tempo, onde novos clientes e microsserviços fossem desenvolvidos ou atualizados, isso poderia causar uma demanda de manutenção para gerenciá-los. Para resolver esse problema, foi criado o microsserviço API Gateway, que centraliza todas as requisições dos clientes e consome os outros microsserviços. Assim, caso ocorra alguma alteração ou atualização, o único local que precisaria ser modificado seria na API Gateway. Isso causa uma economia de esforço na manutenção

dos clientes. Para construir a API Gateway foi utilizado o NestJS (2021), que facilita a configuração e fornece ferramentas para criar serviços HTTP em TypeScript (2021), que são executados no Node.js. Ele utiliza um padrão de módulos para injetar as dependências do projeto baseado no framework Angular.

4.3. Serviço: Cliente Web

O serviço responsável por disponibilizar a interface que o usuário interagirá é denominado de Cliente Web. Ele foi projetado para ser um Aplicativo de Página Única, com o objetivo de aprimorar o desempenho da navegação e interatividade do sistema. Para isso, foi utilizado o *framework* Next.JS (2021), que tem como base a biblioteca React (2021) e a linguagem TypeScript (2021). Com o React é possível dividir a interface Web em componentes, provendo uma maior organização e facilitando a manutenção e reutilização de código. Outro benefício é o gerenciamento das informações dinâmicas da página por meio de estados. O TypeScript fornece uma tipagem para a linguagem JavaScript, o que auxilia na sinalização de erros no processo de desenvolvimento.

Com o objetivo de aproveitar os benefícios de trabalhar com componentes que o React traz para as páginas, o projeto utilizou os *Styled Components* para definir os estilos da interface. Essa biblioteca permite criar componentes HTML, como botões, entradas de texto e títulos, que são combinados ao CSS via JavaScript [Styled Components 2021]. Assim, é possível reutilizar esses mesmos componentes em diversos trechos do projeto, contribuindo para a reutilização e manutenção do código. A biblioteca também permite definir parâmetros para esses componentes, sendo muito útil para fazer com que algum aspecto visual da interface seja alterado de acordo com alguma condição ou algum estado do próprio React. Esse fator foi determinante para a criação do tema da aplicação, que reúne, em um único arquivo, informações de cores primárias, secundárias, de aviso, margens e algumas definições para a responsividade da aplicação. Todos os componentes de estilo consomem esses arquivos para definir essas informações. Desse modo, é possível modificar o tema de acordo com a instituição que a aplicação for implantada.

Os primeiros elementos da interface desenvolvidos foram os mais fundamentais para construir as páginas dinâmicas. De início, era necessário um componente de entrada de texto simples, contendo parâmetros para definir os tipos de dados e como eles seriam formatados. Por exemplo, o usuário pode inserir um texto simples, data, código de CPF e telefone. Posteriormente, todos esses dados são formatados de uma maneira diferente pela entrada de texto, e isso é determinado por um dos parâmetros desse componente. Também criou-se um componente de botão que recebe como parâmetro cor de definição e leiaute, sendo que para este último há três tipos disponíveis: a) botão com texto vazado sobre fundo de cor definida como parâmetro para maior destaque, objetivando induzir o usuário a realizar a ação; b) botão com borda de contorno, a ser usado em ações secundárias; e c) botão em formato de link que acaba tendo um destaque menor na interface. Outro componente criado foi o seletor de valores, que pode receber uma lista estática ou dinâmica a partir de uma requisição ao servidor. Para a entrada de arquivos, foi criado um seletor com a capacidade de funcionar como um *drag and drop* (“arrasta e solta”), e fornece também uma visualização prévia. Conforme as páginas foram sendo construídas, os componentes que já tinham sido desenvolvidos eram usados ou aprimorados. Em alguns casos, havia a necessidade da criação de novos componentes. Na Figura 3 é mostrado alguns destes componentes criados.

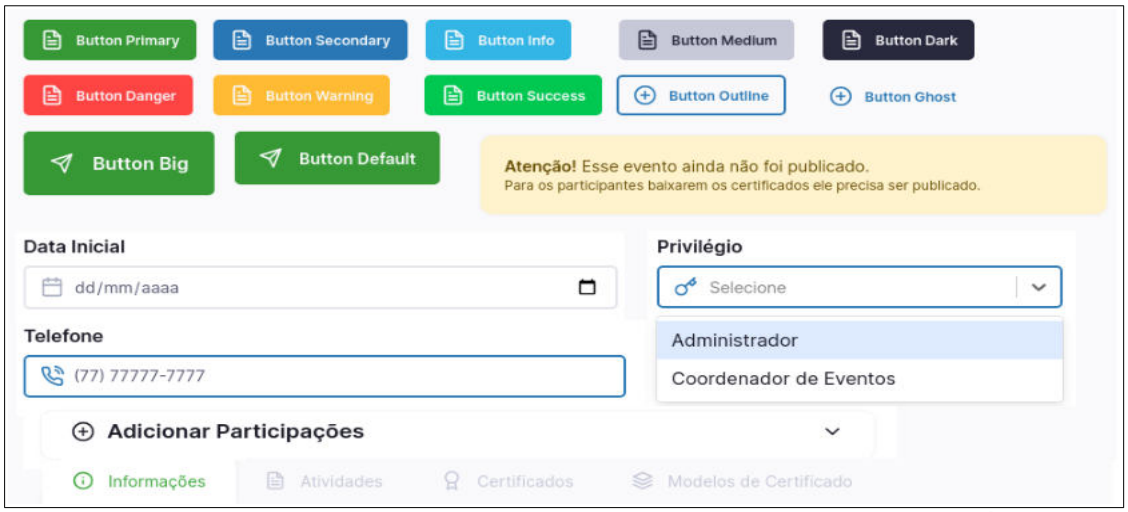

Figura 3. Exemplos de componentes criados no Cliente Web. Fonte: Próprio autor.

O uso desses componentes trouxe uma padronização estética na interface, entretanto, isto não foi o suficiente para melhoria da intuitividade. Para resolver isso, foram adotados alguns padrões de posicionamento dos elementos nas páginas, que estão exemplificados na Figura 4. Para navegação de funcionalidades primárias como eventos, participantes, usuários e configurações são inseridos atalhos em uma barra posicionada ao lado esquerdo (Área 1). Quando se acessa um desses atalhos primários, o usuário pode clicar em abas que ficam posicionadas sempre na parte superior da tela (Área 2). Outro padrão adotado é o posicionamento do botão de adição e campo de filtragem de registros estando sempre acima da tabela que contém os dados atuais (Área 3).

Alterações ou adições sempre são exibidas em janelas flutuantes (Figura 5), com o botão de confirmação com o leiaute padrão localizado à direita (Área 1) e o botão de cancelar contornado e posicionado à esquerda (Área 2). Segundo a UXMovement (2011), quando botões de confirmação são posicionados à direita de janelas, ocorrem menos fixações visuais, ações são mapeadas de uma melhor forma e um fluxo de uso é mais eficiente. Esses padrões são implementados com o objetivo do usuário realizar uma ação intuitivamente antes de fazer uma análise mais aprofundada na página. Deste modo, é possível encontrar a função desejada mais rapidamente, tornando a experiência mais produtiva.

Figura 4. Padrões de posicionamento da interface Evento. Fonte: Próprio autor.

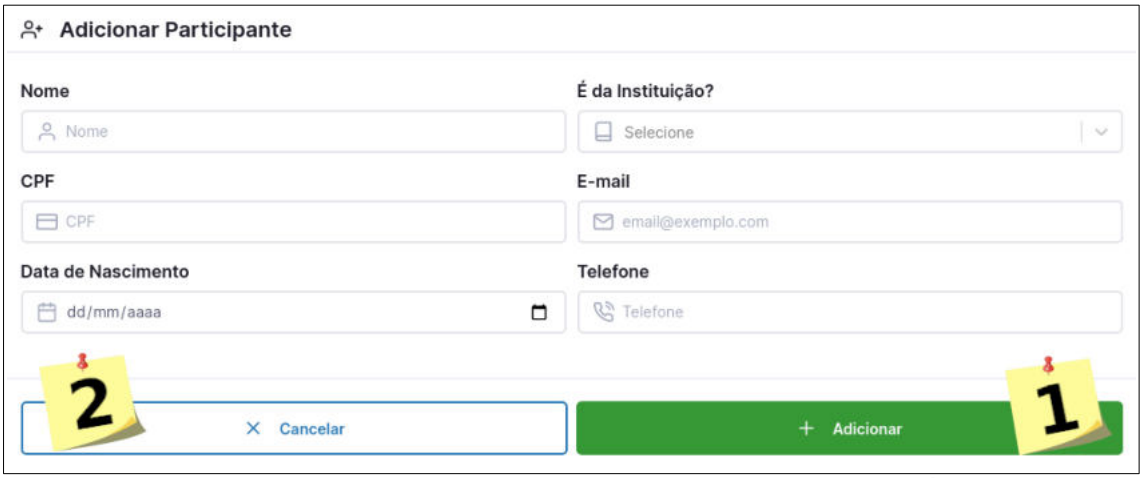

Figura 5. Padrão de janela de adição da interface Participante. Fonte: Próprio autor.

Por ser um sistema que é acessado por dispositivos de diferentes tamanhos de telas, como celulares, *tablets* e computadores, um dos principais requisitos é a responsividade da interface. Para implementar esse padrão, os componentes e as páginas criadas foram desenvolvidas utilizando o recurso de *Media Queries* do CSS3. Com esse recurso é possível escrever expressões condicionais para as folhas de estilo de acordo com largura e altura [Mozilla 2021]. Por exemplo, tamanhos de fontes e margens foram reduzidos em telas pequenas e botões que continham texto e ícone mantiveram apenas o ícone. Na Figura 6 é exibido um exemplo de uso desse recurso. No arquivo, que contém o estilo global da página, é definida uma regra que determina que a fonte deve ter 93,75% do tamanho normal para telas até 1024 pixels. Caso não seja esse o caso, a fonte deve ter o tamanho de 100%.

```
@media (max-width: 1024px) {
  html {
    font-size: 93.75%;
  }
}
```

Figura 6. *Media Query* para ajuste responsivo do tamanho da fonte da página. Fonte: Próprio autor.

O sistema foi dividido em três interfaces, a saber: Interface Administrativa, Interface do Participante e Interface de Validação. Para o acesso da primeira é necessário que o usuário digite o e-mail e a senha. Caso esses dados estejam corretos, a requisição responsável pelo login retorna como resposta um JSON Web Token (JWT) contendo as informações a respeito do usuário. Esse *token* é armazenado nos *cookies* do navegador e a partir disso, será enviado como cabeçalho de autorização, com o objetivo de liberar o acesso para as requisições administrativas. Essa configuração é definida facilmente com a criação de um interceptador na biblioteca Axios (2021), que é usada para efetuar as requisições HTTP entre a página e a API Gateway.

Os coordenadores gerenciam participantes, eventos, atividades e certificados, e administradores criam eventos e manipulam configurações para o funcionamento do sistema por meio da área administrativa. Para facilitar a inserção de alguns dados que são inseridos em grande volume, como participantes, atividades e certificados, foi criada a funcionalidade de importação por planilha (Figura 4, Área 3). Ela fornece opção de

download de uma planilha modelo, que deve ser preenchida pelo usuário, e depois inserida no sistema. A função faz a leitura das linhas da planilha e com auxílio da biblioteca XLSX [SheetJS 2021] as converte em objetos JSON com atributos idênticos aos do cadastro via interface, o que possibilita a utilização da mesma validação. Caso o registro esteja válido, uma requisição é enviada para o servidor para cada linha correta. Desta forma, é possível utilizar o mesmo método, tanto na interface quanto na importação auxiliando na reutilização e facilitando a manutenção de código.

Para o acesso da Interface do Participante, é preciso inserir o CPF e a data de nascimento, descartando assim a criação de uma conta pelo participante, que é um requisito importante do sistema. Essa requisição também retorna um JWT, no qual é armazenado em memória com o objetivo de manter o usuário autorizado apenas enquanto a página estiver aberta. Por essa interface, o usuário efetua o *download* dos certificados e visualiza eventos e atividades das quais participou. A última interface, que é a de Validação, é pública, ou seja, não necessita de uma autorização prévia. Por ela, qualquer usuário, com um código de validação, poderá verificar a validade de um certificado, visualizar suas informações, e baixá-lo se for preciso.

5. Resultados

Além de desenvolver todos os requisitos da aplicação anterior, como as interfaces do administrador, participante e validação, também foram incrementadas algumas novas funcionalidades. Uma delas é a etapa de publicação de evento, exibido na Figura 7, que consiste na revisão das informações, antes da disponibilização dos certificados aos participantes, com o objetivo de evitar possíveis erros. Um dos erros que acontecia na plataforma anterior era uma configuração de tipo de atividade com função sem um modelo de certificado definido (Figura 8). Na nova versão da plataforma, o evento apenas é disponibilizado caso todas as atividades e funções tenham um modelo cadastrado. Foi criado um componente chamado *stepper* para o desenvolvimento da tela de publicação da Figura 7, que é utilizado para representar a navegação em etapas sequenciais. Ele foi reutilizado no componente de importação de registros via planilha.

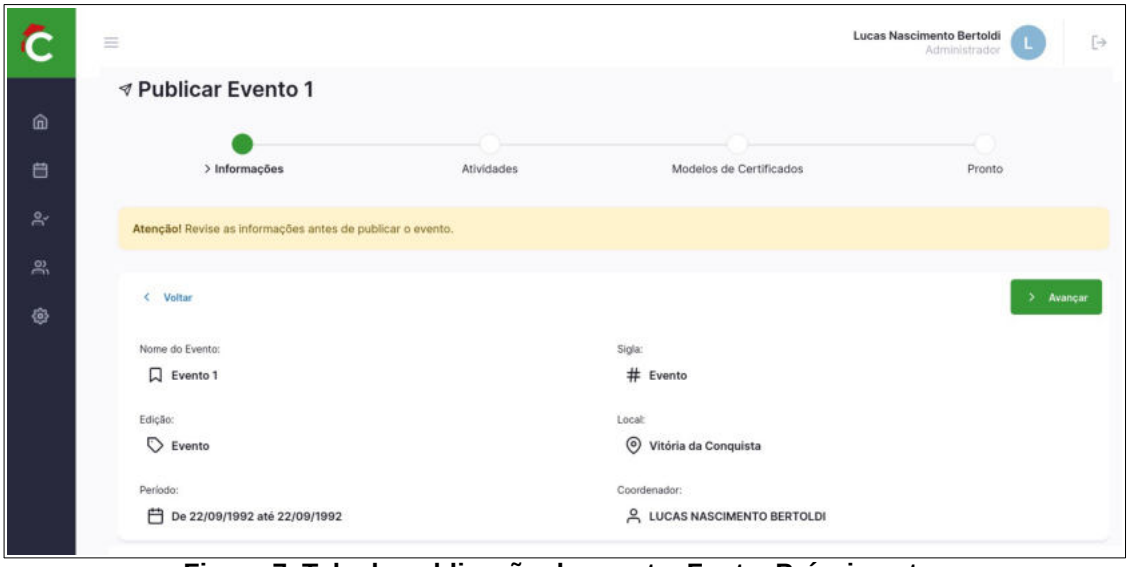

Figura 7. Tela de publicação de evento. Fonte: Próprio autor

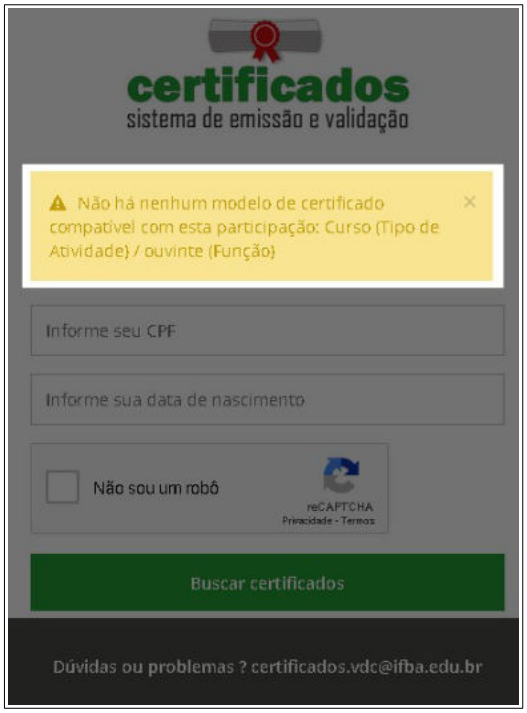

Figura 8. Erro da plataforma anterior ocasionado pela ausência de um modelo cadastrado. Fonte: Ramos *et al.* (2018)

Outra melhoria foi a possibilidade de customizar o modelo do certificado (Figura 9), sendo possível definir tamanho e formatação do texto (Área 1), adição de novas tags (Área 2) e edição de margens (Área 3), além da visualização em tempo real das mudanças (Área 4).

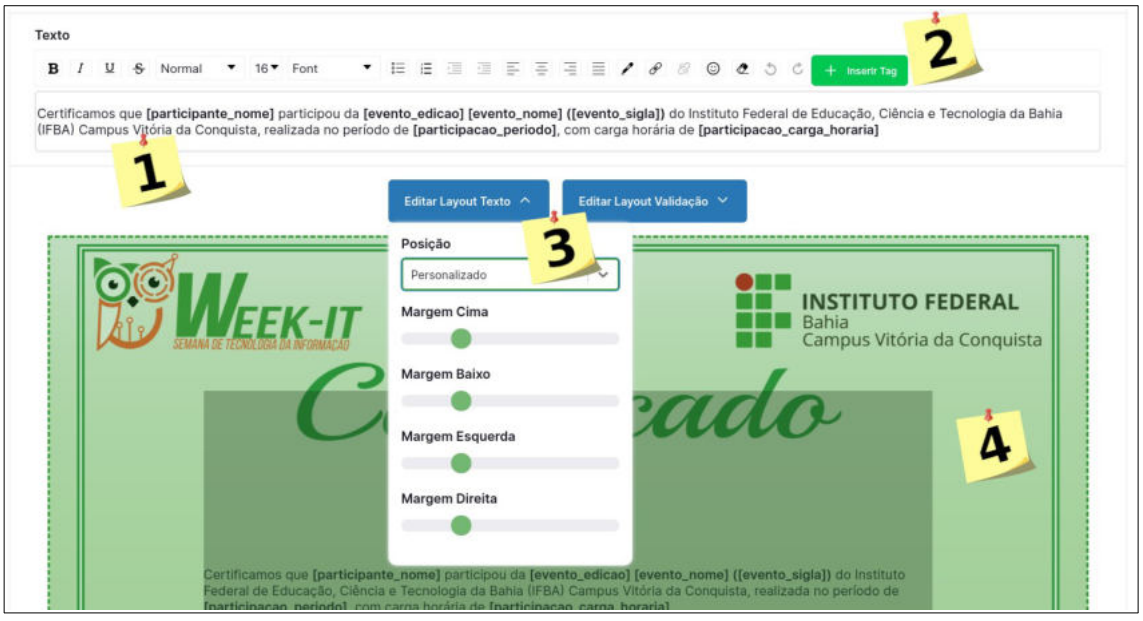

Figura 9. Tela de edição do modelo do certificado. Fonte: Próprio autor

Também houve uma melhoria na fluidez na navegação, evidenciado por usuários da plataforma anterior. Isto foi possível, por conta da nova página utilizar o conceito de Aplicativo de Página Única. A nova interface desenvolvida propôs um *design* mais padronizado e moderno que a versão anterior da aplicação. Isso proporcionou uma maior identidade visual em todas as páginas. Nas Figuras 10 e 11 é possível observar a

tela de disponibilização de certificados, respectivamente, na versão anterior e na atual desenvolvida neste trabalho. Além disso, espera-se uma maior facilidade dos usuários no primeiro acesso à plataforma, por conta da padronização da posição dos elementos.

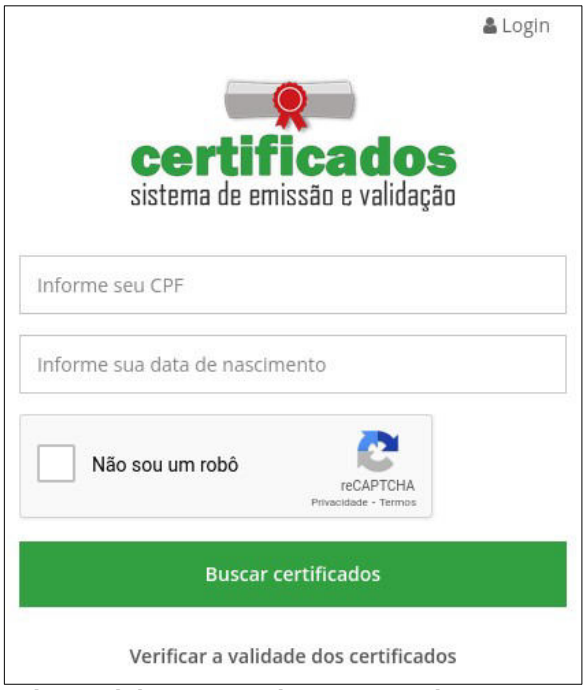

Figura 10. Interface do participante no sistema anterior. Fonte: Ramos *et al.* (2018)

Figura 11. Interface do participante no novo sistema. Fonte: Próprio autor

A nova versão do Sistema de Certificados também é *open source* e o código fonte está disponível em https://github.com/Certificados-Ifba/certificates. O sistema está em processo de implantação no IFBA *Campus* Vitória da Conquista no domínio: https://certificados.vca.ifba.edu.br.

6. Considerações Finais

O desenvolvimento da nova versão do sistema de gerenciamento de emissão e validação de certificados teve como principal objetivo aprimorar a usabilidade de participantes e coordenadores de eventos. Para isso, foi construída uma interface Web utilizando o conceito de apenas uma página, o que tornou a navegação muito mais ágil. Outro fator determinante para este aprimoramento foi o *design* responsivo, que possibilitou uma experiência agradável em qualquer dispositivo, independente do tamanho da tela. A nova versão incluiu também melhorias identificadas por usuários do sistema anterior.

A nova arquitetura de microsserviços auxiliará a manutenção e a escalabilidade do projeto. Como consequência, é esperado que seja possível construir novos microsserviços e aprimorar outros, sendo que o desenvolvimento pode ser feito de uma forma mais simples por diferentes equipes quando comparado com o desenvolvimento na arquitetura monolítica, conforme evidenciado neste artigo. O desenvolvimento a partir de componentes se provou muito útil para a melhoria constante do projeto, onde novos requisitos ou melhorias eram refletidos em toda a interface. Os *frameworks* NestJS (2021) e Next.JS (2021) tornaram o código fonte organizado e padronizado, além de fornecerem ferramentas que facilitaram a configuração de alguns aspectos da aplicação. Outro benefício foi a ferramenta de reflexão de alterações em tempo real, que possibilitou construir a aplicação a partir de pequenas alterações, o que torna o processo de desenvolvimento muito mais produtivo.

Existem diversas melhorias que podem ser realizadas como trabalhos futuros. Por exemplo, o serviço de Cliente Web pode ser dividido em três microsserviços, um para cada interface principal abordada, a saber: Administrativo, Participante e Validação. Também é possível desenvolver a disponibilidade de alterar a interface para um tema escuro. O microsserviço Gateway API facilita o desenvolvimento de um aplicativo para dispositivos móveis, que teria como funcionalidade principal o controle de participação. O coordenador poderia gerar um QR-Code pela área administrativa e a confirmação do participante ser feita pelo escaneamento no local da atividade.

Referências

Axios (2021). “Axios”. Disponível em: <https://github.com/axios/axios>. Acesso em: 05 dez. 2021.

Docker (2021). “Docker Docs”. Disponível em: <https://docs.docker.com/>. Acesso em: 05 dez. 2021.

Doity (2021). “Doity - Organizar eventos presenciais ou online ficou fácil”. Disponível em: <https://doity.com.br/>. Acesso em: 05 dez. 2021.

Dragoni, N., Giallorenzo, S., Lafuente, A. L., Mazzara, M., Montesi, F., Mustafin, R. e Safina, L. (2017) "Microservices: yesterday, today, and tomorrow.". Present and ulterior software engineering, 195-216.

Duckett, J. (2010) “Beginning HTML, XHTML, CSS, and JavaScript”. Indianapolis: Wiley Publishing, Inc., p. 21.

Even3 (2021). “Even3 - Organizar um evento online nunca foi tão simples”. Disponível em: <https://www.even3.com.br/>. Acesso em: 05 dez. 2021.

Flanagan, D. (2011) “JavaScript: The Definitive Guide”. São Paulo: Bookman Companhia Editora Ltda., p. 281.

Gerar Certificado (2021). “Gerar Certificado - Gerador de certificado gratuito”. Disponível em: <https://gerarcertificado.com.br/gerador-certificado-gratuito.php>. Acesso em: 05 dez. 2021.

Jiang, W., Zhang, M., Zhou, B., Jiang, Y. e Zhang, Y. (2014) "Responsive web design mode and application". In: IEEE Workshop on Advanced Research and Technology in Industry Applications (WARTIA), p. 1303-1306.

Lewis, J. (2014) “Microservices - a definition of this new architectural term”, Disponível em: <https://martinfowler.com/articles/microservices.html>. Acesso em: 05 dez. 2021.

Mesquita, C. (2013) “Usabilidade na Web - Metodologias para a Avaliação Qualitativa da Usabilidade em dispositivos Mobile no sítio Web da Universidade do Porto”.

Mikowski, M. e Powell, J. (2013) “Single page web applications: JavaScript end-to-end”. Shelter Island: Manning Publications Co., p. 2-3.

Mozilla (2021) “Usando Media Queries”. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/CSS/Media_Queries/Using_media_queries>. Acesso em: 05 dez. 2021.

Negri, P. (2021) “API gateway: o que é e qual a sua função?”. Disponível em: <https://www.iugu.com/blog/api-gateway>. Acesso em: 05 dez. 2021.

NestJS (2021). “NestJS - A progressive Node.js framework”. Disponível em: <https://nestjs.com/>. Acesso em: 05 dez. 2021.

Next.JS (2021). “Next.js by Vercel - The React Framework”. Disponível em: <https://nextjs.org/>. Acesso em: 05 dez. 2021.

Penna, W. (2021) “Arquitetura Monolítica e Microsserviços”, Disponível em: <https://www.zappts.com/blog/arquitetura-monolitica-e-microsservicos/>. Acesso em: 05 dez. 2021.

Ramos, L. L., Silva, J. P., Sobreira, A. D. e Matos, P. F. (2018) “Sistema Web e Open Source de Gerenciamento de Emissão e Validação de Certificado nos Institutos Federais”. In: XII Congresso Norte Nordeste de Pesquisa e Inovação, Recife, PE, p. 1-10.

React (2021). “React – Uma biblioteca JavaScript para criar interfaces de usuário”. Disponível em: <https://pt-br.reactjs.org/>. Acesso em: 05 dez. 2021.

SheetJS (2021). “SheetJS Community Edition - Spreadsheet Data Toolkit”. Disponível em: <https://github.com/SheetJS/sheetjs>. Acesso em: 05 dez. 2021.

Styled Components (2021). “Styled Components - Visual primitives for the component age”. Disponível em: <https://styled-components.com/>. Acesso em: 05 dez. 2021.

Sympla (2021). “Sympla - A plataforma de eventos, cursos e transmissão online”. Disponível em: <https://www.sympla.com.br/>. Acesso em: 05 dez. 2021.

TypeScript (2021). “TypeScript is JavaScript with syntax for types”. Disponível em: <https://www.typescriptlang.org/>. Acesso em: 05 dez. 2021.

UXMovement (2011) “Why ‘Ok’ Buttons in Dialog Boxes Work Best on the Right”. Disponível em: <https://uxmovement.com/buttons/why-ok-buttons-in-dialog-boxes-work-best-on-the-right/>. Acesso em: 05 dez. 2021.