Sistema de Gerenciamento de Certificados

Matheus C. Andrade
matheusbdo1@gmail.com
Instituto Federal da Bahia
Vitória da Conquista, Bahia, Brasil

Lucas N. Bertoldi
lucasn.bertoldi@gmail.com
Instituto Federal da Bahia
Vitória da Conquista, Bahia, Brasil

Alexandro S. Silva
alexandrossilva@ifba.edu.br
Instituto Federal da Bahia
Vitória da Conquista, Bahia, Brasil

Pablo F. Matos
pablofmatos@ifba.edu.br
Instituto Federal da Bahia
Vitória da Conquista, Bahia, Brasil

Resumo

In view of the variety of events that take place today, there was a need for online integration between participants and the organizing committee. For this, websites are used that gather user data, information about events and other functionalities, one of which is the generation of certificates. The Federal Institute of Bahia Campus of Vitória da Conquista has a certificate management system, developed in an architecture that is currently obsolete due to limitations in terms of scalability, performance and difficulty of updating. This paper presents a new approach for issuing and validating certificates, detailing how it was developed. We used the microservices architecture, which has advantages over the old system, in addition to the replacement of the relational database by the non-relational database. It is also proposed an approach to the infrastructure to be used by the project.

Keywords: management certificates, events, microservices

1 Introdução

Em um mundo onde se tem uma variedade de eventos, sejam eles educacionais ou empresariais, sejam públicos ou privados, existe a necessidade de integração entre os participantes e a organização. Com as tecnologias atuais, existem plataformas online que fazem essa intermediação, gerenciando dados dos usuários e informações do evento. Algumas dessas plataformas, por exemplo, Doity[1], Even3[2] e Sympla[3], também oferecem a emissão de certificados devido à necessidade de documentação comprobatória, principalmente os eventos que ocorrem no meio acadêmico.

Para atender a essa necessidade, foi implantado um sistema online de emissão e gerenciamento de certificados no Instituto Federal da Bahia (IFBA) *Campus* de Vitória da Conquista. Ele foi desenvolvido em arquitetura monolítica, que apresenta algumas desvantagens frente a outras opções existentes atualmente, além de ter sido implementada em um *framework* que foi descontinuado [5]. Verificou-se, então, a necessidade de atualização dessa plataforma para melhoria da usabilidade do sistema e para oferecer um serviço com resposta rápida e otimizada às requisições por parte dos usuários e organizadores [6].

A arquitetura em monólitos não oferece tais vantagens, pois a aplicação em bloco único faz com que a sua atualização seja complicada, além de desperdiçar recursos de processamento, uma vez que de acordo com o escalonamento, há um aumento do uso de recursos computacionais. Outros modelos de arquitetura surgiram e tais problemas podem ser evitados com seu uso [8].

Também foi feito a substituição do banco de dados relacional por um banco de dados não relacional. Identificou-se que o projeto possui a possibilidade de escalar seus recursos para que sejam utilizados em outras instituições ou em eventos avulsos, gerando assim uma grande base de dados. Tais características apresentam problemas no esquema relacional, pois ele enfrenta dificuldade em processar grandes bases de informações gerando uma maior lentidão à medida que a aplicação escale horizontalmente. A abordagem não relacional tem como propósito solucionar essas demandas [4].

Nesse sentido, este trabalho apresenta o Sistema de Gerenciamento de Certificados, baseado na arquitetura de microsserviços e com a utilização de banco de dados não relacional, explicando suas metas, licença, principais funcionalidades e arquitetura.

2 Metas da Aplicação e Licença

Este projeto tem como objetivo o desenvolvimento de uma plataforma que permita o gerenciamento e emissão de certificados. Ele é uma nova versão de um sistema desenvolvido e utilizado no IFBA *Campus* Vitória da Conquista. Essa antiga versão tem como abordagem uma arquitetura monolítica utilizando a linguagem de programação PHP[4] com o Zend

[1]Doity: https://doity.com.br
[2]Even3: https://www.even3.com.br
[3]Sympla: https://www.sympla.com.br
[4]PHP: https://www.php.net

In: XXI Workshop de Ferramentas e Aplicações (WFA 2022), Curitiba, Brasil. Anais Estendidos do Simpósio Brasileiro de Sistemas Multimídia e Web (WebMedia). Porto Alegre: Sociedade Brasileira de Computação, 2022.

ISSN 2596-1683

Framework[5] e o banco de dados relacional MySQL[6]. A meta é desenvolver uma nova versão desse sistema migrando para uma arquitetura de microsserviços e utilizando um banco de dados não relacional, uma vez que o sistema anterior foi implementado em 2017 e desde então não teve nenhuma melhoria ou atualização, sofrendo com problemas de performance e com vulnerabilidades [7]. Além disso, construir uma nova interface Web que aprimore a usabilidade de participantes e coordenadores de eventos, que implemente os conceitos de UX/UI, entregando uma interface responsiva e amigável que pode ser acessada por todos os dispositivos atuais [2][3].

O sistema tem como função fornecer a criação, edição e gerência da emissão e validação de certificados dos mais variados eventos de maneira rápida e prática, reduzindo o esforço e insumos na organização dos eventos da instituição [7].

A plataforma tem o código aberto e é desenvolvida em parceria com as instituições públicas de ensino.

3 Arquitetura

Nesta seção são apresentados os aspectos da arquitetura utilizada para o Sistema de Gerenciamento de Certificados (Figura 1). Em seguida são descritos cada componente que a integra, além de apresentar as tecnologias utilizadas.

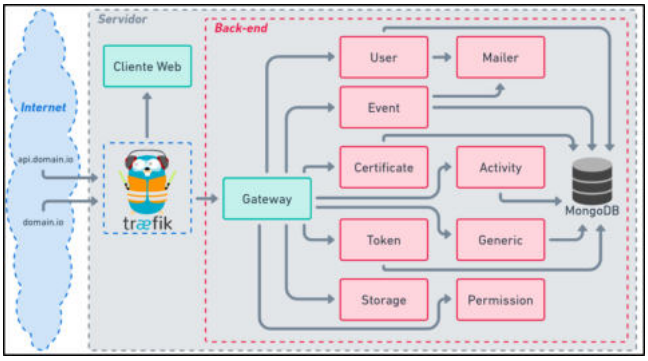

Figura 1. Arquitetura do sistema

- **Treafik:** o traefik é responsável pelo *Proxy* Reverso da aplicação, recebendo as solicitações do cliente web e da API, e as encaminhando para os serviços responsáveis pelo seu processamento. Foi escolhido pela sua fácil configuração (realizada através de um arquivo YAML[7]) e para ajudar a aumentar a segurança, desempenho e confiabilidade do sistema;
- **Cliente Web:** serviço responsável por servir o *front-end* do sistema;
- ***Gateway*:** serviço que atua como *gateway* do *back-end*, sendo responsável por centralizar todos os serviços de forma a atuar como um *Back-end for Front-end* (BFF), garantindo um melhor desempenho e segurança;
- ***User*:** serviço responsável pela autenticação, listagem, exibição, criação, alteração e remoção do usuário, bem como pela confirmação, reenvio e recuperação da senha;
- ***Event*:** serviço responsável pela listagem, exibição, busca, alteração, criação e remoção do evento;
- ***Certificate*:** serviço responsável pela listagem, emissão, exibição, validação e remoção do certificado, bem como pela listagem, exibição, alteração, criação e remoção do modelo de certificado;
- ***Activity*:** serviço responsável pela listagem, exibição, alteração, criação e remoção da atividade;
- ***Generic*:** serviço responsável pela listagem, exibição, alteração, criação e remoção de informações genéricas do sistema, podendo ser um tipo de atividade ou função do participante;
- ***Storage*:** serviço responsável pelo *upload*, exibição e remoção das imagens utilizadas no modelo do certificado;
- ***Token*:** serviço responsável pela geração, remoção e validação dos *tokens* JWT[8], sendo utilizados na autenticação das requisições realizadas pelo usuário, garantindo uma maior segurança;
- ***Permission*:** serviço responsável pela validação das permissões de acesso de cada rota do *gateway*;
- ***Mailer*:** serviço responsável pelo envio de e-mail do sistema, podendo ser de confirmação e recuperação da senha, além de informar a disponibilização do certificado ao participante;
- **MongoDB:** banco de dados não relacional orientado a documentos utilizado para persistência de dados.

A arquitetura de microsserviços foi utilizada pela possibilidade de entrega e implantação contínua de aplicativos grandes e complexos, da manutenibilidade aprimorada, da melhor testabilidade e capacidade de implantação e da viabilidade de divisão da equipe em times menores responsáveis por um ou mais serviços. Além disso, elimina o compromisso de longo prazo com uma determinada linguagem e/ou tecnologia, facilitando a utilização e alteração de tecnologias durante o seu desenvolvimento [8].

O sistema tem três tipos de usuários com permissões e características diferentes: Administrador, responsável por gerenciar os Coordenadores, os tipos de eventos e as funções, além de ter total acesso ao sistema; Coordenador, responsável por gerenciar os participantes, eventos, atividades, modelos de certificados e participações; Participante, acessa os certificados disponíveis, podendo realizar seu *download*.

[5]Zend Framework: https://framework.zend.com
[6]MySQL: https://www.mysql.com
[7]YAML: https://yaml.org
[8]JTW: https://jwt.io

Na Figura 2 há uma comparação entre a tela de *Dashboard* do sistema antigo e do sistema novo. A nova abordagem apresenta uma tela mais limpa ao aplicar conceitos de *UI/UX Design*, como navegabilidade, IHC, usabilidade, acessibilidade, responsividade, entre outros.

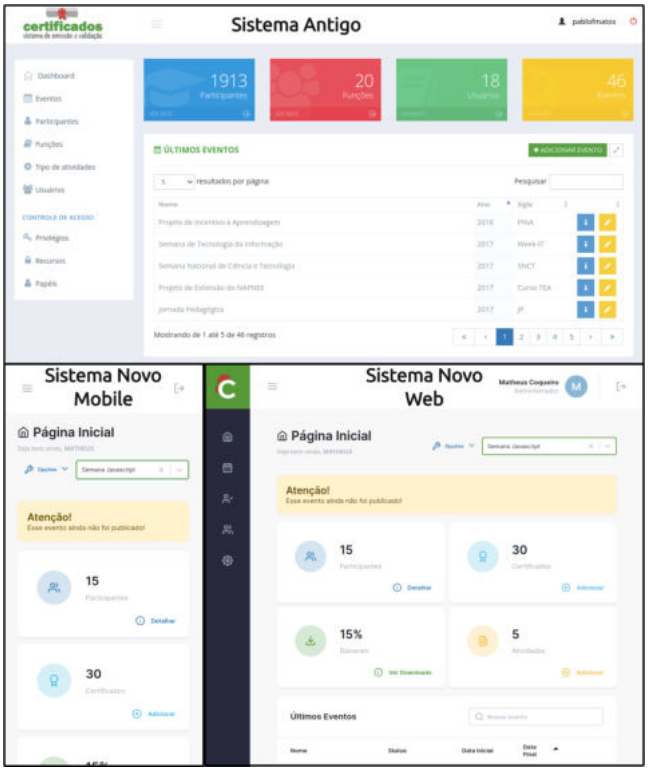

Figura 2. Tela de *Dashboard* do sistema antigo e como ela ficou no novo sistema

Os serviços do *back-end* foram desenvolvidos utilizando o Nest.JS[9], um *framework* Node.JS cuja estrutura se inspira no *framework* Angular[10] e que utiliza um padrão de módulos para injeção de dependências. O Nest.JS fornece muitas ferramentas e integrações com as mais diversas tecnologias da atualidade, sendo muito utilizado na criação de microsserviços, pois possui fácil configuração. O *framework* suporta vários meios de comunicação entre os serviços, como Redis, MQTT, NATS, RabbitMQ, Kafka e gRPC. Por se tratar de um sistema que ainda não possui tanta demanda, foi optado por implementar a comunicação via protocolo TCP, podendo ser substituído facilmente no futuro por algum serviço de mensageria caso tenha necessidade. Já o *front-end* foi desenvolvido utilizando a linguagem Typescript[11] com o *framework* Next.JS[12], que utiliza como base a biblioteca React[13].

[9]Nest.JS: https://nestjs.com
[10]Angular: https://angular.io
[11]Typescript: https://www.typescriptlang.org
[12]Next.JS: https://nextjs.org
[13]React: https://pt-br.reactjs.org

O banco de dados foi reestruturado, substituindo-se o modelo relacional pelo não relacional. Tomou-se essa decisão para otimizar o desempenho da aplicação e por conta da flexibilidade nos esquemas do banco (9 tabelas no esquema original foram reduzidas para 6 coleções). Também houve uma redução no tamanho, uma vez em que muitos campos no esquema antigo eram setados como *NULL* por não serem preenchidos. Outro motivo foi a fácil integração com a arquitetura adotada, já que os documentos do banco são muito semelhantes ao JSON[14]. Existem várias soluções que fornecem um banco de dados não relacional, porém foi optado pela utilização do MongoDB por ser o mais popular[1] e ter um ótimo suporte.

4 Funcionalidades

Nesta seção são apresentadas algumas telas do sistema e descrito suas principais funcionalidades.

A plataforma oferece a opção de importação via planilha de algumas informações, como participantes, atividades e participações. Essa função é muito importante, uma vez que é comum o uso de planilhas para coletar esses tipos de informações, assim facilita os cadastros dos mesmos. A importação é feita em 3 etapas, onde a primeira é disponibilizado o *template* com explicação de cada campo, depois é feito o envio da planilha e, por fim, é feito o processamento e exibido o resultado da importação (Figura 3).

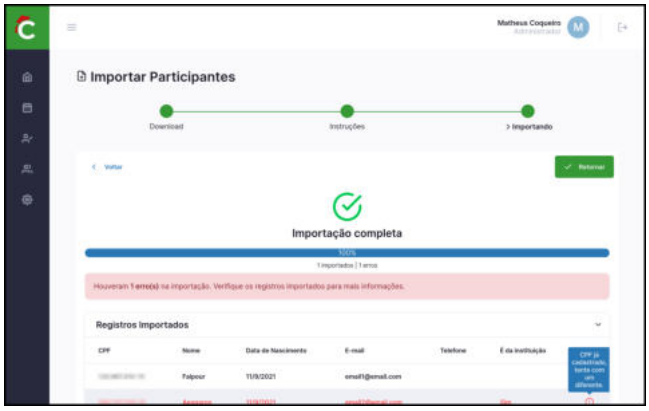

Figura 3. Tela de importação de participantes via planilha

Os eventos possuem os seguintes status: Rascunho, Publicado e Revisão. Rascunho é o status inicial após sua criação, quando o coordenador ainda está preenchendo as atividades, os modelos de certificados e as participações; Publicado é quando as informações do evento já foram preenchidas e os certificados foram disponibilizados aos participantes. O evento passa por uma série de validações até ser publicado e, uma vez feito, nenhuma informação poderá ser alterada; Revisão, caso o coordenador identifique alguma informação errada e queira alterar alguma informação, o evento precisa

[14]JSON: https://www.json.org

entrar em revisão. Durante essa etapa, os certificados ficarão indisponíveis e só voltarão após ser publicado novamente (Figura 4).

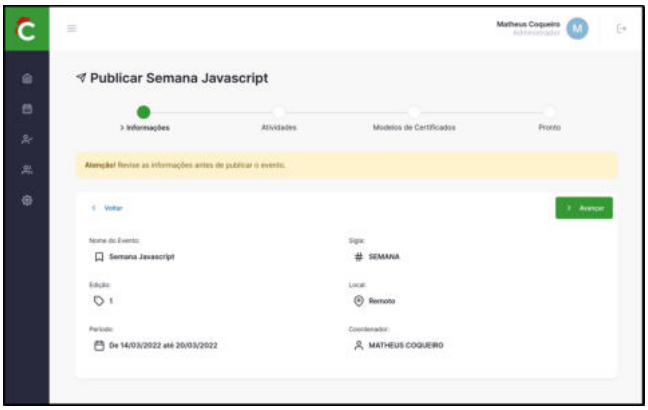

Figura 4. Tela de publicação de um evento

Outra funcionalidade essencial é a de poder verificar se um determinado certificado é válido e se suas informações não foram adulteradas. Para isso, é necessário informar o código de validação presente no certificado (Figura 5).

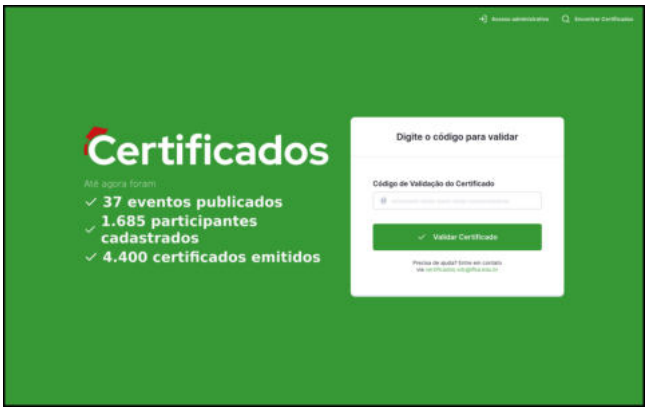

Figura 5. Tela de validação do certificado

Na Figura 6 é demonstrada a tela de edição do modelo de certificado, onde é possível definir o tamanho e formatação do texto (Área 1), adição de novas *tags* (Área 2), edição de margens (Área 3), além da visualização em tempo real das mudanças (Área 4). O Coordenador pode criar vários modelos para o mesmo evento, escolhendo os critérios em que cada um será utilizado, como também pode criar um modelo padrão.

O Participante pode consultar os eventos que participou e os certificados de participação, podendo visualizar e baixar os certificados. Para isso, deve informar somente o CPF e a data de nascimento.

Toda a codificação do sistema abordado neste trabalho está disponível gratuitamente para ser acessado através do

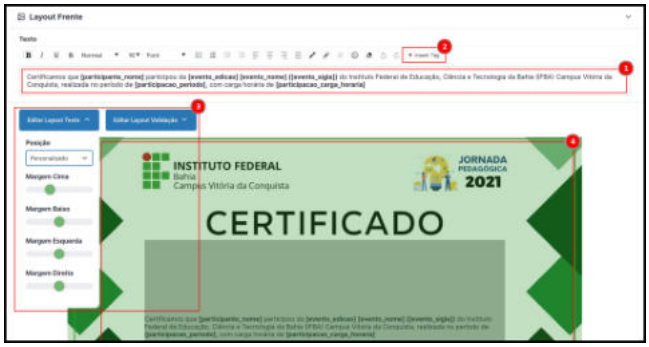

Figura 6. Tela de edição do modelo de certificado

endereço https://github.com/Certificados-Ifba/certificates. Já a codificação da infraestrutura utilizada pode ser acessada através do endereço https://github.com/Certificados-Ifba/infra. As funcionalidades do sistema podem ser visualizadas através do endereço https://www.youtube.com/watch?v=JxBKOgrv30o.

5 Considerações Finais

Este trabalho apresenta uma nova abordagem para o sistema de certificados que o IFBA *Campus* Vitória da Conquista utiliza atualmente. Esta abordagem é baseada na arquitetura de microsserviços, na substituição do banco de dados relacional pelo não relacional e na infraestrutura utilizada para a sua implementação. Essa nova estrutura foi pensada a partir da necessidade de atualizar o sistema atualmente utilizado e descreve as metas da aplicação, a licença, a arquitetura, as tecnologias utilizadas e as principais funcionalidades.

Referências

[1] Solid IT. 2022. DB-Engines Ranking of Document Stores. Disponível em: https://db-engines.com/en/ranking/document+store. Acesso em 5 setembro 2022.

[2] Wei Jiang, Meng Zhang, Bin Zhou, Yujian Jiang, and Yingwei Zhang. 2014. Responsive web design mode and application. In *2014 IEEE Workshop on Advanced Research and Technology in Industry Applications (WARTIA)*. IEEE, 1303–1306.

[3] Heonsik Joo. 2017. A study on understanding of UI and UX, and understanding of design according to user interface change. *International Journal of Applied Engineering Research* 12, 20 (2017), 9931–9935.

[4] Mwai Karimi. 2019. *Benchmarking of RDBMS and NoSQL performance on unstructured data.* Ph. D. Dissertation.

[5] James Lewis and Martin Fowler. 2014. Microservices. Disponível em: https://martinfowler.com/articles/microservices.html. Acesso em 21 julho 2022.

[6] Catarina Alexandra Sousa Mesquita. 2013. Usabilidade na Web-Metodologias para a Avaliação Qualitativa da Usabilidade em dispositivos Mobile no sítio Web da Universidade do Porto. (2013).

[7] Leandro Lopes Ramos, Joabe Pinheiro da Silva, André Daniel Sobreira, and Pablo Freire Matos. 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.

[8] Chris Richardson. 2014. Pattern: Microservice Architecture. Disponível em: https://microservices.io/patterns/microservices.html. Acesso em: 27 julho 2022.