Associação Propagadora Esdeva
Centro Universitário Academia – UniAcademia
Curso de Engenharia de Software
Trabalho de Conclusão de Curso – Artigo

Desenvolvimento de um Sistema para a Comissão Própria de Avaliação: Uma Abordagem FullStack Utilizando C# e React.js

Jhonathan Meireles de Oliveira[1]
Centro Universitário Academia, Juiz de Fora, MG

Tassio Ferenzini Martins Sirqueira[2]
Centro Universitário Academia, Juiz de Fora, MG

Marco Antônio Pereira Araújo[3]
Centro Universitário Academia, Juiz de Fora, MG

Linha de pesquisa: Engenharia de Software

Resumo

Este trabalho se concentra no desenvolvimento de um sistema para a Comissão Própria de Avaliação (CPA), com a meta expressa de melhorar a implementação e gestão das avaliações institucionais, considerando sua relevância na atualidade. O sistema foi arquitetado e construído utilizando tecnologias contemporâneas e robustas - C# juntamente com o framework .NET para o back-end, React para o front-end, e PostgreSQL para o armazenamento de dados - justificando-se pela necessidade de eficácia na avaliação institucional. O sistema engloba uma área administrativa intuitiva, permitindo aos gestores de sistema cadastrar avaliações de maneira fácil e eficiente. Um painel de controle visual fornece uma exibição dos resultados das avaliações respondidas, apresentando uma variedade de dados e gráficos, além de relatórios, de maneira a garantir a confidencialidade das respostas. O acesso ao sistema é facilitado através do e-mail institucional e uma senha padrão, necessitando apenas a confirmação da matrícula durante o primeiro

[1] Discente do Curso de Engenharia de Software do Centro Universitário Academia - UniAcademia. E-mail: jhon4than1995@gmail.com
[2] Docente do Curso de Engenhara de Software do Centro Universitário Academia - UniAcademia. Orientador.
[3] Docente do Curso de Engenhara de Software do Centro Universitário Academia - UniAcademia. Orientador.

acesso para garantir segurança. Se uma avaliação estiver disponível para o semestre em curso, os usuários têm a opção de respondê-la.

Palavras-chave: Comissão Própria de Avaliação, avaliação institucional, desenvolvimento de software, C#, React.js, PostgreSQL.

Abstract:

This work focuses on the development of a system for the Comissão Própria de Avaliação (CPA), with the express goal of improving the implementation and management of institutional evaluations, considering their relevance today. The system was architected and built using contemporary and robust technologies - C# along with the .NET framework for the back-end, React for the front-end, and PostgreSQL for data storage - justified by the need for effectiveness in institutional assessment. The system encompasses an intuitive administrative area, allowing system managers to register evaluations easily and efficiently. A visual dashboard provides a display of the results of completed evaluations, presenting a variety of data and graphs, as well as reports, to ensure the confidentiality of responses. Access to the system is facilitated through institutional email and a default password, requiring only confirmation of enrollment during first access to ensure security. If an evaluation is available for the current semester, users have the option to take it.

INTRODUÇÃO

A avaliação institucional é uma prática essencial para o aprimoramento contínuo das instituições de ensino. Com o objetivo de tornar esse processo mais eficiente e gerenciável, este trabalho apresenta o desenvolvimento de um sistema para a Comissão Própria de Avaliação (CPA) onde os usuários discentes, docentes e coordenadores de curso terão acesso ao sistema e poderão responder as avaliações destinadas aos usuários. Através da utilização da programação C#[4] e do *framework* .NET[5], foi criada uma API[6] em back-end[7], enquanto o front-end[8] foi desenvolvido em React.JS[9] usando bibliotecas para agilizar desenvolvimento como MUI[10] que possui uma gama de componentes que auxiliaram no desenvolvimento desse projeto com qualidade e agilidade, proporcionando ao usuário uma

[4] C#. Disponível em: <https://learn.microsoft.com/pt-br/dotnet/csharp/tour-of-csharp/>. Acesso em: 02 junho 2023.
[5] .NET. Disponível em: <https://dotnet.microsoft.com/pt-br/>. Acesso em: 02 junho 2023.
[6] API. Disponível em: <https://aws.amazon.com/pt/what-is/api/>. Acesso em: 02 de junho 2023.
[7] Back-end. Disponível em: <https://harve.com.br/blog/desenvolvimento-web/o-que-e-backend-guia-completo/>. Acesso em: 02 junho 2023.
[8] Front-end. Disponível em: <https://ebaconline.com.br/blog/desenvolvedor-front-end-o-que-faz>. Acesso em: 02 junho 2023.
[9] REACT. Disponível em: <https://pt-br.legacy.reactjs.org/>. Acesso em: 02 junho 2023.
[10] MUI. Disponível em: <https://mui.com/>. Acesso em: 02 junho 2023.

visualização amigável e tornando o sistema totalmente responsivo para dispositivos moveis e desktop.

A decisão pelo *PostgreSQL*[11] se deu em grande parte devido à sua natureza de código aberto. Isso significa que é livre para ser usado, modificado e distribuído, oferecendo uma grande flexibilidade tanto em termos de adaptação às necessidades específicas do projeto, quanto em relação ao custo, uma vez que não há taxas de licenciamento. Outro aspecto importante é que o *PostgreSQL* é um banco de dados relacional, um modelo amplamente utilizado que organiza os dados em uma forma que permite fácil acesso e manipulação.

O sistema está hospedado em nuvem e conta com uma área administrativa que possibilita aos coordenadores de cursos de graduação e mestrado cadastrar avaliações de forma simplificada, sem a necessidade de conhecimentos técnicos avançados. Também, com um painel de controle visual, utilizando gráficos e dados estatísticos para oferecer uma visão abrangente e detalhada de todas as avaliações realizadas e todas as respostas. Ademais, o sistema possibilita a geração automática de relatórios que, exportados nos formatos DOCX e CSV, apresentam os resultados de maneira estruturada. Os administradores podem então formatar esses documentos de acordo com o modelo original do relatório da CPA.

Um aspecto fundamental do sistema é a preservação da confidencialidade dos usuários que respondem aos questionários. Todos os dados são tratados de forma sigilosa, garantindo a privacidade dos participantes, as senhas são criptografadas usando o ASP.NET Core Identity[12]. Os usuários conseguem acessar o sistema e, caso exista uma avaliação disponível para o semestre atual, têm a possibilidade de responder de maneira rápida e fácil, com todas as respostas sendo atualizadas em tempo real para o administrador. É importante enfatizar que o administrador não possui acesso aos nomes e dados pessoais dos usuários, somente à resposta final de cada avaliação.

De forma geral, este trabalho demonstra o processo completo de desenvolvimento de um *software*, desde a criação da base de dados até a implantação do sistema em um ambiente de produção. Ao fornecer uma solução completa para a Comissão Própria de Avaliação, o sistema visa contribuir para a melhoria contínua das instituições de ensino, proporcionando uma avaliação institucional eficiente, confiável e de fácil utilização. O *software* foi projetado com a flexibilidade de incorporar diversos módulos, graças à sua arquitetura baseada em

[11] PostgreSQL. Disponível em: <https://www.postgresql.org/about/>. Acesso em: 02 de junho 2023.
[12] ASP.NET Core Identity. Disponível em: <https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-6.0&tabs=visual-studio>. Acesso em: 03 junho 2023.

APIs. Isso significa que o sistema não só permite a integração APIs internas, mas também está preparado para consumir APIs externas conforme a necessidade.

Embora não tenha sido realizado testes de desempenho específicos, o sistema foi construído com as mais atuais tecnologias do mercado, utilizando *frameworks* avançados que são conhecidos por oferecer desempenho superior. Esperamos que o sistema seja leve e rápido, com consultas ao banco de dados sendo realizadas em milésimos de segundos, proporcionando alto desempenho em produção. Essa expectativa é fundamentada na eficiência comprovada dos *frameworks* e das tecnologias que escolhemos usar no projeto.

1. REVISÃO BIBLIOGRÁFICA

1.1. Avaliação Institucional

Para compreender a importância da avaliação institucional, é essencial mergulhar em sua história. Leite, Tutikian e Holtz (2000) ressaltam que o início das avaliações em universidades data de 1977, sendo inicialmente conduzido pela Coordenação de Aperfeiçoamento do Pessoal de Nível Superior (CAPES). Desde então, muitos estudos foram realizados com o objetivo de implementar um sistema de avaliação amplo, capaz de analisar os traços de uma universidade.

Segundo Carvalho, Oliveira e Lima (2018), o caminho para uma avaliação institucional mais sólida foi repleto de desafios. Requiriu uma série de ajustes e decisões coletivas para garantir que todos os objetivos fossem atingidos, que incluem a avaliação institucional, a avaliação de cursos e o Enade (Exame Nacional de Desempenho de Estudantes), uma avaliação padronizada aplicada a estudantes para medir a qualidade do ensino nas instituições.

Conforme Seldin e Miller (2009), a avaliação institucional oferece uma oportunidade para as instituições medirem a eficácia dos processos educacionais e administrativos, ao passo que evidenciam realizações. Esta prática é fundamental para a melhoria contínua, pois promove a responsabilidade, a transparência e a melhoria da qualidade do ensino.

A avaliação institucional tem sido amplamente adotada na maioria das instituições de ensino superior, consolidando-se como uma prática essencial no contexto educacional (LEITE; TUTIKIAN; HOLTZ, 2000). Todavia, a efetiva implementação desta avaliação requer uma abordagem meticulosa e um planejamento cuidadoso (CARVALHO; OLIVEIRA; LIMA, 2018). Nesse sentido, destaca-se a importância do comprometimento da alta administração, assim como

a imprescindível participação de todos os envolvidos no processo educacional, como professores, estudantes e demais colaboradores da instituição.

1.2. A Comissão Própria de Avaliação (CPA)

A Comissão Própria de Avaliação (CPA) é um órgão responsável pela autoavaliação institucional nas instituições de ensino superior, cuja existência e atuação são determinadas pelo Sistema Nacional de Avaliação da Educação Superior (SINAES), regulamentado pela Lei nº 10.861, de 14 de abril de 2004, no Brasil. A CPA tem a função de conduzir os processos internos de avaliação da instituição, de sistematização e de prestação das informações solicitadas pelo Instituto Nacional de Estudos e Pesquisas Educacionais Anísio Teixeira (INEP).

Os resultados da autoavaliação promovida pela CPA são fundamentais para a melhoria da qualidade da educação e para a manutenção ou ajustes do projeto pedagógico, sendo uma ferramenta importante para o planejamento e o desenvolvimento institucional (DIAS SOBRINHO, 2004).

Entretanto, a CPA enfrenta diversos desafios na condução de atividades, incluindo:

1. A dificuldade de envolver toda a comunidade acadêmica no processo de autoavaliação, que inclui alunos, professores, funcionários e coordenadores de cursos (CARVALHO; OLIVEIRA; LIMA, 2018);
2. A complexidade em lidar com a grande quantidade de dados que precisa ser coletada, analisada e interpretada;
3. A necessidade de garantir que os resultados das avaliações sejam utilizados de maneira efetiva na melhoria contínua da qualidade da instituição.

Dado estes desafios, é fundamental que a CPA seja assistida por um sistema de gestão de avaliações eficiente. Este sistema pode ajudar na coleta, análise e interpretação dos dados, facilitando a participação de toda a comunidade acadêmica e garantindo a transparência do processo. Além disso, pode contribuir para a utilização dos resultados na tomada de decisões estratégicas que visem a melhoria contínua da instituição (GALDINO, 2006).

Por fim, é importante ressaltar que a efetividade da CPA é fundamental para a melhoria da qualidade da educação superior, já que é por meio dela que a instituição consegue se autoavaliar e identificar os pontos fortes e fracos, contribuindo para a constante evolução do processo educativo (GALDINO, 2006).

1.3. Sistemas de Avaliação

Os sistemas de avaliação institucional desempenham um papel crucial no funcionamento eficaz da CPA. Este fornece um meio estruturado de coletar e analisar dados, facilitando a autoavaliação e a melhoria contínua.

O impacto desses sistemas na eficiência da CPA é significativo, como apontado por Galdino (2006), um sistema de avaliação bem projetado pode facilitar o trabalho da CPA, fornecendo uma estrutura clara e diretrizes para a avaliação. No entanto, esses sistemas também podem ser complexos e exigir uma quantidade significativa de tempo e recursos para implementar e gerenciar.

Adicionalmente, é fundamental destacar que, mesmo que esses sistemas ofereçam uma estrutura útil, não eliminando a necessidade de uma avaliação ponderada e crítica. A CPA deve manter uma análise contínua da eficiência do sistema de avaliação e ajustá-lo conforme necessário para atender às demandas específicas da instituição (GALDINO, 2006).

1.4. C#, .NET, React.js e PostgreSQL em Desenvolvimento de Software

Durante o desenvolvimento do projeto, foi dada ênfase ao back-end, elaborado em uma API com C#, e empregamos as seguintes tecnologias principais: C#, .NET, React.js e *PostgreSQL*. Estas tecnologias são amplamente reconhecidas no campo do desenvolvimento de *software*, trazendo benefícios expressivos para a construção de aplicações.

As APIs, ou Application Programming Interfaces, desempenham um papel crucial na moderna engenharia de software, servindo como um intermediário entre diferentes componentes de software e permitindo que se comuniquem e troquem informações de maneira eficiente (MENG, STEINHART & SCHUBERT, 2018). As APIs definem os métodos e dados que um componente de *software* deve fornecer e são fundamentais para o desenvolvimento de aplicações complexas e interoperáveis (MENG, STEINHART & SCHUBERT, 2018).

Quando focamos especificamente na API com C# criada para este projeto, devemos ressaltar que foi construída utilizando o ASP.NET Core, um *framework* moderno para desenvolvimento de aplicações web e APIs, ideal para o C#, desenvolvido pela *Microsoft* (MICROSOFT, 2023). A combinação de C# com o ASP.NET Core fornece uma maneira poderosa e flexível de desenvolver APIs robustas e escaláveis. O ASP.NET Core foi projetado com um foco especial na performance e permite muito controle sobre o comportamento da aplicação, ao mesmo tempo que oferece muitos recursos para acelerar o desenvolvimento.

O uso de C# no desenvolvimento de APIs proporciona robustez e eficiência, características atribuídas à essa linguagem de programação. A extensa biblioteca

do *.NET Framework* agiliza e torna o processo de desenvolvimento mais eficaz (MICROSOFT, 2023). Adicionalmente, C# é uma linguagem fortemente tipada, o que auxilia na prevenção de diversos erros de programação, aprimorando assim a qualidade do código (MICROSOFT, 2023).

React.js é uma biblioteca *JavaScript*[13] de código aberto para construção de interfaces de usuário, inicialmente desenvolvida pelo Facebook, agora mantida pela Meta e pela comunidade *open-source*. É conhecido pelo modelo de programação declarativo e eficiente que facilita a criação de interfaces interativas. A arquitetura baseada em componentes do React.js permite a reutilização de código, melhorando a eficiência do desenvolvimento e a manutenção do *software* (WATMORE, 2020; CODE INSTITUTE, S.D.; TORQUE, 2021).

PostgreSQL é um sistema de gerenciamento de banco de dados relacional de código aberto que é conhecido pelo desempenho, robustez e capacidade de lidar com grandes volumes e variedade de dados, além de ter um forte cumprimento dos padrões SQL, tornando-o uma escolha popular para muitas aplicações web (MOMJIAN, 2001).

Em projetos semelhantes, a combinação dessas tecnologias tem proporcionado muitos benefícios. Por exemplo, em um tutorial publicado pela *Microsoft* (2023), a combinação de C#, .NET, React.js foi demonstrada para criar uma aplicação ASP.NET Core com uma interface de usuário React.js.

O sistema foi desenvolvido para operar de forma independente, sem a necessidade de integração com um *Learning Management System* [14] (LMS). Ele se baseia nas tabelas de usuários, turmas e disciplinas para seu funcionamento, simplificando o processo de implementação e evitando a necessidade de manter conexões com sistemas externos. A arquitetura flexível do nosso sistema permite futuras integrações com LMSs, se for identificada essa necessidade.

A seguir, é exposto a construção do sistema, abordando aspectos tecnológicos de desenvolvimento, o uso de determinados *frameworks* e bibliotecas, e a implementação do sistema gerenciador de banco de dados, bem como os diagramas para melhor entendimento do sistema, arquitetura do sistema e geração de relatórios finais, e implementação do sistema em produção.

---

[13] JavaScript. Disponível em: <https://developer.mozilla.org/pt-BR/docs/Web/JavaScript>. Acesso em: 03 junho 2023.

[14] Learning Management System (LMS). Disponível em: <https://www.techtarget.com/searchcio/definition/learning-management-system#:~:text=A%20learning%20management%20system%20is,assess%20a%20specific%20learning%20process>. Acesso em: 19 jun. 2023.

2. Metodologia de Desenvolvimento

O desenvolvimento do *software* surgiu da necessidade de modernização e aprimoramento dos processos da CPA no Centro Universitário UniAcademia, localizada em Juiz de Fora, Minas Gerais. Foi utilizada a linguagem C# ASP.NET Core para o desenvolvimento do back-end, React.JS para o front-end e *PostgreSQL* como gerenciador de banco de dados. Diagramas ajudam a visualizar a arquitetura do sistema. Para implementação em ambiente de produção, utilizamos serviços de nuvem da Amazon Web Services[15] (AWS), incluindo Amplify[16], Route53[17] e Docker[18] para eficiência e robustez.

2.1. Ferramentas de Desenvolvimento:

2.1.1. Linguagens de Programação:

- Escolha de C# .NET para o Back-end

A escolha de utilizar o C# e o .NET no back-end do sistema se deu por algumas razões. A principal delas é que o .NET é um *framework* estável e maduro, amplamente reconhecido por seu suporte robusto para construir APIs Web eficientes e escaláveis (MICROSOFT, 2023). Esta plataforma também disponibiliza uma grande variedade de bibliotecas e ferramentas integradas que simplificam a implementação de funcionalidades complexas, como autenticação e medidas de segurança, tornando-a extremamente conveniente para os desenvolvedores.

Além disso, o .NET fornece um ambiente de desenvolvimento que permite muito controle, enquanto ainda mantém um nível significativo de abstração, tornando o desenvolvimento mais eficiente (FREEMAN, 2019).

- Escolha do React.js para o Front-end

O React.js é uma biblioteca JavaScript desenvolvida pelo Facebook para a construção de interfaces de usuário ricas e interativas (FACEBOOK, 2021). A escolha do React.js para o front-end do sistema também se baseia em várias razões, permitindo a criação de componentes reutilizáveis, que podem ser combinados de várias formas para construir interfaces complexas. Isto aumenta a manutenibilidade e a escalabilidade do código.

Além disso, o React.js é conhecido por sua performance, devido a sua implementação de um DOM virtual e algoritmos de reconciliação eficientes. Isto

[15] Amazon Web Services. Disponível em: <https://aws.amazon.com/pt/free/>. Acesso em: 03 junho 2023.
[16] Amplify. Disponível em: <https://aws.amazon.com/pt/amplify/>. Acesso em: 03 junho 2023.
[17] Amazon Route 53. Disponível em: <https://aws.amazon.com/pt/route53/>. Acesso em: 03 junho 2023.
[18] DOCKER. Disponível em: <https://www.docker.com/>. Acesso em: 03 junho 2023.

torna o React.js uma excelente escolha para aplicações que exigem atualizações de interface de usuário frequentes e rápidas.

- Escolha do *PostgreSQL* como Sistema de Gerenciamento de Banco de Dados (SGBD)

O *PostgreSQL* foi escolhido como o SGBD para este projeto devido à sua extensa lista de recursos, sua reputação de confiabilidade, desempenho e integração com várias ferramentas e bibliotecas.

PostgreSQL é um sistema de banco de dados relacional de código aberto e é amplamente reconhecido por sua robustez e conjunto completo de recursos, que incluem transações ACID, integridade referencial, visões, funções armazenadas, triggerrs, entre outros. Além disso, *PostgreSQL* é altamente extensível, permitindo aos usuários definir os próprios tipos de dados, operadores e funções agregadas (MOMJIAN, 2001).

A modelagem dos dados foi realizada levando em consideração a natureza relacional do *PostgreSQL*. As tabelas foram projetadas para manter a integridade referencial, evitando a duplicação de dados e facilitando a atualização, recuperação e deleção de dados. Ao modelar o banco de dados foi criada a tabela *AspNetUsers* servindo como a principal tabela de usuários, enquanto as tabelas *Evaluation*, *DataQuestions*, *Questions* e *QuestionTurma* são utilizadas para lidar com as avaliações e questões do sistema, enquanto a tabela *Answers* é utilizada para armazenar as respostas dos usuários.

A gestão dos dados é feita através da criação, atualização, recuperação e deleção de registros nas tabelas. Para manter a integridade dos dados, foram implementadas restrições de chave primária e estrangeira nas tabelas. Além disso, foram implementados índices para otimizar o desempenho das consultas.

- Aplicação no Projeto

No projeto em questão, a combinação de C# .NET e React.js permitiu a criação de uma solução escalável e de alto desempenho. O back-end, construído com C# .NET, proporciona uma API robusta e segura. O front-end, construído com React.js, oferece uma interface de usuário interativa e reativa. Esta combinação de tecnologias permite que o sistema lide efetivamente com as demandas de sua base de usuários.

2.2. Diagramas do Sistema:

2.2.1. Diagrama de Tabelas Relacionais (DTR)

O Diagrama de Tabelas Relacionais (DTR) ilustrado na Figura 1 é uma ferramenta que esboça a estrutura do banco de dados de um sistema. Esse

instrumento visualiza tabelas (ou entidades), colunas (atributos) e as relações entre as tabelas, proporcionando uma visão clara e sistemática do esquema do banco de dados. O DTR, alinhado com a abordagem de Larman para a modelagem de dados, fundamenta-se na análise e projeto orientados a objetos e no desenvolvimento iterativo, estratégias que são altamente aplicáveis aos sistemas de banco de dados relacionais que têm dominado a paisagem da tecnologia da informação nas últimas décadas (LARMAN, 2005).

Figura 1 – Diagrama de Tabelas Relacionais (DTR).

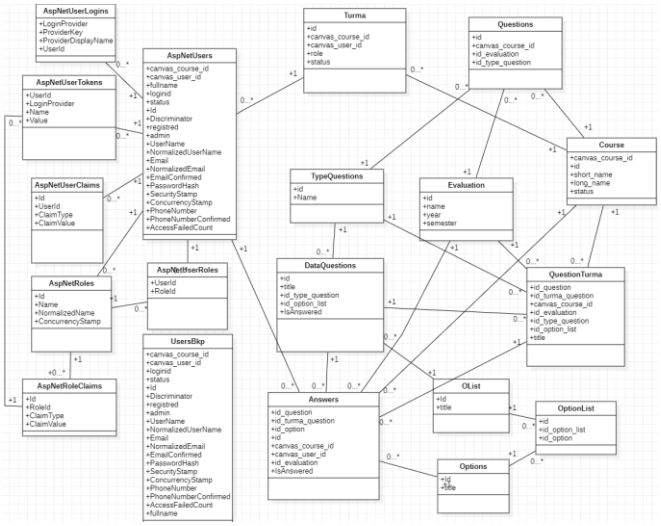
Fonte: Elaboração Própria.

2.2.2. Diagrama de Classes

No desenvolvimento orientado a objetos, o diagrama de classes como mostra na Figura 2 é uma representação visual das classes que compõem um sistema e das relações entre essas classes. Este diagrama é uma parte crucial do projeto de sistemas de informação e fornece uma visão geral da estrutura e arquitetura do sistema (FOWLER, 2003). No contexto do sistema de avaliação, o diagrama de classes contribui para o entendimento dos diferentes componentes do sistema que interagem entre si para realizar a funcionalidade desejada.

Este diagrama oferece uma visão geral de como o sistema está organizado e como diferentes partes interagem entre si. Com base neste diagrama, é possível não só identificar as classes presentes no sistema, mas também observar os relacionamentos existentes, fornecendo uma visão abrangente do sistema (LARMAN, 2005).

Figura 2 - Diagrama de Classes.



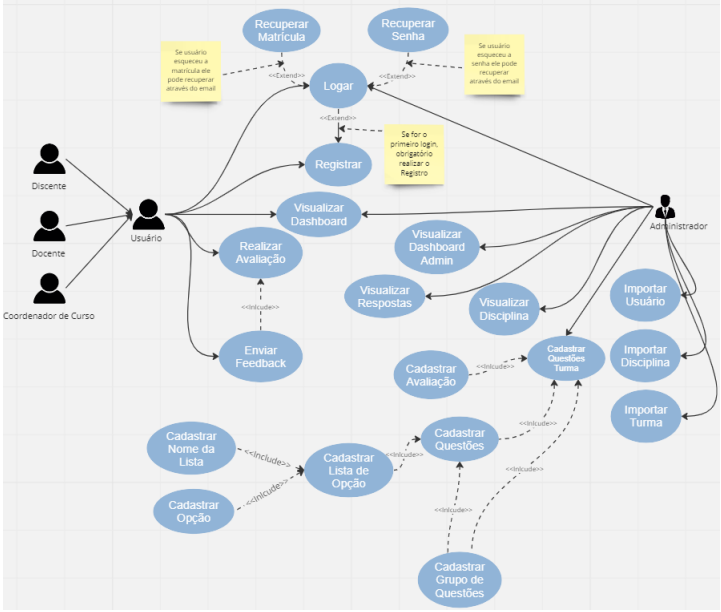
Fonte: Elaboração Própria.

2.2.3. Diagrama de Caso de Uso

O Diagrama de Caso de Uso da Figura 2Figura 3 é uma representação gráfica que ilustra as interações entre os atores (usuários ou sistemas externos) e o sistema em estudo. Esta representação ajuda a entender o que o sistema deve fazer em resposta a ações específicas de um ator. Portanto, é uma ferramenta importante no processo de análise e design de sistemas (LARMAN, 2005).

O sistema oferece funcionalidades distintas para usuários regulares e administradores. O usuário regular pode fazer login e logo após se registrar, recuperar matrícula ou senha, visualizar o dashboard principal, realizar avaliações (quando disponíveis) e enviar feedback. Por outro lado, o administrador, além de ter capacidades similares de login, pode visualizar o dashboard de administração, visualizar respostas e disciplinas, importar usuários, disciplinas e turmas, além de poder cadastrar questões para as turmas, desde que uma avaliação, questão e grupo de questões estejam previamente cadastrados, assim como uma lista de opções para a nova questão.

Figura 3 - Diagrama de Caso de Uso.



Fonte: Elaboração Própria.

2.2.4. Arquitetura do Sistema

A arquitetura do sistema representada na Figura 4 é uma representação conceitual que define a estrutura, comportamento e visões mais importantes de um sistema. Ela serve como o plano azul para o sistema e a visão compartilhada por todas as partes interessadas (BASS, CLEMENTS & KAZMAN, 2003). Neste caso, o sistema em discussão é composto por um back-end baseado em C# .NET com uma API REST e um front-end React.js.

- **Back-end: C# .NET API REST**

O back-end está em formato de API REST construída com C#. NET, um *framework* maduro e robusto conhecido por sua escalabilidade e desempenho (MICROSOFT, 2023). A API REST segue as principais convenções RESTful, incluindo o uso de URLs consistentes, o uso de verbos HTTP para indicar ações e a transferência de dados em formato JSON, e toda conexão e feita usando os controladores que contêm os métodos que fazem inserção e buscas no banco de dados.

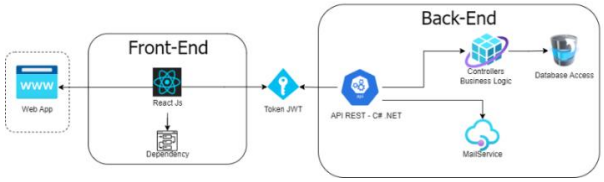
Além disso, o back-end também possui um serviço de email integrado, permitindo ao sistema enviar emails para os usuários quando necessário. Este é um recurso crucial para tarefas como recuperação de senhas e recuperação de matrículas no sistema.

- **Front-end: React.js**

O front-end do sistema foi construído com React.js, uma biblioteca JavaScript popular para a construção de interfaces de usuário interativas (FACEBOOK, 2021). A escolha do React.js permite o desenvolvimento rápido e eficiente de componentes reutilizáveis, que podem ser combinados para criar interfaces complexas.

A comunicação entre o front-end e o back-end é feita através de solicitações HTTP, usando tokens JWT para autenticação. Este método de autenticação oferece segurança, pois cada solicitação ao servidor é acompanhada por um token que o servidor pode verificar para autenticar o usuário (HASURA, 2020).

Figura 4 - Arquitetura do Sistema.



Fonte: Elaboração Própria.

As informações no sistema não são inseridas manualmente, mas sim alimentadas através da API desenvolvida com ASP.NET Core. Este método automatizado permite a inserção de dados de forma eficiente e precisa, minimizando a possibilidade de erros humanos e garantindo a integridade dos dados. Além disso, por meio da API, conseguimos gerenciar e controlar melhor o fluxo de dados, garantindo assim um melhor desempenho e segurança do sistema.

Em conclusão, a arquitetura do sistema apresentada na Figura 4 é bem estruturada, dividida entre front-end e back-end, ambos construídos com tecnologias adequadas para tarefas respectivas. A combinação de C# .NET para o back-end e React.js para o front-end resulta em um sistema robusto e escalável, capaz de fornecer uma experiência de usuário fluida e responsiva.

2.3. Implementação do Sistema em Produção

A implementação do sistema em produção foi um processo meticuloso que envolveu a utilização de diversas tecnologias contemporâneas, tais como Docker, NGINX, Amazon Route 53, Amazon EC2, Amazon Amplify e Amazon RDS.

O Docker foi empregado para a criação e gestão de containers, garantindo a portabilidade e isolamento do ambiente de execução do sistema. O NGINX, por sua vez, foi usado como servidor web para manipulação de solicitações HTTP e balanceamento de carga, proporcionando uma entrega de serviço eficiente e ágil.

As ferramentas da Amazon Web Services (AWS) desempenharam um papel central na hospedagem e manutenção do sistema. A Amazon Route 53 foi usada para o gerenciamento de DNS, facilitando a roteamento de usuários para os

recursos da web. Já a Amazon EC2 forneceu a capacidade de computação na nuvem, permitindo a rápida escalabilidade do sistema de acordo com as demandas. O Amazon Amplify, por sua vez, facilitou o desenvolvimento e a implantação de aplicações web e móveis escalonáveis.

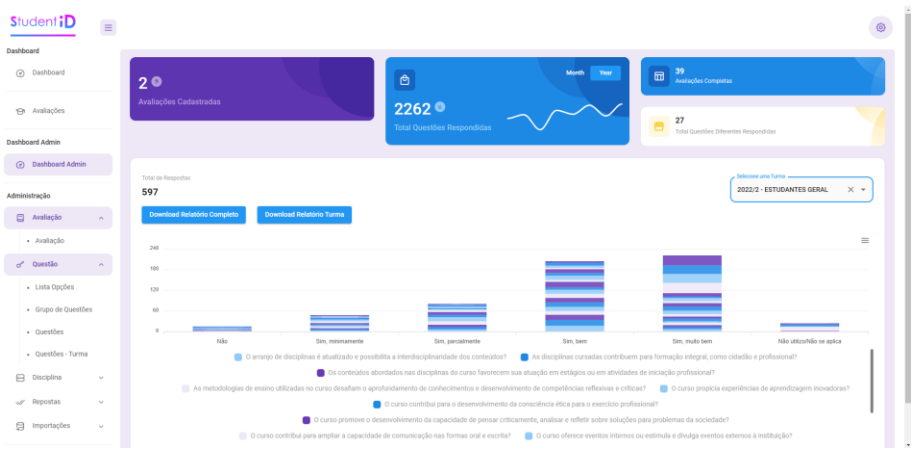
Por fim, a Amazon RDS forneceu um serviço de banco de dados relacional escalável e de alta performance, facilitando a gestão de operações de banco de dados, como replicação, configuração, *backup* e restauração.

O sistema, efetivado em produção, demonstra o sucesso da combinação de ferramentas modernas e escaláveis. Cada componente contribui para sua funcionalidade e eficiência, ressaltando a necessidade de uma seleção criteriosa de tecnologias para garantir uma arquitetura adaptável e ampliável. Esse processo exemplifica a força da arquitetura atual de *software* e do potencial da computação em nuvem.

Resultados são apresentados em um dashboard intuitivo, facilitando a análise dos dados. Ademais, a geração de relatórios detalhados é possível diretamente do painel de controle. Avaliações semestrais serão agilizadas por meio de um eficaz processo de importação de usuários, disciplinas e turmas. Graças à implementação estratégica de ferramentas gratuitas e de código aberto, o sistema foi concebido sem despesas financeiras, embora melhorias futuras possam requerer investimentos.

Na Figura 5, é apresentada uma visão do dashboard, que nos oferece a capacidade de gerar relatórios automáticos e visualizar dados referentes às respostas de maneira simplificada.

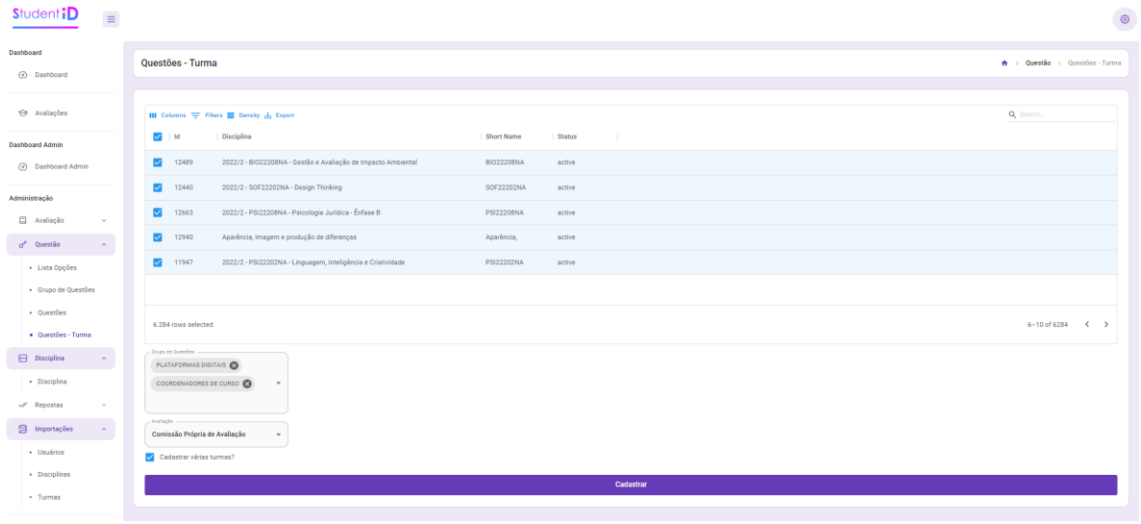
Figura 5 - Dashboard Admin



Fonte: Elaboração Própria.

A Figura 6 ilustra o processo de adicionar Grupos de Questões a múltiplas turmas simultaneamente. Essa metodologia foi desenvolvida para simplificar o cadastro das questões por turma. Que inicialmente só poderia ser feito uma por vez.

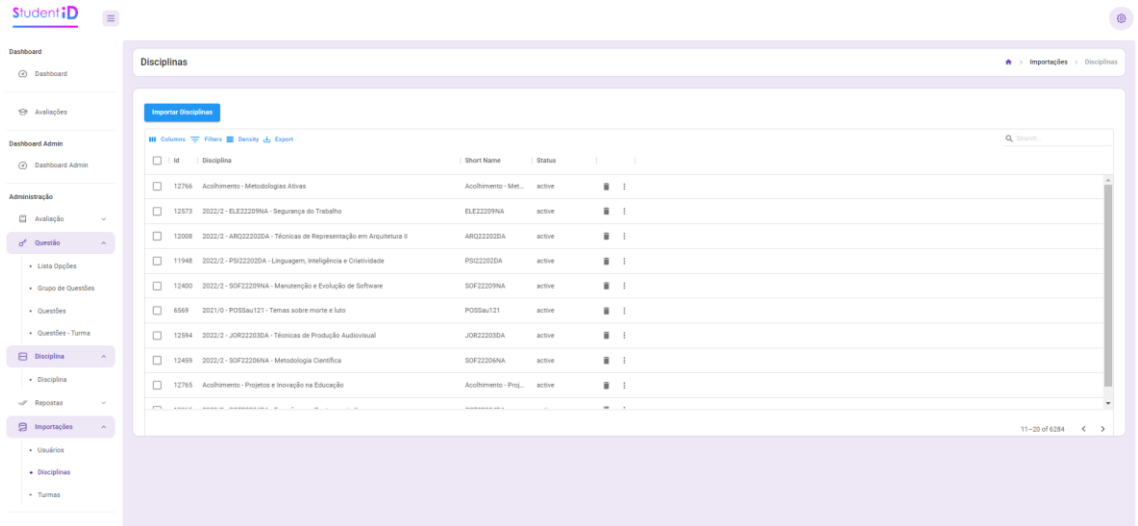
Figura 6 - Cadastro Questão Turma



Fonte: Elaboração Própria.

A fim de simplificar a importação de Usuários, Disciplinas e Turmas, foi criado um módulo especificamente para essa finalidade. Basta possuir o arquivo correspondente em formato CSV para cada categoria, conforme demonstrado na Figura 7.

Figura 7 - Módulo de Importações



Fonte: Elaboração Própria.

A interface do usuário comum apresenta-se consideravelmente diferente da interface do usuário administrador, com acentuada restrição em termos de opções

visíveis. O módulo crucial para o usuário comum é o de resolução de avaliações, identificado como "Avaliações". A Figura 8 exibe como se apresenta a seção de avaliações, onde o usuário pode visualizar as avaliações disponíveis para ele no momento. Se existirem mais avaliações, estas aparecerão listadas abaixo. Instruções para responder as avaliações são apresentadas, juntamente com um botão para iniciar a avaliação.

Figura 8 - Módulo de Avaliações dos Usuários

Fonte: Elaboração Própria.

Ao iniciar a avaliação, indicado na Figura 9, o usuário tem a possibilidade de responder a todas as questões vinculadas à sua turma. Informações sobre a avaliação, tais como a disciplina, o nome do professor da disciplina avaliada, se houver, bem como o grupo de questões, e as questões em si, são disponibilizadas. É imprescindível notar que não é permitido enviar questões sem respostas. Além disso, o usuário tem a opção de interromper e retomar a avaliação mais tarde, caso necessário. Após a finalização da avaliação, existe a oportunidade de avaliar o sistema.

Figura 9 - Realização de Avaliações e Feedback

Fonte: Elaboração Própria.

Alguns exemplos de demonstração do *software* podem ser vistos em https://youtube.com/playlist?list=PLvtXrfTfZZoTY6vSAh-k15TaD3KVoARtN. Além disso ficará um documento extra com informações complementares do sistema https://mega.nz/file/MYxG2BDL#pui0IKnIfeOMUcz8-lsj6ZVlJlXsqaGT9HDvDhH5L6s

3. Considerações Finais

Ao longo deste trabalho, demonstramos o desenvolvimento e a implementação de um sistema de avaliação para uma instituição de ensino. Esta solução digital é um avanço significativo na forma como a instituição coleta e analisa os feedbacks dos alunos, professores e coordenadores.

O sistema foi construído com tecnologias modernas e confiáveis, como C# ASP.NET Core para a API, React.js para o front-end e Amazon RDS para o banco de dados. Além disso, adotamos a infraestrutura da AWS para implementar o sistema em produção, aproveitando os benefícios do Docker e Nginx para um fluxo de implantação eficiente e seguro. Isso possibilitou um desempenho confiável e escalável para o sistema, além de garantir segurança por meio de certificados SSL.

O sistema foi desenvolvido utilizando a ferramenta Git[19] para controle de versão, que é uma prática padrão na indústria de desenvolvimento de *software* para rastrear e gerenciar mudanças no código fonte ao longo do tempo. Isso permite que múltiplas versões do sistema sejam mantidas e acessadas conforme necessário. Além disso, o sistema está hospedado em um repositório da universidade no GitHub[20], garantindo o acesso seguro e controlado a todas as versões do código-fonte. Isso não apenas facilita a manutenção e futuras melhorias do sistema, mas também permite a colaboração entre desenvolvedores, caso o sistema seja expandido ou modificado por futuros alunos.

Este sistema foi desenvolvido para auxiliar na melhoria do processo de avaliação da CPA, pretendendo ser uma ferramenta essencial para o UniAcademia. Com propósito prático, o sistema visa à utilização contínua e efetiva, contribuindo para o cotidiano da instituição. O planejamento para futuras operações prioriza agilidade, com processos integrados facilitando atualizações semestrais. Além disso, sua arquitetura modular garante a capacidade de evolução e ajuste a novas demandas institucionais.

---

[19] Git. Site oficial. Disponível em: <https://git-scm.com/.> Acesso em: 19 jun. 2023.
[20] GitHub. ces-jf/CPA. Disponível em: <https://github.com/ces-jf/CPA>. Acesso em: 19 jun. 2023

Durante o desenvolvimento, enfrentamos desafios significativos. O problema de usuários com múltiplas matrículas, por exemplo, foi uma questão complexa que foi resolvida com a criação de um script inteligente para gerenciar as matrículas e preservar a integridade dos dados. A resolução desse desafio é um exemplo do compromisso do projeto em fornecer uma solução eficaz e focada no usuário.

Em conclusão, este projeto demonstrou a importância e a viabilidade de implementar sistemas digitais em ambientes de educação. Desde a necessidade de adaptação contínua e desenvolvimento para enfrentar os desafios e mudanças que surgem ao longo do caminho.

Referências:

AMAZON WEB SERVICES. **Amazon Relational Database Service (RDS).** (s.d.). Disponível em: https://aws.amazon.com/pt/free/database/. Acesso em: 2 jun. 2023.

AMAZON WEB SERVICES. **O que é uma API?** Disponível em: https://aws.amazon.com/pt/what-is/api/. Acesso em: 02 junho 2023.

AWS. **AWS Amplify**. Disponível em: https://aws.amazon.com/pt/amplify/. Acesso em: 03 de junho 2023.

AWS. **AWS Free Tier**. Disponível em: https://aws.amazon.com/pt/free/. Acesso em: 03 de junho 2023.

BASS, Len; CLEMENTS, Paul; KAZMAN, Rick. **Software Architecture in Practice.** 4ª ed. São Paulo: Addison-Wesley Professional, 2021.

CARVALHO, H. A. de; OLIVEIRA, O. S. de; LIMA, I. A. de. **Avaliação Institucional em uma universidade pública brasileira multicâmpus: processos e desafios na qualificação da gestão.** Avaliação, Campinas, SP, v. 23, n. 1, p. 217-243, 2018. ISSN 1982-5765.

CODE INSTITUTE. **What is React.js? Introduction and Definition.** Disponível em: https://codeinstitute.net/global/blog/what-is-react-js/. Acesso em: 2 jun. 2023.

DOCKER. **Docker: Página inicial.** Disponível em: https://www.docker.com/. Acesso em: 03 de junho 2023.

EBAC. **Desenvolvedor Front-end: o que faz.** Disponível em: https://ebaconline.com.br/blog/desenvolvedor-front-end-o-que-faz. Acesso em: 02 junho 2023.

FACEBOOK. React – **A JavaScript library for building user interfaces**. 2021. Disponível em: https://reactjs.org/. Acesso em: 5 jun. 2023.

FIA. **Full Stack: o que é, o que faz e habilidades deste profissional.** Disponível em: https://fia.com.br/blog/full-stack-o-que-e-o-que-faz-e-habilidades-deste-profissional/. Acesso em: 02 junho 2023.

FOWLER, Martin. UML Distilled: **A Brief Guide to the Standard Object Modeling Language.** 3. ed. São Paulo: Addison-Wesley Professional, 2003. ISBN 0321193687.

GALDINO, M. N. D. **A autoavaliação institucional no ensino superior como instrumento de gestão. Universidade do Grande Rio "Prof. José de Souza Herdy**" /Fundação CESGRANRIO. Disponível em: https://unigranrio.com.br/_docs/cpa/autoav-inst-ensino-sup-instr-gestao-mary-galdino.pdf. Acesso em: 2 jun. 2023.

Git. **Site oficial**. Disponível em: https://git-scm.com/. Acesso em: 19 jun. 2023.

GitHub. **ces-jf/CPA**. Disponível em: https://github.com/ces-jf/CPA. Acesso em: 19 jun. 2023.

HARVE. **O que é Back-end: Guia completo.** Disponível em: https://harve.com.br/blog/desenvolvimento-web/o-que-e-back-end-guia-completo/. Acesso em: 02 junho 2023.

HASURA. **JWT Authentication: A Practical Guide.** Disponível em: https://hasura.io/blog/best-practices-of-using-jwt-with-graphql/. Acesso em: 5 jun. 2023.

LARMAN, Craig. **Utilizando UML e Padrões: Uma introdução à análise e projeto orientados a objetos e ao desenvolvimento iterativo.** 3ª ed. Porto Alegre: Bookman, 2005.

LEITE, D.; TUTIKIAN, J.; HOLTZ, N. (Orgs.). **Avaliação e compromisso: Construção e prática da avaliação institucional em uma universidade pública.** Porto Alegre: Ed. Universidade/UFRGS, 2000.

MENG, Michael; STEINHARDT, Stephanie; SCHUBERT, Andreas. **Application programming interface documentation: What do software developers want?.** Journal of Technical Writing and Communication, v. 48, n. 3, p. 295-330, 2018.

MICROSOFT**. .NET**. Disponível em: https://dotnet.microsoft.com/pt-br/. Acesso em: 02 junho 2023.

MICROSOFT. **Overview of ASP.NET Core Identity.** Disponível em: https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-6.0&tabs=visual-studio. Acesso em: 03 junho 2023.

MICROSOFT. **Tour of C#**. Disponível em: https://learn.microsoft.com/pt-br/dotnet/csharp/tour-of-csharp/. Acesso em: 02 junho 2023.

MICROSOFT. **Tutorial: Get started with C# and ASP.NET Core in Visual Studio. 2023.** Disponível em: https://learn.microsoft.com/en-us/visualstudio/get-started/csharp/tutorial-aspnet-core?view=vs-2022. Acesso em: 2 jun. 2023.

MOMJIAN, Bruce. **PostgreSQL: Introduction and Concepts**. 1ª ed. Boston: Addison-Wesley, 2001.

MDN. **JavaScript**. Disponível em: https://developer.mozilla.org/pt-BR/docs/Web/JavaScript. Acesso em: 03 de junho 2023.

MUI. **MUI: Página inicial**. Disponível em: https://mui.com/. Acesso em: 02 junho 2023.

NGINX. (s.d.). Disponível em: https://www.nginx.com/. Acesso em: 2 jun. 2023.

POSTGRESQL. **PostgreSQL: About, 2023.** Sobre. Disponível em: <https://www.postgresql.org/about/>. Acesso em: 02 de junho 2023.

REACT. **React: Página inicial.** Disponível em: https://pt-br.legacy.reactjs.org/. Acesso em: 02 junho 2023.

SELDIN, P.; MILLER, J. E. **The academic portfolio: a practical guide to documenting teaching, research, and service**. 1ª ed. San Francisco, CA: Jossey-Bass, 2009.

TechTarget. **Learning Management System** (LMS). Disponível em: https://www.techtarget.com/searchcio/definition/learning-management-system#:~:text=A%20learning%20management%20system%20is,assess%20a%20specific%20learning%20process. Acesso em: 19 jun. 2023.

TORQUE. **How to Create Powerful Interfaces Using React**. Disponível em: https://torquemag.io/2021/02/create-interfaces-using-react/. Acesso em: 2 jun. 2023.

WATMORE, Jason. **How to use the Facebook SDK in a React App**. Disponível em: https://jasonwatmore.com/post/2020/10/28/react-facebook-how-to-use-the-facebook-sdk-in-a-react-app. Acesso em: 2 jun. 2023.