OSLIVE: PERSISTÊNCIA DE DADOS E CONSTRUÇÃO DE UMA ARQUITETURA BACKEND PARA O OSLIVE

Jansley Carvalho Mendes Corrêa[1], Fábio Castro Araújo[1],

[1] Departamento de Computação – Centro Universitário Luterano de Palmas (ULBRA/PALMAS)

jansley@rede.ulbra.br, fabio.araujo@ulbra.br

Resumo. O OSLive é uma plataforma educacional voltada para o ensino de Sistemas Operacionais (SO), oferecendo recursos interativos para simulações e exercícios. No entanto, atualmente, a plataforma apresenta algumas limitações, como a ausência de um histórico de simulações e a impossibilidade de revisitar simulações passadas. A implementação de recursos de persistência de dados surge como uma solução para melhorar a experiência de alunos e professores, eliminando tais limitações e permitindo posteriormente uma análise de dados para evidências de aprendizado. Este trabalho aborda especificamente a criação de um módulo para armazenamento de dados no OSLive, uma plataforma educacional para o ensino de Sistemas Operacionais (SO). O problema central consiste em como armazenar dados dos módulos do OSLive para controlar acesso a eles e garantir a persistência. O desenvolvimento de uma API e a utilização de um banco de dados podem viabilizar esse controle de acesso e essa persistência. O objetivo principal deste trabalho é a implementação dos recursos necessários para a persistência dos dados dos módulos do OSLive. Os objetivos específicos incluem a elicitação e definição dos requisitos, a documentação de entidades e informações a serem persistidas, a modelagem do banco de dados, a construção do banco de dados e a definição arquitetural da API.

1. Introdução

Uma das disciplinas base dentro da matriz curricular da computação é a disciplina de Sistemas Operacionais (SO). Essa relevância se explica pela importância e presença quase ubíquas dos SOs no nosso cotidiano, seja para profissionais atuantes na área da computação ou para usuários em geral. Profissionais da área da computação dependem diretamente desses sistemas em suas atividades diárias, enquanto usuários comuns os utilizam em suas interações cotidianas com dispositivos eletrônicos, como computadores e smartphones. Como Tanenbaum (2009) explica em alto nível, "o programa que esconde a verdade sobre o hardware ao programador e apresenta uma visão simples e agradável de arquivos que podem ser lidos e gravados é, obviamente, o SO". Embora seja uma disciplina de extrema relevância, é densa e apresenta desafios significativos no que diz respeito ao ensino e à aprendizagem de como os SOs funcionam na prática.

Alguns autores apresentam a complexidade da disciplina. Buendía e Cano (2006) argumentam que:

> "os estudantes devem adquirir bases sólidas em sistemas operacionais e questões relacionadas. Especificamente, os alunos devem aprender os princípios de como funciona um SO, quais são os principais componentes; como um SO faz interface entre um computador e o usuário; e como os

> recursos do computador são gerenciados (por exemplo, memória, dispositivos de armazenamento ou unidades de processamento). Além disso, os alunos também devem estudar como o sistema fornece comandos para interagir com computadores. (...) O trabalho prático é necessário para reforçar os conceitos previamente explicados nas aulas teóricas e para ajudar os alunos a adquirir competências técnicas."

Anderssen e Myburgh (1992) citam que "os alunos que atendem um curso de SO geralmente experimentam dificuldades em dominar conceitos abstratos complexos, como concorrência, escalonamento de processos e deadlock". De maneira similar, no trabalho de Pamplona et al. (2018), os autores citam que "lecionar um curso de graduação em SO sempre foi um desafio. Os alunos tradicionalmente têm dificuldade em compreender os conceitos de SO como concorrência, memória virtual e troca de contexto". Ainda, Webb e Taylor (2014) citam uma dificuldade específica desta disciplina, onde "os alunos apresentam dificuldade em analisar cenários de troca (trade-off) e interpretar as implicações de soluções concorrentes".

Tais desafios e dificuldades motivam a busca por estratégias para o ensino e aprendizagem de SOs, unindo teoria e prática, como frameworks práticos, construção e disponibilização de ambientes de simulação ou sistemas educacionais, cada qual com um foco específico.

É possível identificar trabalhos e pesquisas com propósitos de ensino diferentes, seja com nível mais baixo de abstração, auxiliando estudantes a codificar e testar partes de SOs por meio de ambientes de simulação (Hughes e Pfleeger, 1978; Christopher et al., 1993), ou abordagens mais alto nível, com o intuito de simular e exibir a execução de algoritmos de escalonamento, resolução de exercícios e outros (Zareie e Najaf-Zadeh, 2013; da Silva et al., 2020).

O sistema OSLive é um exemplo de software educacional para o ensino de SOs que vem sendo aplicado na prática. Este vem sendo evoluído ao longo dos últimos semestres por alunos de computação do CEULP/ULBRA, cujas evoluções foram apresentadas nos trabalhos de Santos et al. (2017), Borghesan et al. (2021), Figueiredo e Marioti (2021) e da Silva et al. (2020).

As funcionalidades do OSLive até o momento são:

- simulação de algoritmos de escalonamento de processos (FIFO (First In First Out), SJF (Shortest Job First), RR (Round Robin), dentre outros);
- exercícios sobre escalonamento de processos;
- simulação de algoritmos de paginação;
- simulação da gerência de memória paginação por demanda, com algoritmos de substituição de páginas fcfs e segunda chance;
- exercício pgd com subs (fcfs, seg. chance e histórico de bit de referência);
- simulação de algoritmos de segmentação;
- exercícios sobre a gerência de memória por segmentação.

Atualmente, o OSLive tem algumas limitações, pois não dispõe de um mecanismo de armazenamento para as simulações realizadas e dos dados dos usuários. Essa lacuna

na funcionalidade da plataforma impede um controle de usuários e também que usuários tenham um histórico de suas simulações.

Como implementar um módulo para armazenar os dados de usuário e exercícios do OSLive para controlar o acesso a eles e garantir a persistência?

Entende-se que o desenvolvimento de uma API específica, a modelagem e uso de um banco de dados relacional pode tornar possível o armazenamento das informações dos módulos do OSLive de forma a realizar o controle de acesso e permitir que eles possam ser acessados posteriormente.

Implementação dos recursos necessários para a persistência de dados dos módulos do OSLive.

- Elicitação e definição dos requisitos do OSLive
- Documentar entidades e informações a serem persistidas
- Modelagem, dicionário de dados e diagramas do banco de dados
- Construção do banco de dados
- Definição arquitetural da API
- Implementação da camada de controle de acesso
- Implementação dos endpoints para a persistência dos dados

O desenvolvimento e aplicação de software para fins educacionais em disciplinas de Sistemas Operacionais (SO) é uma realidade motivada pela não trivialidade de uma disciplina carregada de conceitos e práticas sobre as diversas responsabilidades de alta complexidade de SOs. "O problema essencial com os SOs é que mesmo os pequenos e simples são grandes e complexos" (Tanenbaum, 1987). Com isso, diversos grupos de pesquisa e universidades têm aplicado sistemas educacionais no ensino de SO com o objetivo de tornar mais prático o entendimento de como estes tipos de sistemas realmente funcionam e, consequentemente, melhorar a qualidade da aprendizagem.

O Departamento de Computação do Centro Universitário Luterano de Palmas (CEULP/ULBRA) vem desempenhando o papel de manter um sistema de software educacional (SSE) desenvolvido pelos próprios alunos dos cursos de computação. OSLive, como é chamado, é uma plataforma online que disponibiliza recursos de simulação de algoritmos e exercícios que auxiliam na compreensão dos conteúdos relacionados a SO. Este sistema já tem sido aplicado semestre a semestre, contudo, apesar de suas funcionalidades e recursos atuais, avanços e melhorias específicos tem sido necessário.

O OSLive na sua versão atual (2022) possui as seguintes limitações: a) não há um histórico de simulações executadas; b) não é possível rever simulações de algoritmos que foram executados previamente.

Entende-se que estas limitações indicam a necessidade de recursos para a persistência dos dados de simulações e suas propriedades de modo que cada estudante possa revisitar as simulações disparadas e que os professores possam ter bases de simulações prontas para diversos cenários (inputs) de execução dos algoritmos, estando propriamente armazenados para utilização em sala de aula.

Portanto, a disponibilização de recursos de persistência de dados no OSLive poderia impactar positivamente na praticidade do uso da plataforma por alunos e professores. A persistência dos dados tem o potencial de maneira indireta, apoiar na geração de evidências sobre os níveis reais de aprendizado entre os alunos e em determinados contextos, uma vez que os dados de simulações e dos discentes passarão a ser propriamente mantidos em bases de dados.

2. Referencial Teórico

2.1. Desafios no Ensino e Aprendizagem de SOs

A disciplina de SO é fundamental dentro do currículo de Ciência da Computação e áreas relacionadas. O SO é o software mais importante de alguns dispositivos, responsável por gerenciar os recursos do sistema como a memória, os dispositivos de entrada e saída e os processos de execução. Do ponto de vista de educação, o ensino e aprendizagem de SO apresenta desafios aos professores e alunos devido à complexidade envolvida. Para muitos estudantes, os conceitos envolvidos em SO são muito desafiadores (Ziegler, 1999).

Um dos desafios mais significativos é a complexidade conceitual. Os Sistemas Operacionais são sistemas de software altamente complexos, responsáveis por gerenciar uma variedade de recursos. Esses sistemas operam em um nível muito próximo ao hardware, o que torna a compreensão de suas funcionalidades um desafio. Tanenenbaum (1987) diz que “o problema essencial com os sistemas operacionais é que mesmo os pequenos e simples são grandes e complexos”.

Alguns autores relatam e apresentam indícios sobre a dificuldade envolvida no contexto de ensino e aprendizagem na disciplina de SO. Timithy Bower (2006) relata que:

> “Em uma aula de sistemas operacionais, nós queremos que os alunos compreendam as estruturas de dados e algoritmos usados em sistemas operacionais reais. Como tal, as aulas de sistemas operacionais sempre incluem explicações pesadas sobre tópicos como arquitetura de sistemas, gerenciamento de dispositivos e E/S, gerenciamento de processos, gerenciamento de memória, sincronização e gerenciamento de sistema de arquivos. No entanto, as explicações por si só não são capazes de ilustrar esses princípios em ação.” (Timithy Bower, 2006).

Do ponto de vista do tema "concorrência", Lincke (2005) afirma que multi-threading, concorrência e semáforos (incluindo problemas de produtor-consumidor, leitores-escritores etc.) são um grande desafio para estudantes de graduação em sistemas operacionais, e que um grande problema com o ensino da concorrência é que requer experiência/prática para aprender. Contudo, a aula expositiva tem sido o modo normal de ministração em cursos de ciência da computação.

Além da concorrência, os alunos enfrentam dificuldades em lidar com outros conceitos abstratos como escalonamento de processos e deadlock. Esses conceitos são fundamentais para o funcionamento dos SOs, mas podem ser desafiadores de entender, especialmente sem exemplos práticos e claros que ajudem os alunos a visualizarem esses conceitos. Como afirmou Hughes (1978):

> "Embora muito possa ser aprendido através da leitura de livros apropriados, a experiência em programação ainda é necessária para completar a educação de um programador de sistemas. Por exemplo, é prática comum incluir a programação de vários componentes de um compilador em um curso sobre construção de compiladores. Infelizmente, uma experiência semelhante raramente é oferecida em cursos de Sistemas Operacionais."

Os relatos previamente apresentados indicam a necessidade de transição da teoria para a prática. O ensino de Sistemas Operacionais não se resume à compreensão dos conceitos. Os alunos também precisam aprender a aplicá-los na prática. Hughes (1978) afirma que "claramente, os alunos devem ter a oportunidade de escrever um pequeno sistema operacional". Isso envolve a compreensão de algoritmos de escalonamento de processos, gerenciamento de memória, sincronização de processos e outros aspectos práticos dos SOs.

A falta de contexto do mundo real é mais um desafio. Os alunos muitas vezes lutam para relacionar os conceitos de Sistemas Operacionais com situações da vida cotidiana. A ausência de exemplos práticos e de conexão com cenários reais pode dificultar a compreensão e a aplicação desses conceitos.

Para superar esses desafios, o ensino de Sistemas Operacionais muitas vezes exige o uso de ferramentas interativas que permitem aos alunos experimentar e visualizar os conceitos em ação. Também é importante criar tarefas práticas e projetos que permitam aos alunos demonstrar seu conhecimento, uma vez que muitos dos conceitos não podem ser facilmente avaliados por meio de exames tradicionais.

O ensino de Sistemas Operacionais é um desafio devido à natureza complexa e abstrata dos conceitos envolvidos. Isto nos leva a compreender a necessidade de adoção de estratégias mais eficazes, ferramentas interativas e exemplos práticos para melhorar a compreensão e o aprendizado dos alunos, preparando-os para enfrentar os desafios reais no campo de sistemas operacionais e computação.

2.2. Estratégias para o Ensino e Aprendizagem de Sistemas Operacionais

No ensino de SO é essencial adotar estratégias pedagógicas que auxiliem os alunos a superar os desafios inerentes à compreensão de conceitos complexos. A dificuldade de aprendizado da disciplina de SO tem sido objeto de estudo por muitos pesquisadores, resultando na proposta de diversos tipos de abordagens e estratégias para facilitar a aprendizagem de SO. Perez-Davila (1995) disse:

> "Ministrar aulas de sistemas operacionais sempre foi um desafio. Há uma quantidade considerável de teoria que precisa ser abordada e, ao mesmo tempo, como professor você sente a obrigação de envolver o aluno em pelo menos uma quantidade mínima de experiência prática. Isto não é feito facilmente e no passado várias abordagens foram tentadas. Eles vão desde tarefas simples de simulação até o uso de um pequeno sistema operacional desenvolvido do zero para uso em sala de aula, por exemplo MINIX [Tan87] XINU [Com88]."

Pamplona, Medinilla e Flores (2018) realizaram um mapeamento sistemático com o objetivo de analisar as abordagens que têm sido usadas para melhorar o ensino e a aprendizagem de SO. Um total de 55 artigos foram incluídos no estudo. As principais abordagens identificadas para melhorar o ensino e a aprendizagem de OS foram:

- Sistema operacional real: na abordagem de sistema operacional real, os alunos são envolvidos no desenvolvimento código do kernel para um sistema operacional real. Hughes (1978) observou que “os alunos que têm a chance de escrever um SO, mesmo que seja simples, adquirem uma compreensão mais profunda dos princípios e desafios envolvidos”. Esses projetos facilitam a transição da teoria para a prática e consolidam o aprendizado.
- SO instrucional: os estudos incluídos na abordagem de SO Instrucional propõem o uso de projetos de programação de kernel que evitam a complexidade do sistema operacional real. O objetivo é proporcionar aos alunos um sistema operacional real e de fácil entendimento, para que possam estudar e modificar. Os laboratórios práticos desempenham um papel fundamental ao proporcionar aos alunos a oportunidade de experimentar conceitos de SO em um ambiente controlado. De acordo com Perez-Davila (1995) o uso de laboratórios permite que os alunos sejam guiados por meio de exercícios com crescente complexidade e fornecendo um estudo mais aprofundado de um sistema que os alunos provavelmente encontrarão quando saírem do curso.
- Projetos de programação: a abordagem de Projetos de Programação não envolve programação em nível de kernel. Em vez disso, está centrado em projetos menores, no nível do usuário. O principal princípio subjacente a esta abordagem é que a parte mais importante de um curso de SO é a compreensão dos conceitos.
- Ambientes de aprendizagem: o objetivo da abordagem de Ambientes de Aprendizagem é usar aplicativos de software para ensinar e aprender SO.
- Aprendizado colaborativo: na abordagem de Aprendizagem Colaborativa, os estudos utilizam técnicas de aprendizagem colaborativa para melhorar a aprendizagem em sistemas operacionais. O termo “aprendizagem colaborativa” foi interpretado de diversas maneiras, mas o aspecto comum é a ênfase nas interações dos alunos, em vez do aprendizado como uma atividade solitária.

Figura 1. Interface do jogo StarFight (fonte: Jong et al., 2012)

- Jogos: esta abordagem baseia-se no uso de jogos como ferramenta de ensino e aprendizagem. Este método implementa jogos educativos para motivar os alunos e proporcionar-lhes oportunidades de praticar habilidades que a sala de aula tradicional pode não ter. Jong et al. (2012) desenvolveram um jogo (Figura 1) integrado ao conteúdo do curso e o utilizam no ensino da disciplina de SO. Ao jogar este jogo, os membros da equipe cooperam entre si para competir com outra equipe. Após experimentos, os autores relatam que o tipo de aprendizagem cooperativa online, multijogador e baseada em jogos utilizado na pesquisa se mostrou eficaz para melhorar o desempenho da aprendizagem e apresentou um efeito positivo no processo de aprendizagem. A figura 1 apresenta um snapshot da interface de um jogo que foi desenvolvido com o propósito de auxiliar os alunos da disciplina de SO. Jong et al. (2012) explica que nesta tela tem duas equipes participantes rotuladas como União e Tribo. Quando o jogo inicia ele decide aleatoriamente quem vai iniciar atacando, se é a equipe União ou Tribo. O jogo consiste em várias rodadas, onde a cada rodada um membro da equipe atacante escolhe uma pergunta para um membro da equipe que está na defesa. Na rodada seguinte, os times trocam suas posições de atacante e de defensor.
- Simuladores Gráficos: o princípio básico subjacente aos Simuladores Gráficos é que os alunos nunca vêem o interior de um sistema operacional, o que dificulta a compreensão do seu funcionamento. A solução proposta por estudos desta linha de pesquisa é fornecer uma representação gráfica que permite aos alunos entender como um sistema operacional funciona observando os acontecimentos a cada momento. Perez-Davila (1995) entende que, por meio de simulações, pode proporcionar aos alunos uma compreensão mais aprofundada de um sistema real, reunindo teoria e prática de maneira realista.

> Simulações permitem que os alunos visualizem o funcionamento interno de um SO tornando o aprendizado mais eficaz.

O sistema OSLive é um exemplo de estratégia na categoria de simuladores gráficos para auxiliar no aprendizado de conceitos complexos e abstratos relacionados aos sistemas operacionais. Essa plataforma permite que os alunos interajam com simulações para aprimorar a compreensão dos conceitos de SO, fornecendo uma maneira prática de explorar conceitos de gerência de memória e paginação por demanda.

2.3. Arquitetura de Aplicações Web

A arquitetura de aplicações web é inerentemente ligada à capacidade de proporcionar acesso remoto a diversos recursos, desempenhando um papel crucial em diversos contextos, especialmente no campo educacional. O crescimento da Web trouxe mudanças para diversos setores. Agindo como um instrumento de globalização, reduzindo distância entre povos e criando uma infinidade de possibilidades (Dantas, 2003). A remoção das barreiras geográficas não apenas amplia o alcance do ensino, permitindo que alunos acessem simulações, laboratórios virtuais e conteúdo educacional de qualquer local com conexão à internet, mas também promove a colaboração e a flexibilidade de acesso.

Ao explorar a arquitetura de aplicações web, emergem conceitos e paradigmas essenciais para a concepção e implementação de sistemas distribuídos. A arquitetura não apenas fornece a estrutura fundamental para o desenvolvimento de aplicações acessíveis por diversos dispositivos, mas também introduz desafios significativos relacionados às decisões arquiteturais.

A definição precisa de arquitetura de software é uma tarefa desafiadora, Martin Fowler (2002) diz que:

> "Alguns dos padrões neste livro podem ser chamados arquiteturais, já que representam decisões importantes sobre essas partes; outros são mais sobre design e o ajudam a implementar essa arquitetura. Não faço nenhuma forte tentativa de separar esses dois, uma vez que aquilo que é arquitetural ou não é subjetivo".

Além de Fowler a universidade de Carnegie Mellon possui diversas definições e aborda a grande dificuldade que é definir a diferença exata entre implementação, design e arquitetura. Martin Fowler (2004) diz que "o termo arquitetura envolve a noção dos principais elementos do sistema, as peças que são difíceis de mudar. Uma fundação na qual o resto precisa ser construído".

Ao analisar uma aplicação e compreender as interações entre suas diversas partes, surge a necessidade de tomar decisões arquiteturais cruciais. Questões como a escolha do modelo em camadas, a seleção do framework para mapeamento de objeto relacional, a adoção de uma arquitetura de comunicação assíncrona e a decisão entre uma arquitetura monolítica ou de microsserviços são exemplos de desafios que podem definir o sucesso ou fracasso de um projeto.

A divisão em *layers* (camadas) representa uma separação lógica do código da aplicação, organizando-o em estruturas com responsabilidades distintas. O padrão MVC

é um exemplo dessa abordagem, proporcionando menor acoplamento, maior coesão e facilitando mudanças e manutenção. Outros autores propõem divisões adicionais de *layers*, conforme mencionado por Martin Fowler (2022).

A divisão em *tiers*, por sua vez, busca uma separação física entre as partes do sistema. Essa abordagem caracteriza *tiers* diferentes como componentes do sistema que operam em máquinas distintas. Um exemplo prático seria uma aplicação web em execução em um servidor e o banco de dados associado funcionando em outro servidor, configurando dois *tiers* distintos.

2.3.1. Arquitetura Monolítica e de micros serviços

Dentre as arquiteturas destacadas, duas merecem atenção especial: a monolítica e a de microsserviços. A arquitetura monolítica centraliza todas as funcionalidades em uma única aplicação, facilitando o desenvolvimento inicial, mas com o decorrer do tempo pode gerar dificuldades de escalabilidade e manutenção em projetos complexos.

Por outro lado, a arquitetura de microsserviços adota uma abordagem distribuída e assíncrona, dividindo a aplicação em serviços independentes. Essa modularidade proporciona maior flexibilidade, escalabilidade e facilidade de manutenção, mas introduz desafios de coordenação e comunicação entre os serviços.

2.4. Persistência de dados e banco de dados

Além da arquitetura de software, a persistência de dados tem uma importância significativa para o funcionamento eficiente de aplicações web. A capacidade de armazenar e recuperar informações de forma confiável é essencial. Para Peter Bailis (2015) os sistemas de banco de dados desempenham um papel crucial na organização, armazenamento, recuperação e gerenciamento eficientes de grandes volumes de dados. São fundamentais para inúmeras aplicações e setores. Além disso, os bancos de dados desempenham um papel fundamental na garantia da integridade, segurança e consistência dos dados, fornecendo mecanismos para garantir que a informação seja precisa e confiável. Nesse contexto, a escolha de um sistema de gerenciamento de banco de dados (SGBD) se torna fundamental. Existem diversas categorias de bancos de dados, cada uma projetada para atender a necessidades específicas de armazenamento e recuperação de dados.

- Bancos de dados relacionais: Os bancos de dados relacionais, como MySQL, SQL Server e PostgreSQL, são baseados no modelo relacional proposto por Edgar F. Codd em 1970, que define uma estrutura para armazenar dados em tabelas com colunas e linhas. As tabelas são relacionadas entre si por meio de chaves primárias e estrangeiras, permitindo que os dados sejam organizados e recuperados de maneira eficiente. Os bancos de dados relacionais também fornecem recursos para garantir a integridade e a consistência dos dados, como restrições de integridade referencial, e mecanismos de controle de concorrência.
- Bancos de dados NoSql: Os bancos de dados não relacionais surgiram para resolver as limitações dos bancos de dados relacionais tradicionais no tratamento de dados de grande escala, distribuídos e não estruturados. Os bancos de dados NoSQL são projetados para serem altamente escaláveis, flexíveis e tolerantes a falhas. Frequentemente eles também priorizam a

disponibilidade e a tolerância à partição em vez da consistência, o que é conhecido como teorema CAP. Peter Bailis (2015) fala que

> "No entanto, uma diferença notável e fundamental é que esses armazenamentos NoSQL frequentemente se concentram em fornecer melhor disponibilidade de operações por meio de modelos mais fracos, como foco explícito na tolerância a falhas"

2.5. OSLive: uma ferramenta de software educacional para o ensino de sistemas operacionais

O OSLive é uma plataforma web desenvolvida pelos alunos do CEULP/ULBRA, dedicada ao ensino de Sistemas Operacionais. Acessível em http://ulbra-to.br/oslive/, a ferramenta oferece módulos que ensinam sobre gerenciamento de memória e processos.

No módulo de processos, os alunos participam de exercícios e simulações práticas sobre escalonamento de processos. Já no módulo de memória, há simulações avançadas de algoritmos de paginação e segmentação.

Figura 2. Sistema OSLive

A interface intuitiva facilita a navegação, tornando a experiência educacional acessível. O OSLive não só atende às demandas da disciplina, mas também se destaca como uma contribuição significativa para o ensino de computação, promovendo a assimilação prática de conceitos complexos. Ao integrar tecnologia e pedagogia, a plataforma representa uma inovação valiosa, capacitando os estudantes para enfrentar desafios teóricos e práticos em sistemas operacionais. A figura abaixo apresenta a interface do sistema OSLive durante simulação de um algoritmo FIFO (First in, First Out).

3. Materiais e Métodos

Os materiais que foram utilizados no desenvolvimento deste trabalho se limitam aos recursos relacionados ao ambiente de projeto e desenvolvimento a serem configurados para possibilitar a criação de modelos e documentos técnicos, além da

codificação, testes e implantação de software. Portanto, foram utilizadas as seguintes ferramentas:

- Computador com processador Intel Core i7, 32gb RAM. Sistema operacional Windows 11 Pro. Utilizado no trabalho para o desenvolvimento da aplicação, para a construção do banco de dados e para os testes

- Banco de dados PostgreSQL: O PostgreSQL é um poderoso sistema de banco de dados relacional e de código aberto. Amplamente utilizado para armazenar e gerenciar dados em aplicativos web e outras soluções de software. Segundo a documentação (The PostgreSQL Global Development Group)

> "PostgreSQL é um sistema de gerenciamento de banco de dados objeto-relacional (ORDBMS) baseado no POSTGRES, versão 4.2, desenvolvido no Departamento de Ciência da Computação da Universidade da Califórnia em Berkeley. PostgreSQL é um descendente de código aberto do código original de Berkeley. Ele suporta grande parte do padrão SQL e oferece muitos recursos modernos."

  O PostgreSQL utiliza a linguagem SQL para consultar, modificar e gerenciar dados. Ele oferece recursos como suporte a transações, índices, segurança e escalabilidade. No trabalho esse banco de dados será utilizado para armazenar as informações dos usuários e as simulações realizadas na plataforma OSLive;

- Framework Django: Para a construção da camada de API do OSLive optou-se por utilizar o Django, um framework de desenvolvimento web em Python. Ele facilita a construção de aplicativos web robustos e escaláveis ao fornecer uma estrutura organizada para o desenvolvimento (Django, 2023). Django inclui um ORM (Object-Relational-Mapping) para interagir com bancos de dados, um sistema de templates para gerar conteúdo dinâmico, e um conjunto de ferramentas para lidar com tarefas comuns, como autenticação, URL routing. Este framework será o apoio para a construção da API responsável por receber informações dos módulos dos OSLive e persistir essas informações no banco de dados, essa API também fará o controle de permissões dos usuários.

- IDE Visual Studio Code para a codificação de software. Utilizado no trabalho para a escrita do código.

Considerando o problema e os objetivos do trabalho mencionados anteriormente, delineou-se uma metodologia com a ordem e descrição das etapas pertinentes para

alcançar o escopo pretendido. A figura abaixo ilustra a metodologia em alto nível, nas suas etapas e ordem. Cada etapa foi elucidada na sequência.

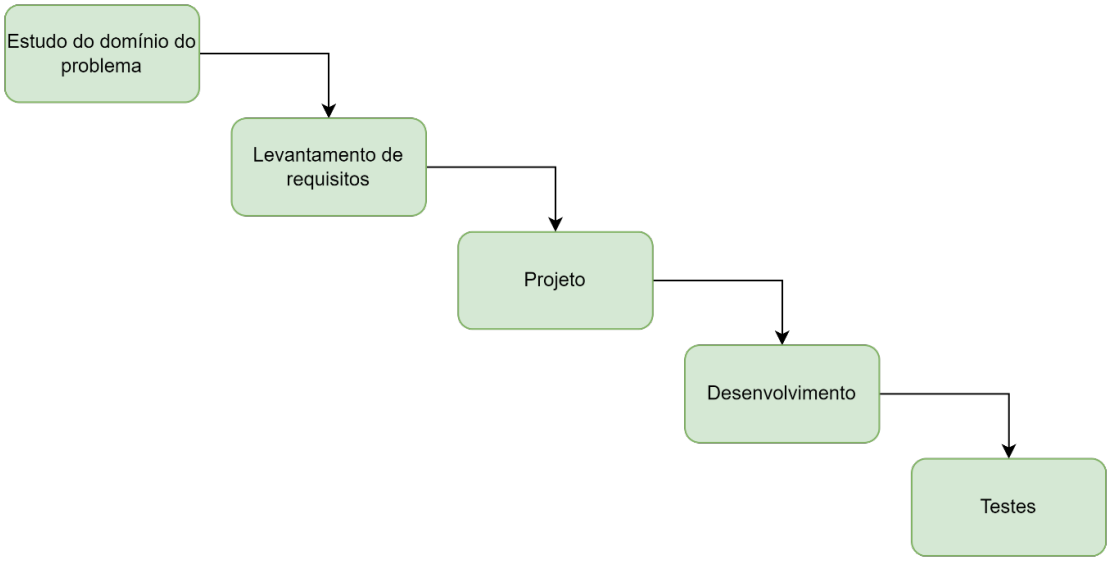

Figura 3. Fluxo da metodologia

A condução deste trabalho foi iniciada com o estudo do domínio do problema visando obter uma compreensão mais profunda das características da versão em que o OSLive se encontrava até o ano de 2024, das oportunidades de evolução e desafios do sistema. Para a melhor condução deste trabalho foi necessário entender como o sistema é utilizado, identificar problemas técnicos e oportunidades de modo a apoiar o levantamento de requisitos. Portanto, os seguintes tópicos foram endereçados na etapa de estudo do domínio do problema:

- Contexto Educacional: Do ponto de vista do contexto educacional, houve um exercício de compreensão de como o OSLive estava sendo aplicado no ensino, de modo a identificar fragilidades, ausência de requisitos funcionais considerados relevantes para o aprendizado dos alunos (usuários).
- Tendências Educacionais: Por meio do estudo de trabalhos identificados na literatura técnica, foi possível identificar tendências no ensino de sistemas operacionais. Entendeu-se que essa investigação revelou abordagens pedagógicas que foram candidatas a incrementar – ou não - os requisitos do software.
- Desafios Técnicos: Conduziu-se uma análise de documentações, manuais de usuário, código-fonte e infraestrutura do sistema a fim de identificar desafios técnicos que representaram oportunidades de evolução e melhoria do software.

A execução do estudo desses tópicos na etapa inicial de estudo do domínio do problema resultou em inputs e uma melhor preparação para dar seguimento com a próxima etapa de levantamento de requisitos uma vez que haverá uma compreensão mais sólida e detalhada do contexto em que o software se encontra e será aplicado.

Após o entendimento do domínio do problema, foi realizado o levantamento e análise de requisitos, que envolveu a identificação das necessidades dos usuários, professores e alunos, bem como a análise das fragilidades do software atual.

Para isso, as seguintes atividades foram realizadas:

1. Entrevistas com professores e alunos para capturar suas expectativas e necessidades com relação ao OSLive.
2. Análise dos requisitos elicitados e documentação
3. Priorização dos requisitos com base em sua importância e impacto na qualidade do software.

Tendo elicitado, documentado e priorizado os requisitos para a evolução do OSLive, projetou-se os requisitos funcionais e não-funcionais, etapa essencial em um processo de desenvolvimento de software que ocorre antes da fase de implementação ou desenvolvimento.

Nesta etapa foram elaborados planos detalhados e especificações técnicas que serviram como guia para o desenvolvimento, tendo atenção aos seguintes aspectos de software:

- Arquitetura de Software: visa projetar os requisitos que impactam a arquitetura de software. Isso inclui a escolha de tecnologias, frameworks e padrões de design que serão utilizados (ou modificados). Os principais elementos da arquitetura de software devem ser especificados, como componentes, módulos e a interação entre eles.
- Modelagem de Dados: a modelagem de dados envolve a definição da estrutura de dados que o software usará para armazenar informações. Nesta sub-etapa foram projetados os esquemas de banco de dados, tabelas, relacionamentos e estratégias de armazenamento de dados. A integridade dos dados e os requisitos de desempenho foram levados em consideração. Abaixo o resultado do diagrama de banco de dados.
- Documentação Técnica: as definições de projeto foram apropriadamente documentadas. Documentos técnicos que descrevem a arquitetura de software, a estrutura de banco de dados e outros aspectos importantes ao projeto foram gerados e armazenados. Essa documentação servirá como um recurso para desenvolvedores, analistas de qualidade e futuras manutenções do software.
- Revisão e Validação do Projeto: antes de prosseguir com o desenvolvimento, o projeto foi revisado e validado. Isso envolve a revisão por partes interessadas, como professores e especialistas em sistemas operacionais. Os ajustes necessários no projeto foram realizados nesta fase.

A etapa de Projeto foi essencial para suportar o desenvolvimento do software, de modo que fosse conduzido de forma eficiente e orientado por um plano adequado. O projeto ajudou a evitar problemas de design e a alinhar o software com as expectativas e requisitos dos usuários.

Com os requisitos especificados, planejados e projetados, foi iniciada a implementação das funcionalidades e melhorias no software. As seguintes práticas de desenvolvimento foram seguidas:

- Desenvolvimento de forma iterativa com o objetivo de exposição iterável dos incrementos para uma obtenção mais rápida do feedback do usuário e adaptabilidade. Durante o desenvolvimento foram realizadas reuniões regulares com os stakeholders para coletar feedback e garantir que o software esteja evoluindo de acordo com as expectativas.
- Controle de versão
- Testes unitários e de integração para verificar a qualidade das novas funcionalidades.

Após a implementação das novas funcionalidades iniciou-se a fase de testes. Isso incluiu: Testes de aceitação para verificar se os requisitos foram atendidos. Testes de segurança para proteger os dados dos usuários.

A metodologia descrita neste trabalho forneceu um processo estruturado para a evolução do OSLive. A abordagem incremental de levantamento de requisitos, priorização, especificação, desenvolvimento e testes permitiu uma evolução controlada e orientada pelas necessidades dos usuários. Com essa metodologia, foi possível conduzir o desenvolvimento para que o software educacional atendesse às expectativas dos professores e alunos, proporcionando uma experiência de aprendizado mais eficaz e condizentes com a realidade do contexto de ensino e aprendizagem da instituição.

4. Resultados e Discussões

4.1. Requisitos da plataforma

Os requisitos da plataforma OSLive foram definidos com o objetivo de atender às demandas específicas do contexto educacional. Primeiramente, priorizou-se a criação de um sistema robusto que pudesse suportar cargas simultâneas e garantir a estabilidade necessária para o uso contínuo em ambientes de ensino.

Cod:	Título	Descrição
RF01	Cadastro de Professor	O endpoint de cadastro de professores permite que usuários se registrem na plataforma, desde que possuam um e-mail institucional da Ulbra. Para realizar o cadastro, o usuário deve enviar um JSON contendo algumas informações essenciais.
RF02	Cadastro de Aluno	O endpoint de cadastro de alunos permite que estudantes se registrem na

		plataforma, desde que possuam um e-mail institucional da Ulbra. Para realizar o cadastro, o usuário deve enviar um JSON contendo algumas informações essenciais.
RF03	Cadastro de Turma	O endpoint de cadastro de turmas permite que professores criem novas turmas na plataforma. Para isso é necessário que o professor esteja cadastrado e autenticado no sistema. O professor deve enviar um JSON contendo algumas informações essenciais.
RF04	Cadastro de Simulação para Turma	O endpoint de cadastro de simulações para turma permite que professores registrem novas simulações para suas turmas na plataforma. Para isso é necessário que o professor esteja cadastrado e autenticado no sistema. O professor deve enviar um JSON contendo algumas informações essenciais.

Quanto às funcionalidades principais, a plataforma foi projetada para permitir o cadastro e a edição de professores, alunos e turmas. Além disso, foi incluído o recurso de cadastro de simulações, tanto no contexto de uma turma quanto individualmente para um aluno. Outro requisito significativo foi o controle de autenticação e permissões, assegurando que diferentes perfis de usuário tivessem acessos restritos de acordo com suas funções na plataforma.

4.2. Desenvolvimento

O padrão REST (*Representational State Transfer*) foi aplicado na plataforma utilizando o Django REST Framework, que oferece uma estrutura robusta e flexível para a criação de APIs. Seguindo esse modelo arquitetural, as *models* atuaram como a base de dados, encapsulando as estruturas principais e relações entre os dados da plataforma, como professores, alunos, turmas e simulações.

As *views*, por sua vez, foram configuradas para lidar com as requisições HTTP, permitindo operações como criação, leitura, atualização e exclusão (CRUD).

Os *serializers* desempenharam um papel essencial na conversão dos objetos *queryset* para formatos como JSON, tornando possível a comunicação eficaz entre a API REST e os consumidores dela. Com a utilização do Django REST Framework, foi possível personalizar os *serializers* para incluir campos adicionais ou lógica específica, garantindo que os dados trafegados fossem claros e seguros.

Com essa estrutura, a API REST assegurou a separação de preocupações e a escalabilidade do projeto, alinhando-se aos objetivos de estabilidade e funcionalidade definidos nos requisitos da plataforma.

O PostgreSQL foi escolhido por várias razões que o tornam a melhor opção para atender aos requisitos da plataforma.

Primeiramente, como um banco de dados relacional, o PostgreSQL permite a definição clara de relações entre as diversas entidades do sistema, como professores, alunos, turmas e simulações. Essa característica é essencial para garantir a integridade dos dados e evitar inconsistências, especialmente em um contexto educacional onde a precisão das informações é indispensável. Ele também suporta transações ACID (Atomicidade, Consistência, Isolamento e Durabilidade), garantindo que as operações sejam realizadas de forma segura e sem perda de dados, mesmo em situações de falha.

A escolha de um banco de dados relacional está alinhada aos princípios estruturados da plataforma e à necessidade de manter um modelo de dados organizado e altamente confiável. Essa abordagem facilita o desenvolvimento contínuo, a manutenção e a futura expansão das funcionalidades da plataforma, assegurando sua estabilidade e longevidade.

Uma decisão central no desenvolvimento da plataforma foi a utilização de containers Docker para o ambiente da aplicação. Essa escolha foi motivada por diversos fatores, sendo o principal a necessidade de garantir consistência entre os diferentes ambientes de desenvolvimento, teste e produção.

O Docker permite encapsular a aplicação e suas dependências em um único ambiente virtualizado, eliminando os problemas decorrentes de discrepâncias no sistema operacional ou nas versões de bibliotecas e ferramentas utilizadas. Além disso, a portabilidade dos containers torna o processo de *deployment* mais ágil e eficiente, reduzindo o tempo e os recursos necessários para configurar novos ambientes.

Outro benefício significativo é a escalabilidade. A arquitetura baseada em containers facilita a replicação de instâncias da aplicação, permitindo que o sistema se adapte rapidamente a variações na demanda. Essa abordagem é particularmente relevante em plataformas educacionais, onde o número de acessos pode variar consideravelmente durante o período letivo.

Por fim, o uso de Docker também promove uma maior segurança, isolando o ambiente da aplicação do sistema operacional principal e minimizando os riscos associados a conflitos ou vulnerabilidades externas. Essa decisão estratégica reforça o compromisso com a estabilidade e a confiabilidade da plataforma.

4.3. Configuração do projeto (manual de execução do projeto)

4.3.1. Rodando o Projeto com Docker

Para garantir a execução consistente do projeto OSLive em diferentes ambientes, como desenvolvimento, homologação e produção, utilizamos Docker e Docker Compose. Esses arquivos encapsulam a aplicação e suas dependências em contêineres isolados. Antes de iniciar, é necessário ter o Docker e o Docker Compose instalados. O Dockerfile localizado em `./oslive/Dockerfile` define a construção da imagem Docker da API OSLive, utilizando a imagem oficial do Python 3.12.2, definindo o diretório de trabalho, copiando os arquivos do projeto, atualizando pacotes do sistema e instalando dependências. Além disso, a API roda na porta 8080. Já o `docker-compose.yml` gerencia a execução da API e do banco de dados, com um serviço de PostgreSQL configurado para manter persistência de dados e permitir conexões externas. A API OSLive depende do banco de dados, define variáveis de ambiente e mapeia diretórios para facilitar o desenvolvimento, permitindo que alterações locais sejam refletidas automaticamente.

Para executar a aplicação, basta abrir um terminal na pasta raiz do projeto (`./oslive/`) e executar `docker-compose up -d`, que inicia os contêineres em segundo plano. Caso queira visualizar os logs, o comando pode ser rodado sem `-d`. A API pode ser acessada pelo navegador em `http://localhost:8080`, e o banco de dados pode ser conectado por ferramentas como DBeaver ou DataGrip. Para interromper a aplicação, basta rodar `docker-compose down`, garantindo que os dados do banco permaneçam preservados devido ao volume mapeado. Caso seja necessário remover também os dados do banco, o comando `docker-compose down -v` pode ser utilizado.

4.3.2. Estrutura do Projeto: A Pasta setup

O projeto OSLive segue a estrutura padrão do Django, com a pasta `./oslive/setup/` sendo um dos diretórios centrais da configuração e roteamento da aplicação web. Dentro dessa pasta, o arquivo `settings.py` é responsável pela configuração geral do projeto, definindo parâmetros como a chave secreta (`SECRET_KEY`), o modo de depuração (`DEBUG`), os hosts permitidos (`ALLOWED_HOSTS`), as aplicações instaladas (`INSTALLED_APPS`) e o banco de dados (`DATABASES`), que está configurado para utilizar PostgreSQL. Além disso, ele inclui configurações para autenticação via `JWTAuthentication`, validação de senhas e controle de requisições de diferentes origens através do `CORS_ALLOWED_ORIGINS`.

O arquivo `urls.py`, por sua vez, atua como o "mapa" de rotas do projeto, direcionando requisições para as views apropriadas. Ele importa as views necessárias, configura a documentação interativa da API via Swagger (`drf_yasg`) e define os padrões de URLs, incluindo as rotas para autenticação, administração e acesso à API. Outros arquivos importantes nessa estrutura incluem `wsgi.py`, que serve como ponto de entrada para servidores compatíveis com WSGI, e `asgi.py`, que permite a execução assíncrona da aplicação com suporte a funcionalidades como WebSockets.

Em resumo, a pasta `./oslive/setup/` contém os arquivos essenciais para a configuração global e o roteamento da aplicação, garantindo que o OSLive funcione corretamente e processe requisições conforme necessário.

4.3.3. Estrutura do Projeto: A Aplicação oslive

No projeto Django, a pasta `./oslive/oslive/` abriga a aplicação principal chamada Oslive, onde são definidos os modelos de dados, a forma como esses dados são representados na API e como a API responde às requisições.

O arquivo `models.py` contém a estrutura do banco de dados usando o ORM do Django. Nele, cada classe representa uma tabela e seus atributos correspondem às colunas. Entre os modelos definidos, há Professor, que possui uma relação com o modelo padrão de usuário do Django e um método para validar e-mails, Aluno, que mantém uma relação muitos-para-muitos com Turma, além de Turma, SimulaçãoTurma, SimulaçãoAluno e seus respectivos processos, cada um estruturado para manter a integridade das relações na aplicação.

A pasta `serializers/` é responsável por converter objetos Python em JSON e vice-versa, garantindo validação dos dados e facilitando a comunicação entre a API e os clientes. Cada serializer cuida da manipulação específica de um modelo, como `AlunoSerializer.py`, que inclui validação de senha e tratamento de dados aninhados, `ProfessorSerializer.py`, `TurmaSerializer.py`, e os serializers para simulações de aluno e turma.

A lógica de negócio é tratada na pasta `view/`, onde os *ViewSets* do Django REST Framework processam requisições HTTP e interagem com os modelos e serializers para responder de forma adequada. Arquivos como `aluno.py`, `professor.py`, `turma.py` e os de simulação gerenciam operações CRUD, implementam validações específicas e lidam com permissões para garantir a segurança e controle de acesso.

A pasta `migrations/` gerencia a evolução do esquema do banco de dados por meio do sistema de migrações do Django. Sempre que há alterações nos modelos, comandos como `python manage.py makemigrations` oslive e `python manage.py migrate` são usados para registrar e aplicar mudanças estruturais na base de dados.

Por fim, arquivos adicionais como `apps.py` configuram a aplicação, `tests.py` contém testes automatizados, e `wsgi.py` e `asgi.py` definem pontos de entrada para servidores web síncronos e assíncronos, respectivamente. Essa estrutura modular e bem-organizada garante a separação clara das responsabilidades dentro do projeto, facilitando a manutenção e a escalabilidade da API do OSLive.

4.3.4. Testes de utilização da API

Para assegurar o correto funcionamento do sistema e validar as funcionalidades da API, realizei uma série de testes diretamente no computador utilizado durante o desenvolvimento do projeto. Com a API em execução na própria IDE, utilizei o Postman como ferramenta para enviar requisições aos endpoints e observar as respostas devolvidas pelo sistema. Essa abordagem permitiu simular cenários reais de utilização, facilitando a identificação de possíveis falhas e a verificação da conformidade dos fluxos implementados. Os testes realizados foram os testes de "Cadastro de Professores", "Cadastro de Alunos", Cadastro de Turmas" e "Cadastro de Simulações para Turmas".

O cadastro de professores é uma funcionalidade que assegura a organização e o controle de acesso ao sistema. Para garantir a segurança e a integridade dos dados, o processo é desenvolvido com base em validações e requer o fornecimento de informações específicas por parte do usuário.

Pré-requisitos: Para se cadastrar no OsLive é necessário ter um e-mail institucional.

1. O usuário deve montar um json com os seguintes dados
    a. "password" **(obrigatório)**
    b. "confirm_password" **(obrigatório)**
    c. "username" **(obrigatório, único)**
    d. "first_name" **(obrigatório)**
    e. "last_name"
    f. "email" **(obrigatório, único)**
    g. "inscrição" **(obrigatório, único)**
2. Enviar esse json para a aplicação
3. A aplicação faz uma validação desses dados. É feita uma simples validação no e-mail para checar se ele é um e-mail institucional da ulbra. Em caso de exceção na validação o usuário recebe uma mensagem
4. Aplicação salva os dados na base de dados
5. Usuário recebe mensagem de confirmação
6. Fim do caso de uso

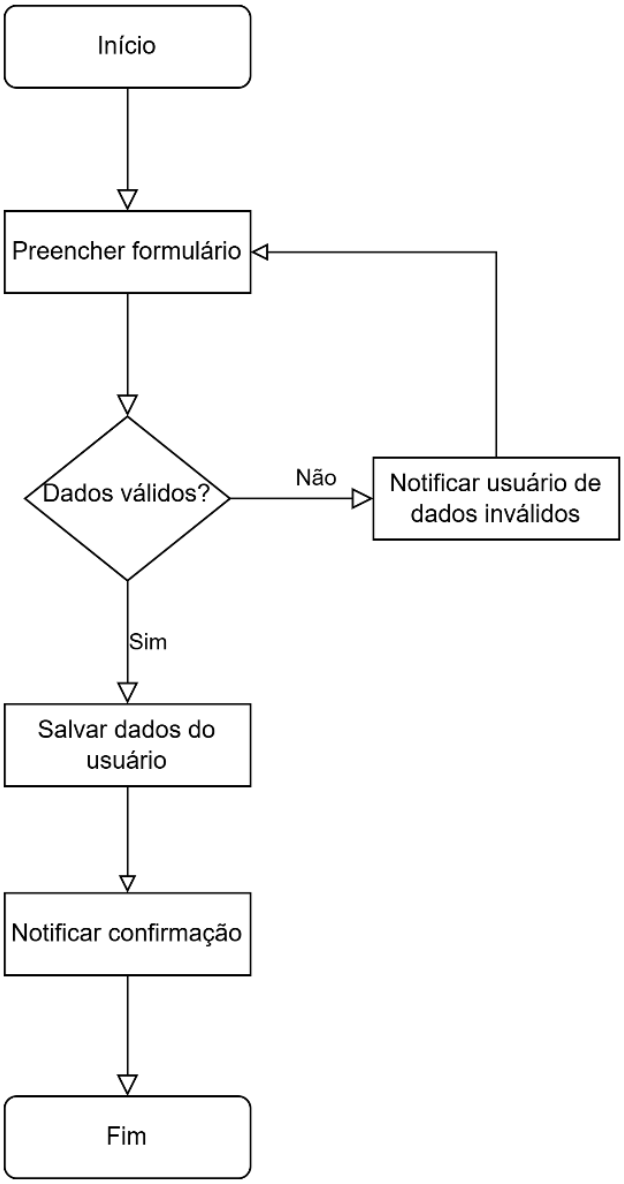

Figura 4. Fluxo do cadastro de professor

A figura acima é um fluxograma que representa de forma visual como é realizado o cadastro de um professor no OSLive.

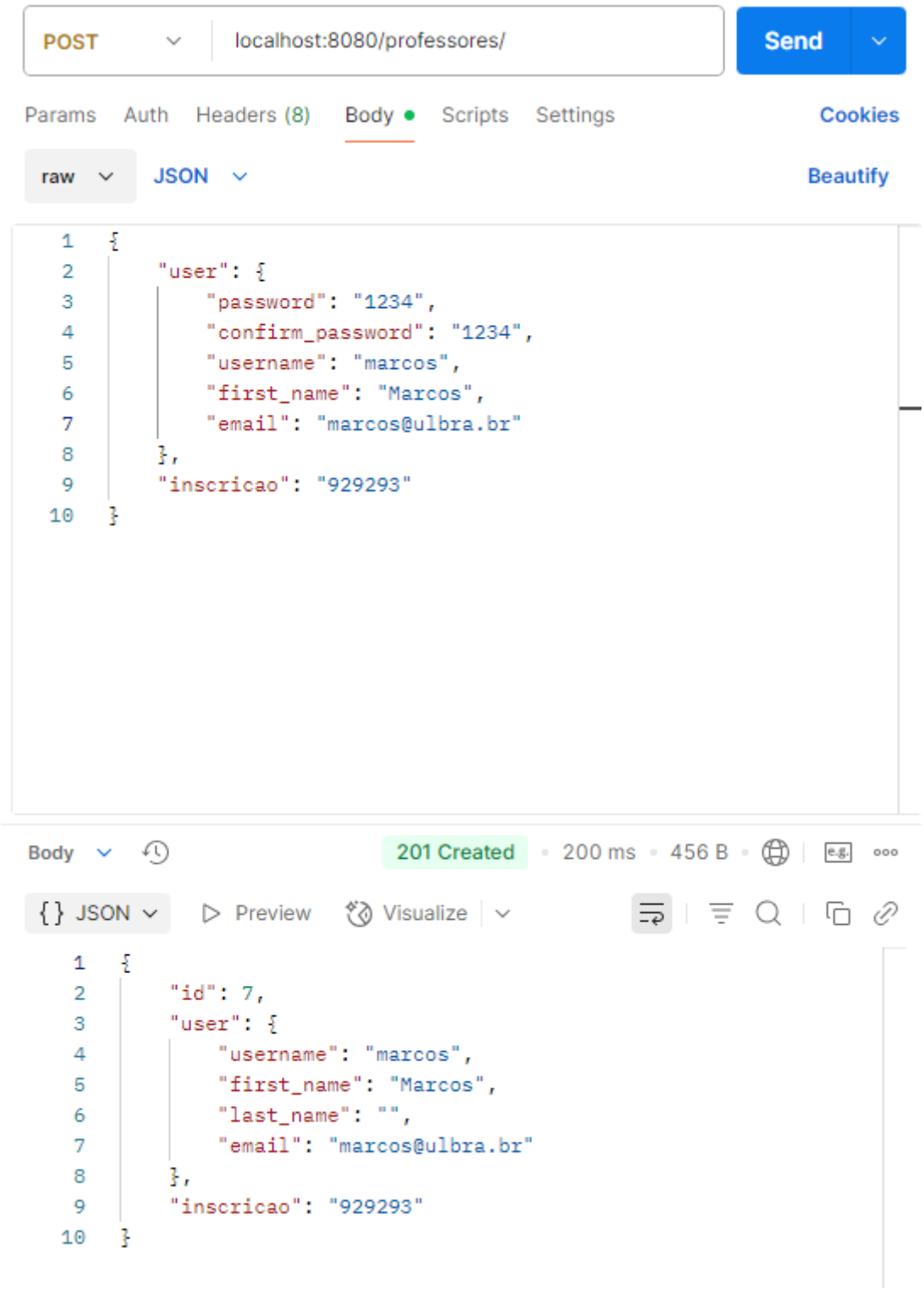

Figura 5. Teste de cadastro de professor realizado com Postman

A figura 5 mostra a estrutura do json necessária que é enviada para a API para cadastrar um professor e mostra também a estrutura do json que é retornada quando o cadastro é efetuado com sucesso.

O cadastro de alunos no OsLive requer o fornecimento de informações necessárias para validação e armazenamento na base de dados, garantindo um processo seguro e eficiente.

Pré-requisitos: Para se cadastrar no OsLive é necessário ter um e-mail institucional.

1. O usuário deve montar um json com os seguintes dados
    a. "password" **(obrigatório)**
    b. "confirm_password" **(obrigatório)**
    c. "username" **(obrigatório, único)**
    d. "first_name" **(obrigatório)**
    e. "last_name"
    f. "email" **(obrigatório, único)**
    g. "matricula" **(obrigatório, único)**
2. Enviar esse json para a aplicação
3. A aplicação faz uma validação desses dados. É feita uma simples validação no e-mail para checar se ele é um e-mail institucional da ulbra. Em caso de exceção na validação o usuário recebe uma mensagem
4. Aplicação salva os dados na base de dados
5. Usuário recebe mensagem de confirmação
6. Fim do caso de uso

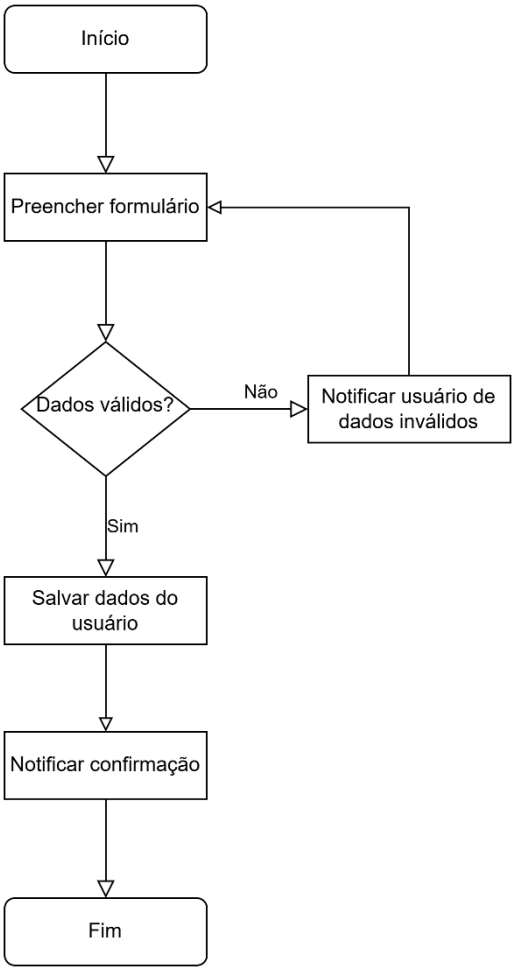

Figura 6. Fluxo do cadastro de aluno

A figura acima é um fluxograma que representa de forma visual como é realizado o cadastro de um professor no OSLive.

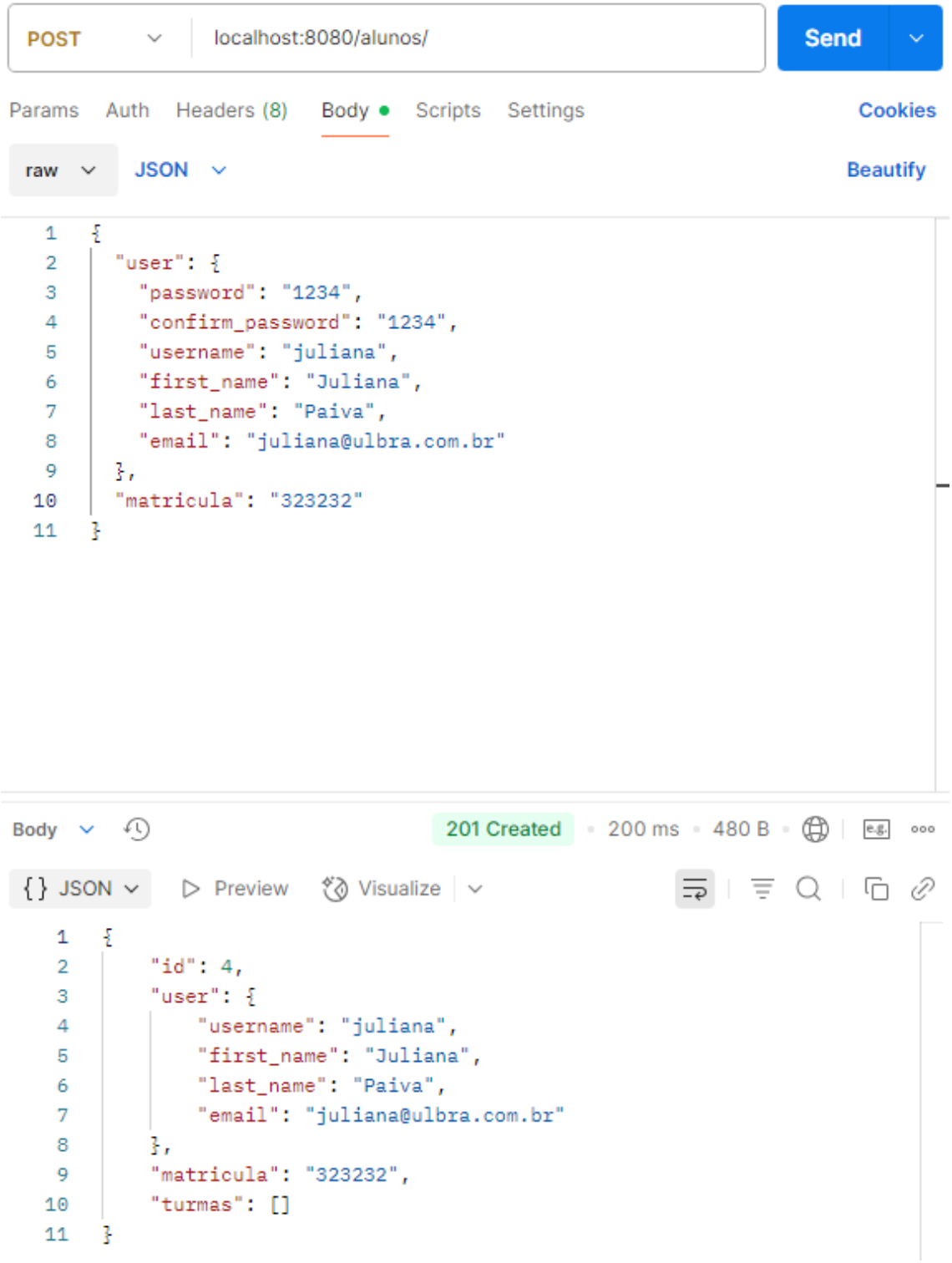

Figura 7. Teste de cadastro de aluno realizado com postman

A figura 7 mostra a estrutura do json necessária para ser enviada para a API para cadastrar um aluno e mostra também a estrutura do json que é retornada quando o cadastro é efetuado com sucesso.

O cadastro de turmas no OsLive é uma funcionalidade essencial para organizar informações como turno, nome, ano e semestre e alunos da turma.

Pré-requisitos: Ter o cadastro de um usuário do tipo professor no OsLive

1. Logar no sistema
2. O usuário deve montar um json com os seguintes dados:
    a. "turno" **(obrigatório)**
    b. "nome" **(obrigatório)**
    c. "ano" **(obrigatório)**
    d. "semestre" **(obrigatório)**
3. Enviar esse json para a aplicação
4. A aplicação salva os dados na base de dados
5. Usuário recebe mensagem de confirmação
6. Fim do caso de uso

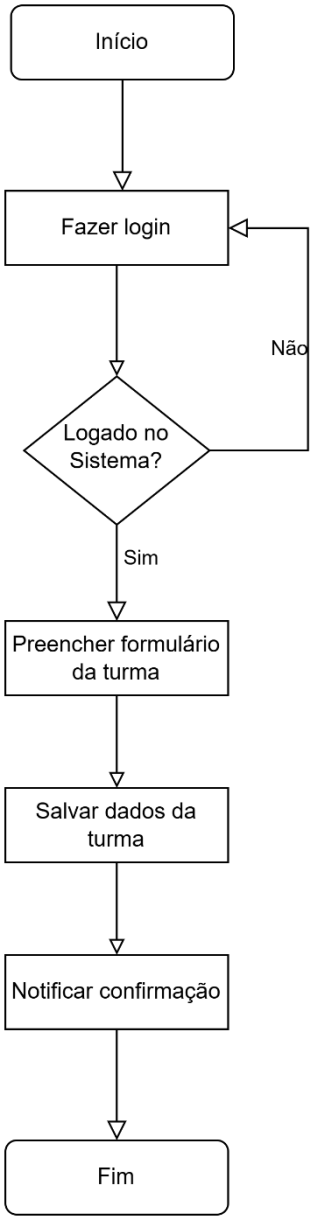

Figura 8. Fluxo do cadastro de turmas

A figura acima é um fluxograma que representa de forma visual como é realizado o cadastro de uma turma no OSLive.

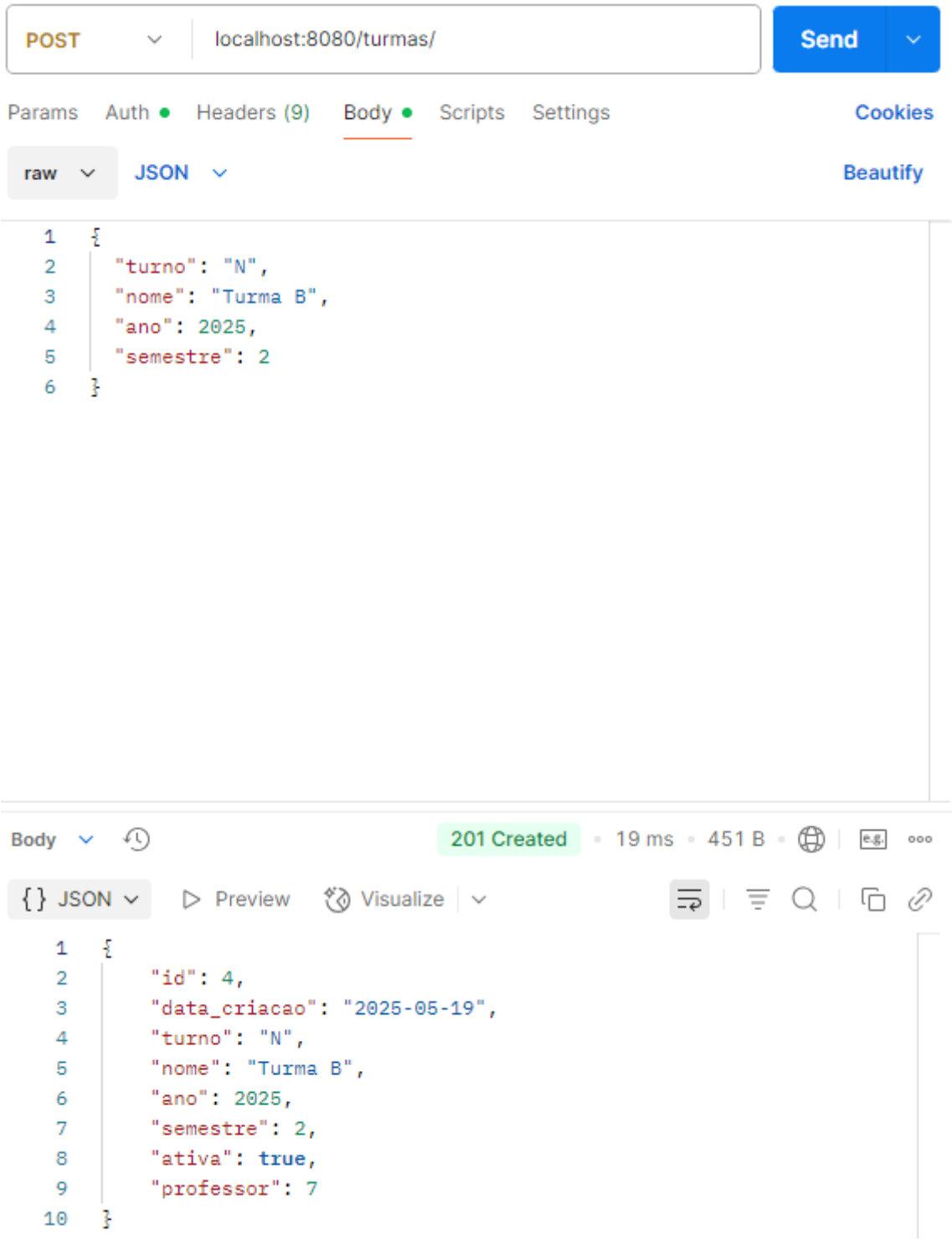

Figura 9. Teste de cadastro de turma realizado com o postman

A figura 9 mostra a estrutura do json necessária que é enviada para a API para cadastrar uma turma e mostra também a estrutura do json que é retornada quando o cadastro é efetuado com sucesso.

O cadastro de simulações para turmas no OsLive é uma funcionalidade essencial que permite a inserção de dados e a validação de algoritmos em um contexto educacional. Professores podem montar um JSON contendo informações como nome, algoritmo, turma e processos específicos (nome, tempo de chegada, tempo de execução), e enviá-lo para aplicação. O sistema verifica a existência da turma, além de validar a vinculação do professor à turma correspondente. Após essa validação, os dados são armazenados na base de dados.

Pré-requisitos: Ter o cadastro de um usuário do tipo professor no OsLive

1. Logar no sistema
2. Usuário deve montar um json com os seguintes dados:
   a. "nome" **(obrigatório)**
   b. "algoritmo" **(obrigatório)**
   c. "turma" **(obrigatório)**
   d. "processos" {"nome", "tempo_chegada", "tempo_execucaçao"} **(obrigatório)**
3. Enviar esse json para a aplicação
4. A aplicação faz uma validação desses dados. Ela vai validar se o id da turma existe e se o professor que está cadastrando está vinculado com essa turma
5. A aplicação salva os dados na base de dados
6. O usuário recebe uma mensagem de confirmação
7. Fim do caso de uso

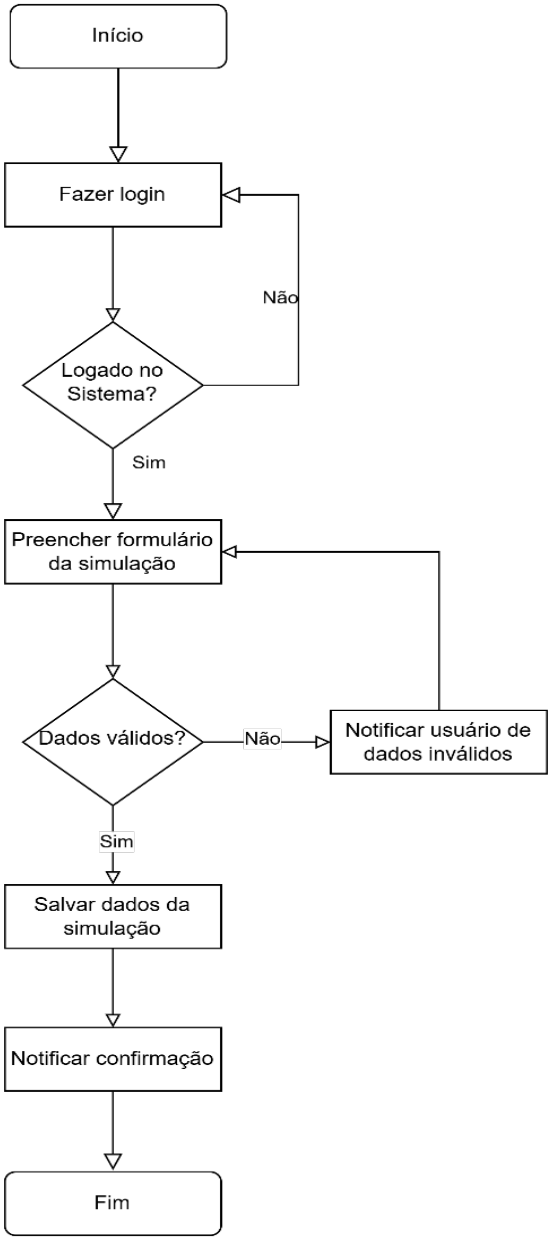

Figura 10. Fluxo do cadastro da simulação para uma turma

A figura acima é um fluxograma que representa de forma visual como é realizado o cadastro de uma simulação para uma turma no OSLive.

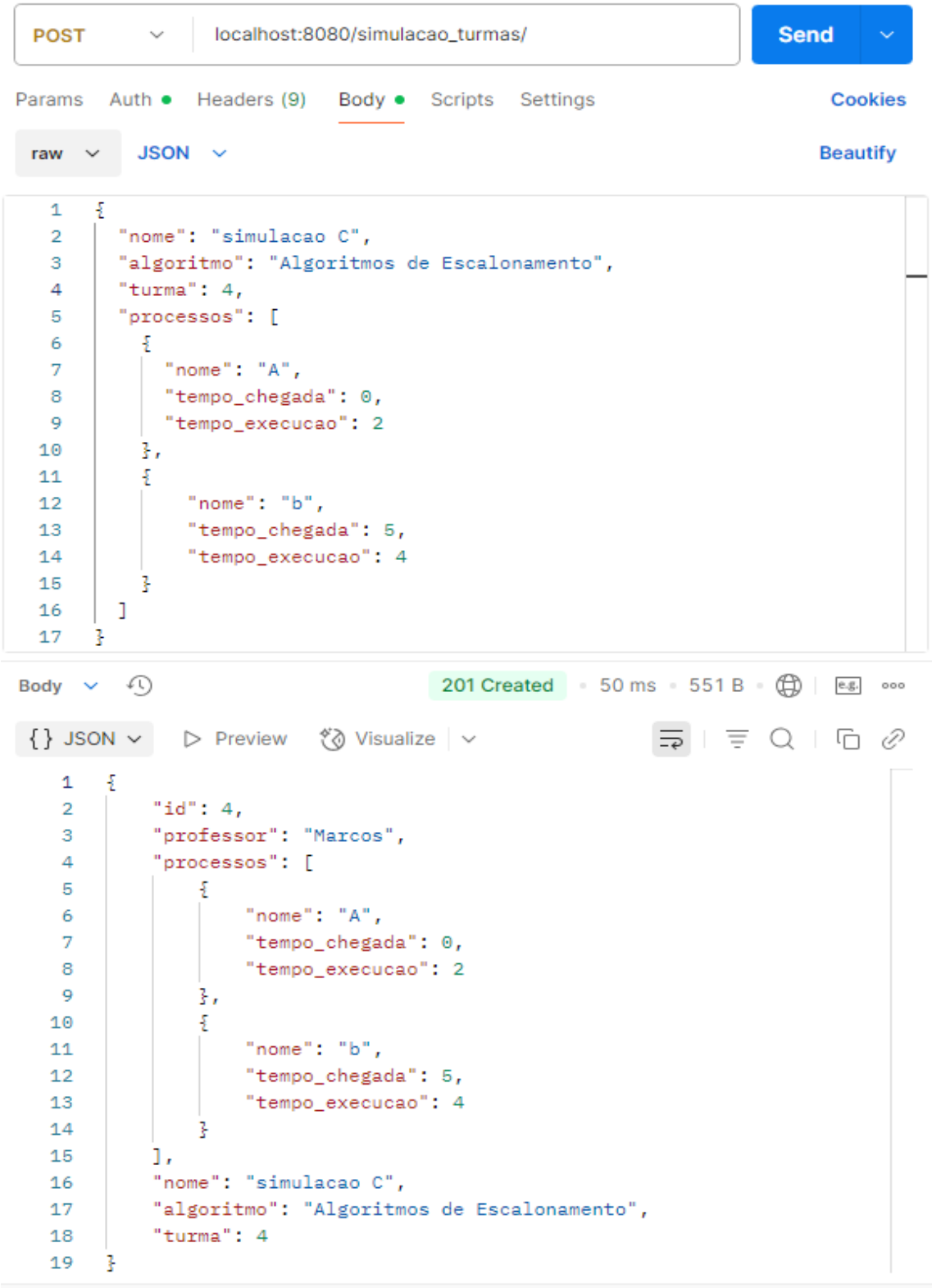

Figura 11. Teste de cadastro de simulação para uma turma

A figura 11 mostra a estrutura do json necessária que é enviada para a API para cadastrar uma simulação para turma e mostra também a estrutura do json que é retornada quando o cadastro é efetuado com sucesso.

5. Considerações Finais

Ao longo deste trabalho, buscou-se sobretudo atingir o objetivo principal de desenvolver e validar um modelo de cadastro de simulações do OSLive. Os resultados obtidos demonstram que o objetivo foi ~~plenamente~~ alcançado: a solução proposta mostrou-se eficaz, promovendo não apenas a correta vinculação entre professores, turmas alunos e simulações de algoritmos, mas também assegurando a integridade e a consistência dos dados durante todo o processo.

Um dos principais impactos observados refere-se à melhoria significativa na manutenção do sistema. A estrutura modular adotada, aliada à validação rigorosa dos dados por meio da utilização de JSON e à persistência eficiente em banco de dados, facilitou futuras modificações e reduziu a incidência de erros, contribuindo para um ambiente mais robusto e sustentável. Tais avanços promovem agilidade na resolução de problemas e ampliam a escalabilidade do sistema.

Como trabalhos futuros, destaca-se a possibilidade de expandir as funcionalidades já implementadas, incluindo novos tipos de simulações e integração com plataformas externas. Além disso, recomenda-se a realização de testes em larga escala, envolvendo diferentes contextos educacionais, a fim de validar a adaptabilidade e a eficácia do modelo em cenários diversos. Essas iniciativas poderão não apenas consolidar os resultados alcançados neste estudo, mas também abrir novas frentes de pesquisa e desenvolvimento no âmbito do ensino de Sistemas Operacionais e tecnologias educacionais.

Referências bibliográficas

Bailis, P., Hellerstein, J. M., Stonebraker, M. Readings in Database Systems. (2015)

Borghesan, L., Marioti, M. B., Filipakis, C., & Fagundes, F. (2021). OSLIVE: simulação do algoritmo de múltiplas filas do módulo de escalonamento de processos.

Buendía, F., & Cano, J. C. (2006). Webgene $ _ {\rm OS} $: A Generative and Web-Based Learning Architecture to Teach Operating Systems in Undergraduate Courses. IEEE Transactions on Education, 49(4), 464-473.

Christopher, W. A., Procter, S. J., & Anderson, T. E. (1993, January). The Nachos instructional operating system. In USENIX Winter (pp. 481-488).

Codd, E. F. (1970). A relational model of data for large shared data banks. Communications of the ACM, 377-387

Dantas, V. F., Garcia, F. P. (2003). WIDE WORK WEB – Uma Metodologia para o Desenvolvimento de Aplicações Web num Cenário Global

Defining Software Architecture. Pittsburgh: Carnegie Mellon University, 2009.

Figueiredo, R. V., & Marioti, M. B. (2021).. OSLIVE: Módulo Web de Simulação da Paginação por Demanda.

Fowler, Martin. Patterns of Enterprise Application Architecture. Boston: Addison-Wesley, 2003

Fowler, Martin. Is Design Dead? 2004

da Silva, G. F., Marioti, M. B., & Araújo, F. C. (2020). OSLIVE: Módulo de Exercícios de Escalonamento de Processos. In 22º CONGRESSO DE COMPUTAÇÃO E TECNOLOGIAS DA INFORMAÇÃO (p. 93).

Hughes, C. E., & Pfleeger, C. P. (1978). ASSIST-V: An Environment Simulator for IBM 360 Systems Software Development. IEEE Transactions on Software Engineering, (6), 526-530.

Jong, B. S., Lai, C. H., Hsia, Y. T., Lin, T. W., & Lu, C. Y. (2012). Using game-based cooperative learning to improve learning motivation: A study of online game use in an operating systems course. *IEEE Transactions on Education*, *56*(2), 183-190.

Lincke, S. J. (2005, October). Creating interest in operating systems via active learning. In *Proceedings Frontiers in Education 35th Annual Conference* (pp. S3C-7). IEEE.

Pamplona, S., Medinilla, N., & Flores, P. (2018). A systematic map for improving teaching and learning in undergraduate operating systems courses. IEEE Access, 6, 60974-60992.

Perez da Villa, A. 1995. “OS–bridge between academia and reality”. ACM SIGCSE Bulletin,27(1):146–148

Santos, A. M., Sousa, C. H., Noleto, F. S., Fagundes, F., & Mairioti, M. B. (2017). OSLive: Protótipo de simulação de algoritmos de escalonamento de processos. In ENCOINFO-Congresso de Computação e Tecnologias da Informação (pp. 80-87). ENCOINFO.

Tanenbaum, A. S. (1987). A UNIX clone with source code for operating systems courses. ACM SIGOPS Operating Systems Review, 21(1), 20-29.

Tanenbaum, A. (2009). Modern operating systems. Pearson Education, Inc.

Webb, K. C., & Taylor, C. (2014, March). Developing a pre-and post-course concept inventory to gauge operating systems learning. In Proceedings of the 45th ACM technical symposium on Computer science education (pp. 103-108).

Zareie, F., & Najaf-Zadeh, M. (2013). OSLab: A Hand-on Educational Software for Operating Systems Concepts Teaching and Learning. WebPub, 1(2), 31-37.

Ziegler, U. (1999, March). Discovery learning in introductory operating system courses. In *The proceedings of the thirtieth SIGCSE technical symposium on Computer science education* (pp. 321-325).

The PostgreSQL Global Development Group. Postgresaql.org, 2025. Disponível em: https://www.postgresql.org/docs/current/intro-whatis.html