INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
CAMPUS CAJAZEIRAS
CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

NOBULLYING API: UMA *API REST* QUE AUXILIE A PREVENÇÃO E COMBATE AO *BULLYING* E ASSÉDIO NO ÂMBITO ACADÊMICO DO IFPB - CAMPUS CAJAZEIRAS

MATHEUS NUNES MIGUEL

Cajazeiras
2023

MATHEUS NUNES MIGUEL

NOBULLYING API: UMA *API REST* QUE AUXILIE A PREVENÇÃO E COMBATE AO *BULLYING* E ASSÉDIO NO ÂMBITO ACADÊMICO DO IFPB - CAMPUS CAJAZEIRAS

Trabalho de Conclusão de Curso apresentado junto ao Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba - Campus Cajazeiras, como requisito à obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador

Prof. Me. Fábio Abrantes Diniz.

Cajazeiras
2023

IFPB / Campus Cajazeiras
Coordenação de Biblioteca
Biblioteca Prof. Ribamar da Silva
Catalogação na fonte: Cícero Luciano Félix CRB-15/750

M636n Miguel, Matheus Nunes.

Nobullying api: uma *API REST* que auxilie a prevenção e combate ao bullying e assédio no âmbito acadêmico do IFPB-campus Cajazeiras / Matheus Nunes Miguel.– 2023.

64f. : il.

Trabalho de Conclusão de Curso (Tecnólogo em Análise e Desenvolvimento de Sistemas) - Instituto Federal de Educação, Ciência e Tecnologia da Paraíba, Cajazeiras, 2023.

Orientador(a): Prof. Me. Fabio Abrantes Diniz.

1. Desenvolvimento de sistemas. 2. *API REST Nobullyng*. 3. Combate ao assédio. 4. Combate ao *bullyng*. I. Instituto Federal de Educação, Ciência e Tecnologia da Paraíba. II. Título.

IFPB/CZ CDU: 004.4(043.2)

MINISTÉRIO DA EDUCAÇÃO
SECRETARIA DE EDUCAÇÃO PROFISSIONAL E TECNOLÓGICA
INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA

MATHEUS NUNES MIGUEL

NOBULLYING API: UMA API REST QUE AUXILIE A PREVENÇÃO E COMBATE AO BULLYING E ASSÉDIO NO ÂMBITO ACADÊMICO DO IFPB - CAMPUS CAJAZEIRAS

Trabalho de Conclusão de Curso apresentado junto ao Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia da Paraíba - Campus Cajazeiras, como requisito à obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador

Prof. Me. Fábio Abrantes Diniz.

Aprovada em: **26 de Fevereiro de 2024.**

Prof. Me. Fábio Abrantes Diniz - Orientador

Prof. Esp. Cristiano Alves Fontes - Avaliador

IFPB - Campus Cajazeiras

Prof. Esp. Afonso Serafim Jacinto - Avaliador

IFPB - Campus Cajazeiras

Documento assinado eletronicamente por:

- **Fabio Abrantes Diniz**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 06/03/2024 15:44:24.
- **Afonso Serafim Jacinto**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 06/03/2024 17:08:26.
- **Cristiano Alves Fontes**, **PROF ENS BAS TEC TECNOLOGICO-SUBSTITUTO**, em 06/03/2024 18:08:57.

Este documento foi emitido pelo SUAP em 06/03/2024. Para comprovar sua autenticidade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/autenticar-documento/ e forneça os dados abaixo:

Código 542905
Verificador: a90114c92f
Código de Autenticação:

Este trabalho dedico a Deus, pois sem Ele eu não conseguiria ter a capacidade necessária para desenvolvê-lo. Este trabalho dedico a todos aqueles a quem este projeto possa ajudar de alguma forma.

AGRADECIMENTOS

Em primeiro lugar, a Deus, que fez com que meus objetivos fossem alcançados, durante todos os meus anos de estudos.

Aos meus pais, que me incentivaram nos momentos difíceis e compreenderam a minha ausência enquanto eu me dedicava à realização deste trabalho.

Ao professor Me. Fábio Abrantes Diniz, por ter sido meu orientador e ter desempenhado tal função com dedicação e amizade.

Aos amigos, que sempre estiveram ao meu lado, pela amizade incondicional e pelo apoio demonstrado ao longo de todo o período de tempo em que me dediquei a este trabalho.

”Educai as crianças para que não seja necessário punir os adultos.”

Atribuído a Pitágoras

RESUMO

No Brasil, a maioria das Instituições de Ensino não oferecem suporte prático ao combate de *bullying* e assédio. O *bullying* e o assédio podem trazer problemas significativos para a saúde mental e emocional dos discentes de uma Instituição de Ensino. Os discentes que são alvo de *bullying* ou assédio podem vivenciar ansiedade, depressão, baixa autoestima e isolamento social. O uso da Tecnologia da Informação pode ser essencial na prevenção do *bullying* nessas instituições. É primordial a disponibilização de uma *Application Programming Interface* (*API*) que forneça serviços seguros e eficazes no gerenciamento do acompanhamento de casos de *bullying* para as Instituições de Ensino criarem suas aplicações de combate ao *bullying* e assédio. Portanto, o presente trabalho desenvolveu uma *API* que oferece serviços que auxiliam o gerenciamento e a criação de denúncias sobre casos de *bullying* e assédios recebidos pelo Núcleo de Combate ao Assédio (NUCA) do IFPB - Campus Cajazeiras. Proporcionando um ambiente escolar seguro e respeitoso para todos os estudantes.

Palavras-chave: API. NUCA. IFPB. assédio. denúncias.

ABSTRACT

In Brazil, most Educational Institutions do not offer practical support to combat of bullying and harassment. Bullying and harassment can cause significant problems for the mental and emotional health of students at an Educational Institution. The students who are targets of bullying or harassment may experience anxiety, depression, low self-esteem and social isolation. The use of Information Technology can be essential in preventing bullying in these institutions. It is essential to provide a Application Programming Interface (API) that provides secure and effective services in managing the monitoring of bullying cases for Institutions of I teach them to create their applications to combat bullying and harassment. Therefore, the present work developed an API that offers services that help manage and the creation of reports on cases of bullying and harassment received by the Center of Combating Harassment (NUCA) of IFPB - Cajazeiras Campus. Providing a safe and respectful school environment for all students.

Keywords: API. NUCA. IFPB. harassment. complaints.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE CÓDIGOS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
CRUD	*Create, Read, Update, Delete*
DTO	*Data Transfer Object*
HTTP	*HyperText Transfer Protocol*
ID	*Identification*
IDE	*Integrated Development Environment*
IEs	*Instituições de Ensino*
IFPB	*Instituto Federal de Educação, Ciência e Tecnologia da Paraíba*
JPA	*Java Persistence API*
JSON	*Javascript Object Notation*
JWT	*JSON Web Token*
NUCA	*Núcleo de Combate ao Assédio*
PDF	*Portable Document Format*
REST	*Representational State Transfer*
SQL	*Structured Query Language*
TI	*Tecnologia da Informação*
URL	*Uniform Resource Locator*
XML	*Extensible Markup Language*
UML	*Unified Modeling Language*
SUAP	*Sistema Unificado de Administração Pública*

SUMÁRIO

1 INTRODUÇÃO

Em 2007 houve um grande aumento de *bullying* no território Brasileiro, chegando a taxa de 45% (FANTE; PEDRA, 2008). Segundo os autores, as principais motivações para esse crescimento no ano de 2007 foram o estímulo à competitividade e ao individualismo, a banalização da violência e a certeza da impunidade, o desrespeito e a desvalorização do ser humano.

Dentre os tipos de *bullying* sofridos pelos discentes estão: o *bullying* psicológico, onde ocorrem ações como espalhar fofocas, excluir de atividades, xingar, ameaçar e ridicularizar, o qual predominou nas situações. 23,3% dos estudantes relataram sofrerem esse tipo de violência no ambiente escolar. Enquanto o *bullying* físico, envolvendo ações como dar tapas, socos, chutes e empurrar, e o *bullying* virtual, que inclui enviar mensagens por via telefone ou internet de ameaça, xingamento, ridicularização e ofensa alcançaram 15% e 5,5% dos discentes, respectivamente. (MARCOLINO et al., 2018).

Sobre o caso que resultou no massacre de Suzano, nas histórias repetidas pelas redes sociais, os atiradores haviam se revoltado após anos de *bullying* e falta de traquejo social, vingando-se dos populares e trazendo atenção para si, invertendo papéis de poder (BRAGA, 2022). Outras formas de *bullying* são praticadas frequentemente nas escolas brasileiras, como evidenciado pelos seguintes exemplos de casos: recentemente, em 31 de março de 2023, na cidade do Rio de Janeiro, uma discente de apenas 11 anos foi vítima de *bullying* por causa de sua deficiência de aprendizado (NORDESTE, 2023). Outro exemplo ocorreu no IFPB campus João Pessoa, em 2019, houve um ocorrido a respeito de *bullying* em que um jovem concluinte do curso Técnico de Eletromecânica teve algumas fotos expostas em um aplicativo de mensagens, no qual passou meses de ansiedade e isolamento. Logo, após se recuperar do acontecimento, o mesmo decidiu iniciar um projeto para ajudar outras pessoas que também foram vítimas de *bullying*. (IFPB, 2019b).

O uso de Tecnologia da Informação (TI) é importante no combate ao *bullying*. Alguns sistemas estão sendo implementados para solucionar esse problemas nas Instituições de Ensino. O trabalho de Maia e Júnior (2022), propõe um sistema intitulado BEKID, com o propósito de auxiliar na identificação de possíveis casos de *bullying*, possibilitando uma relação melhor entre discente e escola. Pois, trata-se de uma aplicação que tem como objetivo prevenir caso de *bullying* no ambiente acadêmico. A proposta de Marcolino et al. (2018) é realizar uma pesquisa, análise e modelagem de

um aplicativo *mobile* que permite os discentes realizarem denúncias de *bullying* para o corpo estudantil, começando por questionários para colher requisitos suficientes para a modelagem e a criação do aplicativo *mobile*.

No IFPB, campus Cajazeiras, o combate ao *bullying* e ao assédio é conduzido pelo Núcleo de Combate ao Assédio (NUCA). O NUCA desempenha um papel fundamental no campus, atuando como o principal ponto de contato para denúncias de *bullying* e assédio por parte dos membros da comunidade acadêmica (IFPB, 2019a). No entanto, para reportar casos de *bullying* e assédio, tanto por parte de discentes quanto de docentes, é necessário realizar uma denúncia manualmente no departamento do NUCA. A falta de um sistema adequado para o tratamento de denúncias de *bullying* e assédio nas instituições de ensino no Brasil frequentemente resulta em um processo manual de denúncias. Isso ocorre devido à falta de suporte prático da maioria das instituições no combate ao *bullying* e assédio, fazendo com que as medidas sejam tomadas somente após a ocorrência da agressão, o que pode ter consequências graves para a vítima.

Portanto, o presente trabalho desenvolveu a *NoBullying API*, uma *API* que auxilia no gerenciamento do acompanhamento nos casos de *bullying* e assédios recebidos pelo NUCA do IFPB campus Cajazeiras. A ideia foi fornecer um serviço que pode ser integrado a diferentes plataformas oriundas do desenvolvimento *web* e *mobile*. Permitindo que as informações sejam recebidas e tratadas de forma anônima, segura e eficaz pelas autoridades responsáveis.

1.1 JUSTIFICATIVA

No Brasil, o fenômeno de *bullying* é presente nas Instituições de Ensino (IEs), esse fenômeno acontece independente de turno de estudo, localização, tamanho e a região em que a IE se encontra (FANTE, 2005). Logo, a criação de um sistema de denúncias de *bullying* anônimas é essencial para garantir que as vítimas e testemunhas se sintam seguras ao relatar as situações de *bullying*. O *bullying* e o assédio causam o silêncio das vítimas com relação a comunicação sobre a violência sofrida, uma vez que as vítimas sentem medo de serem perseguidas com mais intensidade (HIRIGOYEN, 2002).

Como supracitado, o NUCA registra denúncias de *bullying* e assédio no IFPB campus Cajazeiras. Suas denúncias são de forma manual, em que o denunciante deve se dirigir até o departamento do NUCA para registrar sua denúncia. O detalhe da locomoção do denunciante até o departamento pode gerar uma certo receio por parte do denunciante, pois o mesmo pode ser visto por outras pessoas, assim gerando

especulações de uma possível denúncia que pode terminar no conhecimento do agressor.

Logo, com a utilização de uma *API* de denúncias de *bullying* anônimas, é possível facilitar o processo de registro e encaminhamento das informações, agilizando a tomada de providências por parte das autoridades escolares e dos órgãos competentes. O avanço tecnológico e o ambiente inovador têm proporcionado a utilização de ferramentas que podem auxiliar na tomada de decisão, na comunidade escolar (MAIA; JÚNIOR, 2022). Essa agilidade é crucial para evitar a repetição de situações de *bullying* e prevenir danos maiores às vítimas. Com uma ferramenta de denúncias acessível e confiável, a comunidade escolar, incluindo alunos, funcionários e professores, têm a oportunidade de contribuir ativamente no combate ao *bullying*, fortalecendo a cultura de respeito e cuidado mútuo.

Portanto, o presente trabalho apresenta a *NoBullying API*, uma *API* que auxilia o NUCA do IFPB campus Cajazeiras no combate aos casos de *bullying* e assédio nesta Instituição de Ensino. Oferece um canal seguro e eficiente para relatar casos de *bullying*, promovendo ações concretas de prevenção e intervenção. Ao adotar essa abordagem tecnológica, cria-se um ambiente escolar mais seguro, acolhedor e propício ao desenvolvimento saudável de todos os estudantes.

1.2 OBJETIVOS

1.2.1 OBJETIVO GERAL

Este trabalho visa desenvolver uma *API REST*, intitulada *NoBullying API*, contendo serviços que auxiliam no combate e prevenção ao *bullying* e assédios apurados pelo NUCA do IFPB campus Cajazeiras.

1.2.2 OBJETIVOS ESPECÍFICOS

A fim de alcançar o objetivo geral do trabalho, as seguintes metas foram traçadas:

- Identificar como é feito o processo de registro de denúncias e resoluções no IFPB - campus Cajazeiras;
- Identificar as categorização dos tipos de denúncias de *bullying*;
- Realizar estudos sobre as tecnologias do desenvolvimento de *APIs*;

- Escolher uma arquitetura de desenvolvimento de *software* que possibilite a escalabilidade sem a perda na facilidade de manutenção;
- Projetar uma arquitetura que atenda o combate do *bullying*, considerando os requisitos de segurança, usabilidade e integração com diferentes plataformas;
- Realizar a construção de uma documentação detalhada sobre os serviços da *API*;
- Desenvolver uma versão inicial da *API*;

1.3 METODOLOGIA

Com o intuito de resolver o problema de *bullying* e assédio no ambiente acadêmico do IFPB - Campus Cajazeiras, foram realizados um amplo estudo bibliográfico, pesquisas sobre maneiras de combater o bullying e assédio nas IEs, além de várias entrevistas com membros do NUCA, a fim de entender como funciona o fluxo de gerenciamento para a realização de denúncias dentro do IFPB - campus Cajazeiras, pois todas as formas de denúncia que o NUCA recebe são exclusivamente de forma manual e presencial. A partir dos requisitos coletados foi possível definir os objetivos deste trabalho além de realizar a construção do diagrama de casos de uso, ilustrado pela Figura 6.

Através de pesquisas sobre sistemas destinados a auxiliar no processo de denúncias em instituições de ensino ou em outros contextos relacionados a denúncias, foram identificados alguns trabalhos, incluindo sistemas *web* e *mobile*. Dentre eles, destacam-se o sistema “BEKID” desenvolvido por Maia e Júnior (2022) e o sistema “PANDORA” desenvolvido por MARINHO et al. (2018), ambos focados em abordar formas de combater o bullying em ambientes acadêmicos. Essas pesquisas permitiram a comparação e uma compreensão mais aprofundada sobre como sistemas de denúncias funcionam, servindo como base para o desenvolvimento da NoBullying API.

Foram realizados estudos acerca de arquiteturas de *software* para identificar a mais adequada ao desenvolvimento da *NoBullying API*. O critério principal na escolha da arquitetura foi a organização do código em camadas de aplicação, buscando baixo acoplamento, clareza nas responsabilidades do código, além de garantir facilidade de manutenção e escalabilidade. Com base nesses requisitos, a Arquitetura Limpa (*Clean Architecture*) surgiu como a escolha mais apropriada, pois atende integralmente a todos esses critérios.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 *BULLYING*: DEFINIÇÃO E IMPACTO

A palavra *bullying* é derivada do verbo inglês *bully*, que significa usar a superioridade física para intimidar alguém. Também pode ser empregada como adjetivo, no sentido de valentão ou tirano (SCHULTZ et al., 2012). O *bullying*, de acordo com Chalita (2008), é o termo utilizado por educadores para denominar os atos maldosos como apelidos ou qualquer forma de hostilidade com os colegas.

Segundo uma pesquisa feita na rede pública de ensino da região de Florianópolis/SC pela Sociedade Brasileira de Psicologia, relata que as vítimas de *bullying* ficam retraídas, mais quietas e não querem desenvolver trabalhos em equipe (SANTOS et al., 2015). Outra pesquisa realizada no interior do estado de São Paulo, em estabelecimentos de ensino públicos e privados, com um universo de 1.761 alunos, comprovou que 49% dos alunos estavam envolvidos no fenômeno. Desses, 22% figuravam como "vítimas"; 15% como "agressores" e 12% como "vítimas-agressoras" (FANTE, 2002).

Muitas vezes o ambiente escolar se constitui como um local propício para a ocorrências de comportamentos agressivos, sofrimento e medo, assim podendo ocasionar em graves consequências individuais e sociais, principalmente para os jovens envolvidos (SANTOS; KIENEN, 2014). O *bullying* ocorre justamente no lugar em que, em tese, as crianças e adolescentes deveriam aprender a conviver socialmente com respeito ao outro e a exercitar e desenvolver sua individualidade e subjetividade sem coerção (DUQUE, 2007).

As consequências para as "vítimas" desse fenômeno são graves e abrangentes, promovendo no âmbito escolar o desinteresse pela escola, o déficit de concentração e aprendizagem, a queda do rendimento, o absentismo e a evasão escolar. No âmbito da saúde física e emocional, a baixa na resistência imunológica e na autoestima, o stress, os sintomas psicossomáticos, transtornos psicológicos, a depressão e o suicídio (FANTE, 2002).

2.2 COMBATE AO *BULLYING* E ASSÉDIOS

O *bullying* ocorre principalmente nas escolas, mas pode ser identificado precocemente com a observação de alterações de comportamento (ALMEIDA et al., 2008). Essa pronta identificação é crucial para evitar a escalada dos eventos e prevenir con-

sequências mais graves no futuro. Existem vários tipos de sintomas que evidenciam a exposição de adolescentes ao *bullying*, juntamente com vários sintomas de saúde como: ansiedade, depressão, baixa autoestima, autolesão e ideação e comportamento suicida (RUSSO, 2020).

Dentre esses sintomas, os docentes podem identificar sinais comportamentais que algo de errado está acontecendo com aquele aluno específico, tais como: falta de vontade e medo de ir à escola, sentir-se mal ao sair para escola e não querer ir sozinho, baixa auto-estima, ter pesadelos frequentes, perder repetidamente pertences e dinheiro não falar sobre o assunto (ALMEIDA et al., 2008). Ao reconhecer esses sinais, os docentes e demais profissionais da escola têm a oportunidade de agir rapidamente e de forma eficaz para interromper o ciclo de agressões e promover um ambiente escolar seguro.

Propor ações como implementar programas de conscientização, oferecer apoio psicológico aos discentes e incentivar a denúncia de casos de *bullying* possuem resultados momentâneos, mas ainda assim, podem ajudar no combate ao *bullying* (LIMA et al.,). Ao agir rapidamente e de maneira eficaz diante de casos de agressões, é possível criar um ambiente escolar seguro, promovendo o bem-estar dos estudantes e contribuindo para um desenvolvimento saudável em todas as áreas de suas vidas.

A violência de gênero, no que diz respeito ao abuso e assédio sexual é uma forma de violência que persegue as mulheres desde a infancia, em que muitas vezes ocorrem no ambiente escolar, dentro da própria família ou no trabalho (MOREIRA et al., 2016). O assédio pode assumir várias formas, como o assédio moral e o assédio sexual. Segundo Barbieri (2019), existe uma diferença entre assédio moral e *bullying*, geralmente o *bullying* está relacionado às escolas, já o assédio moral está relacionado à cultura organizacional e à hierarquia.

Denunciar essas formas de violência é um passo fundamental para combater e prevenir tais comportamentos prejudiciais. Ao denunciar, as vítimas têm a oportunidade de buscar ajuda, apoio e proteção, além de promover a responsabilização dos agressores. Além disso, as denúncias ajudam a criar um ambiente escolar, de trabalho ou social seguro em que todos são valorizados e respeitados. Segundo Aquino et al. (2012), existem dois níveis para a intervenção de assédio: legislativo e administrativo. A intervenção por meio da legislação ou normas são consideradas pouco satisfatórias. Já as intervenções baseadas em ações administrativas, como a criação para medição e investigação de assédios acompanhados da possibilidade de punição para os assediadores, tem sido um sucesso no combate a na prevenção ao assédio (AQUINO et al., 2012).

2.3 NUCA: NÚCLEO DE COMBATE AO ASSÉDIO DO IFPB

O Núcleo de Combate ao Assédio (NUCA) é um setor presente em cada campus do IFPB, responsável por receber denúncias de discentes e servidores relacionados nos casos de *bullying* e assédio (IFPB, 2021). Sua atuação tem como objetivo principal promover ações preventivas no ambiente acadêmico, visando evitar a ocorrência desses comportamentos indesejados.

O trabalho realizado pelo NUCA, presente em cada campus do IFPB, desempenha um papel fundamental na prevenção e combate ao *bullying* e assédio no ambiente acadêmico (IFPB, 2021). No entanto, se a denúncia não estiver diretamente relacionada a discentes, será encaminhada para outra instância responsável por lidar com esse tipo de ocorrência. Dessa forma, o NUCA concentra seus esforços em tratar especificamente de situações que afetem a comunidade estudantil, garantindo a efetividade do seu papel no combate ao assédio e na promoção de um ambiente acadêmico seguro.

Além de registrar os incidentes relatados, o NUCA oferece suporte e acolhimento às vítimas em diferentes tipos de ocorrência. No caso de uma denúncia sobre assédio sexual, o setor pode encaminhar a vítima para receber apoio psicológico, fornecer orientações sobre seus direitos e auxiliar no processo de realizar uma denúncia formal em instâncias externas, como uma delegacia de polícia (IFPB, 2021).

2.3.1 GERENCIAMENTO DE DENÚNCIAS DE *BULLYING* E ASSÉDIO PELO NUCA DO IFPB - CAMPUS CAJAZEIRAS

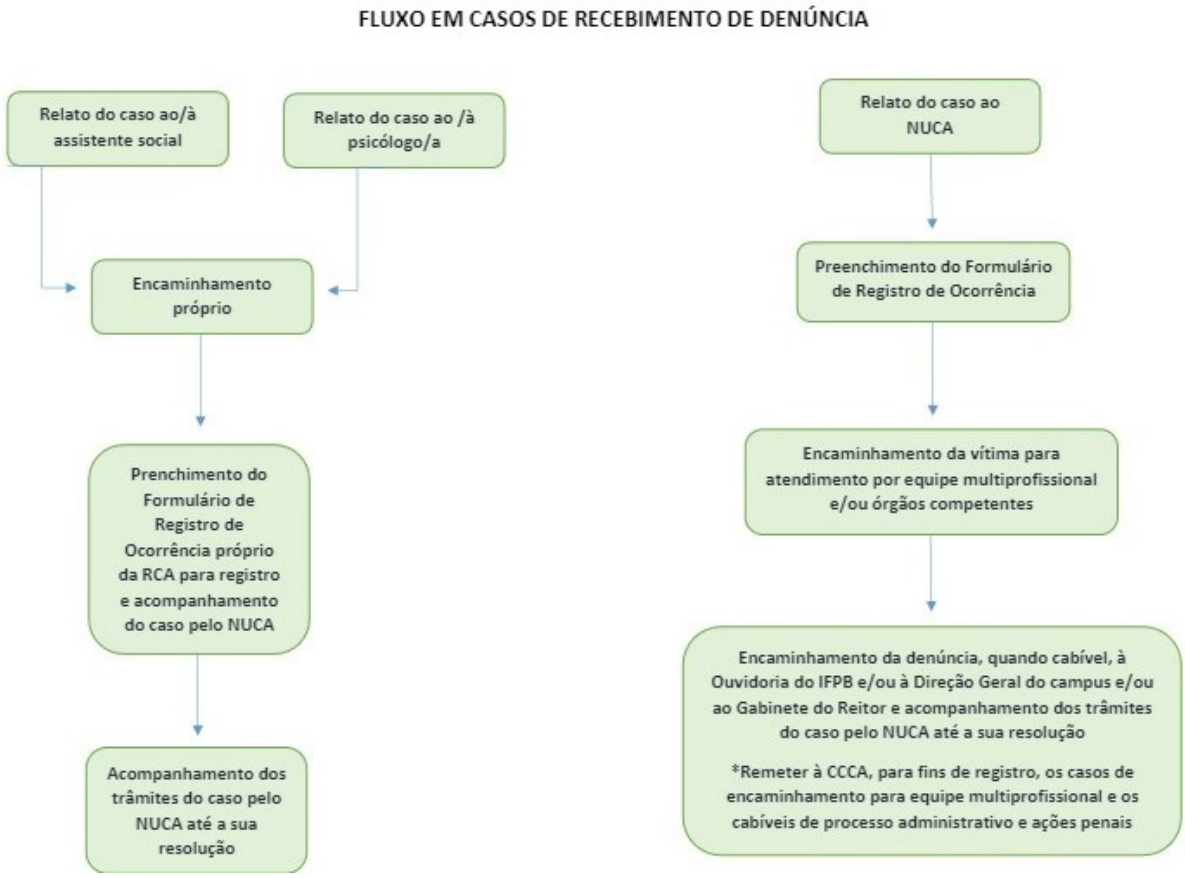
Conforme ilustrado na Figura 1, o processo de realização de uma denúncia de *bullying* e assédio se inicia com a vítima se locomovendo até o departamento do NUCA. Posteriormente uma conversa é conduzida entre a vítima e os representantes do NUCA, permitindo que a denúncia seja feita verbalmente. A vítima recebe um formulário[1] para preenchimento, contendo seus dados pessoais e detalhes sobre o ocorrido. Esse procedimento visa garantir um registro adequado das denúncias e assegurar que haja apoio e testemunhas presentes durante o processo.

O formulário tem como objetivo coletar informações essenciais sobre o relato de *bullying* ou assédio, permitindo uma documentação precisa e eficaz das ocorrências. Ao preencher o formulário, o denunciante fornecerá detalhes relevantes sobre a situação ocorrida, bem como seus próprios dados pessoais para fins de identificação, ou escolher a opção de anonimato. Essas informações serão posteriormente utilizadas para análise, investigação e tomada de medidas apropriadas para lidar com a situação.

[1] https://www.ifpb.edu.br/prae/rede-de-combate-ao-assedio/legislacao-e-documentos-pertinentes/formulario-de-registro-de-ocorrencia.pdf/view

O formulário é dividido em duas partes, a primeira parte é onde o denunciante preenche suas informações pessoais e a segunda parte é onde o denunciante descreve as características do denunciado juntamente com uma descrição do ocorrido.

Figura 1 – Fluxograma da realização de uma denúncia no NUCA



Fonte: Elaborado pelo autor, Luis F. Marques, e Kauê R. Silva (2023).

Após a realização da denúncia e o preenchimento do formulário, os membros do NUCA encaminham a vítima para o atendimento clínico ou psicológico. Logo, em seguida, elaboram um relatório que documenta os detalhes da denúncia, levando em consideração a existência de provas, caso a vítima possua. Uma vez concluído o relatório, é encaminhado por meio de um processo aberto no Sistema Unificado de Administração Pública (SUAP[2]) utilizado pelos servidores do IFPB. Esse processo solicita que a Direção Geral analise o relatório e forneça um retorno dentro de um prazo de 5 (cinco) dias. Posteriormente, o representante do NUCA informa ao denunciante sobre as medidas adotadas pela Direção Geral com base na análise realizada.

[2] https://suap.ifpb.edu.br/accounts/login/

2.4 ARQUITETURA DE *SOFTWARE*: *CLEAN ARCHITECTURE*

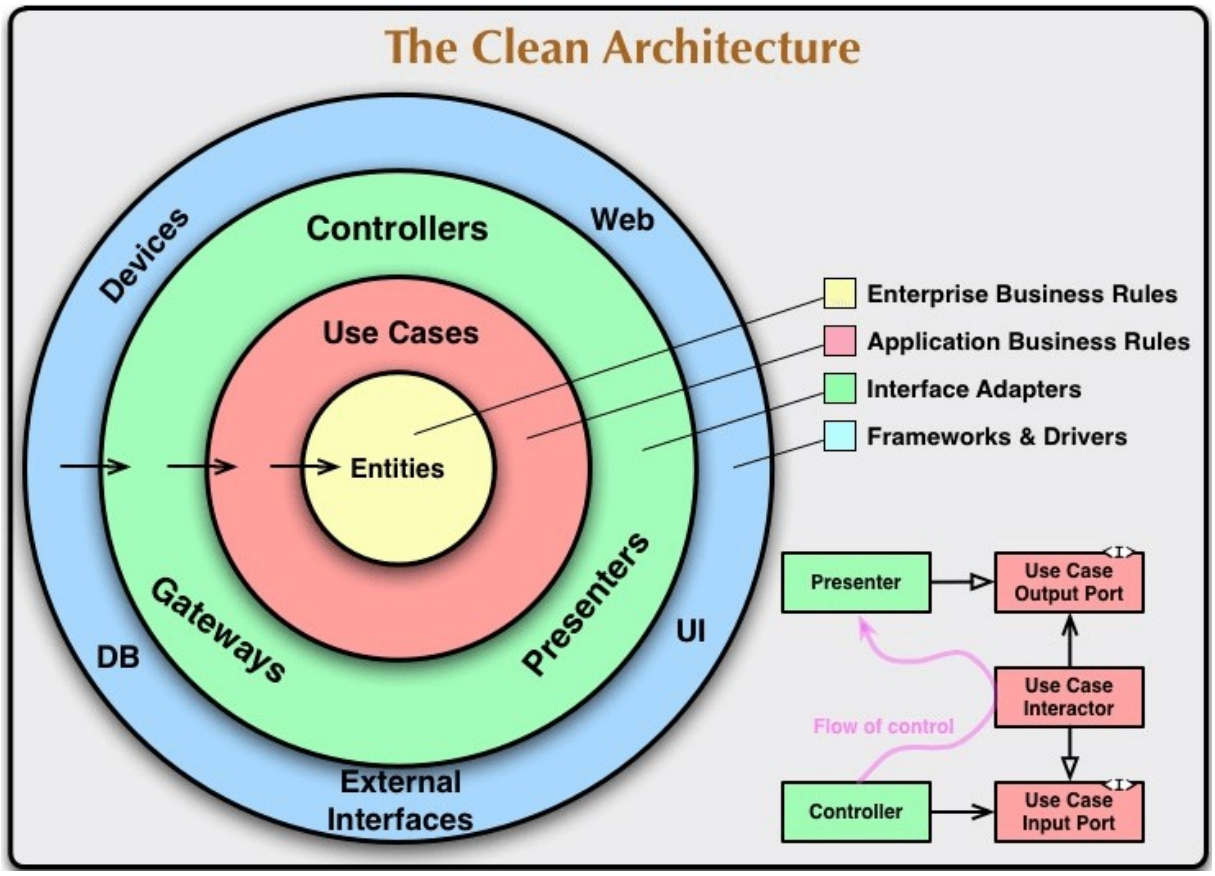
A *Clean Architecture* (Arquitetura Limpa), foi proposto por Martin (2019) em seu livro Arquitetura Limpa: O guia do artesão para estrutura e *design* de *software*. O objetivo deste modelo de arquitetura é ser utilizado no *design* de codificação a fim de facilitar a manutenção, testes e a evolução do *software* em desenvolvimento através de um sistema de camadas e um baixo grau de dependência (MARTIN, 2019).

Segundo Martin (2019), ninguém precisa de muitas habilidades e conhecimentos para fazer um *software* funcionar. Discentes do ensino médio fazem isso o tempo todo. Jovens universitários iniciam projetos bilionários com esboços de algumas linhas de código (MARTIN, 2019). O código que produzem pode não ser bonito, mas funciona. Fazer direito é outra questão. Criar um *software* de maneira correta é difícil e requer conhecimentos e habilidades que a maioria dos jovens programadores ainda não adquiriu (MARTIN, 2019).

De acordo com Martin (2019), não há diferença entre *design* e arquitetura. O autor define a palavra “arquitetura” como uma palavra que é usada no contexto de algo em um nível mais alto que independe dos detalhes de níveis mais baixos. Já por outro lado, define a “*design*” como uma palavra que remete a estruturas e decisões de nível mais baixo, porém, o autor termina afirmando que os detalhes de baixo nível e a estrutura de alto nível são partes de um mesmo todo e que juntos formam um tecido que define e molda o sistema.

A arquitetura representa as decisões de um *software* realizado em seu desenvolvimento resultando em sua estrutura atual, como sua divisão de camadas e organização do código, com o propósito de facilitar seu desenvolvimento, implementação, operação e manutenção (MARTIN, 2019). O principal objetivo da arquitetura de *software* é minimizar os recursos humanos necessários para construir e manter um determinado sistema (MARTIN, 2019).

Figura 2 – Arquitetura limpa



Fonte: Sampaio (2021)

A Figura 2, demonstra como deve ser feita a divisão das camadas do *software* em desenvolvimento, segundo Bueno (2021) essas camadas são divididas em:

- *Entities* (Entidades): As entidades do sistema são responsáveis por encapsular as regras de negócio do sistema e outras funções referentes a manipulação de dados no sistema. São as menos prováveis de haver mudanças caso algo for solicitado;
- *User* Cases (Casos de Uso): Aqui são encapsulados e implementados todos os casos de uso que serão invocados pelas camadas externas. Estes casos de uso são responsáveis por administrar o fluxo de dados para as regras de negócio, validação e persistência. Uma vez que haja modificações nessa camada, não poderá provocar modificações na camada de entidades e uma modificação externa não pode resultar na modificação desta camada;
- *Interface Adapters* (Adaptadores de interface): Esta camada é responsável por

entregar o código responsável por fazer a conversão do formato de dados utilizado pelas entidades e casos de uso;

- *Frameworks & Divers* (*Frameworks* e *Drivers*): Ferramenta mais externa que geralmente é composta por bancos de dados, *frameworks* e etc;

A regra de dependências é mostrada na Figura 2, no qual as dependências da camadas são representadas pelas setas de “fora para dentro”, ou seja, cada camada possui conhecimento somente da camada de um nível mais baixo e nunca ao contrário. Esta forma de fluxo só é possível graças ao princípio de inversão de dependências, que segundo Gonçalves e Mendes (2022), este princípio trata de sistemas flexíveis que não podem ter dependências de código fonte e sim possuir referências a abstrações, isso torna o sistema flexível o suficiente para que suas implementações possam mudar com um mínimo impacto em dependentes.

Diante do propósito de desenvolver uma *API* com características distintas e desejáveis, tais como uma estrutura claramente definida, facilidade de manutenção, a capacidade de criação de testes eficazes e flexibilidade para a incorporação de novos requisitos, optou-se pela implementação da Arquitetura Limpa. Este conjunto de práticas arquiteturais, proposto por Martin (2019), se alinha de maneira ímpar com tais objetivos.

2.5 *APIS REST*: CONCEITO E BENEFÍCIOS

O termo *API* significa “*Application Programming Interface*” (Interface de Programação de Aplicações, em português) (MELO, 2021). Segundo Melo (2021), uma *API* se trata de um conjunto de regras, rotinas e padrões que facilitam a comunicação e a troca de informações entre diferentes sistemas.

Enquanto o termo *REST* significa “*Representational State Transfer*” (Transferência de Estado Representacional, em português), sua principal característica está em fornecer uma interface uniforme para acesso a representações de recursos através de um endereçamento também uniforme (BERENGUEL et al., 2008). Recursos podem ser generalizados como qualquer coisa, como um copo ou um livro. A interface uniforme dos métodos é dada pelos métodos definidos no *Hypertext Transfer Protocol* (HTTP), sendo os principais o *GET*, *POST*, *DELETE* e *PUT* (BERENGUEL et al., 2008).

Segundo Sousa (2020), uma requisição *HTTP* geralmente possui uma estrutura definida que é composta pela seguinte composição:

- **Verbo *HTTP***: Que indica o tipo de operação a ser realizada;
- **O *Header* (Cabeçalho)**: Que permite o cliente enviar informações sobre o pedido;
- **O *Path* (Caminho)**: Indica o caminho para um recurso que a *API* disponibiliza;
- **O *Body* (Corpo)**: Contém dados de uma mensagem do cliente, pode ser opcional, no caso de uma requisição de busca de dados;

Ainda segundo Sousa (2020), os principais verbos *HTTP* e que são mais importantes de serem conhecidos são:

- ***GET***: Utilizado para recuperar dados específicos (por *ID*) ou recuperar uma coleção de dados;
- ***POST***: Utilizado para criar um novo recurso no sistema, por exemplo, persistir dados no banco de dados;
- ***PUT***: Utilizado para atualizar dados específicos (por *ID*) que já estão persistidos no banco de dados;
- ***DELETE***: Utilizado para deletar/apagar dados (por *ID*) que estão persistidos no banco de dados;

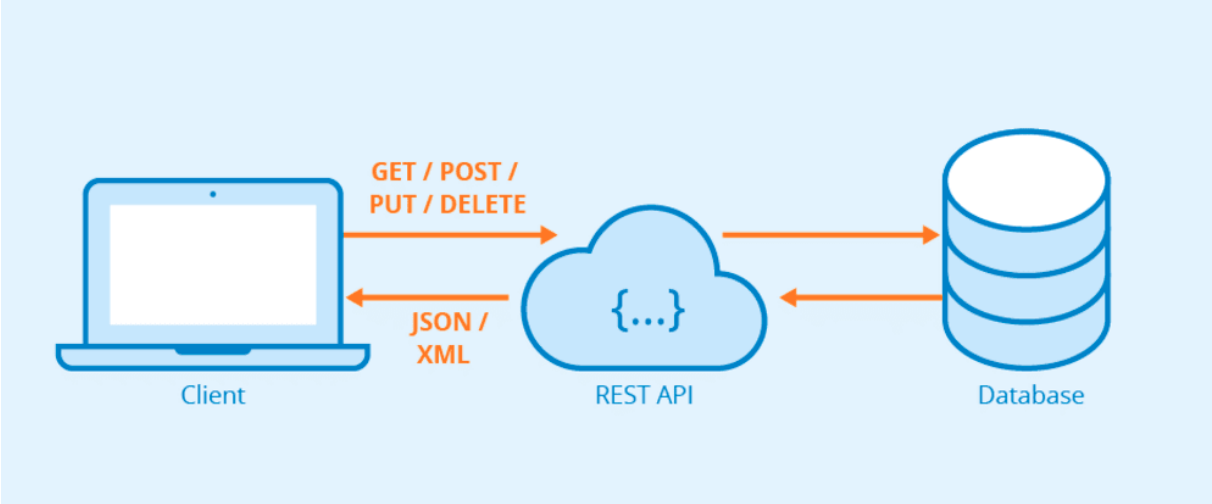
Segundo a AWS, uma importante característica de uma *API REST* é a ausência de um estado, isso significa que o servidor não guarda informações do cliente entre suas solicitações. As solicitações são simples e direcionadas a *endpoints*[3]. que são simples, como uma *URL* de pesquisa de um *site* no navegador, a resposta do servidor corresponde a dados simples e ocorrem por meio de objetos *JSON* que não precisam ser renderizados em páginas *web*.

Ainda de acordo com a AWS, os benefícios da utilização das *APIs REST* são muito benéficas para a integração com sistemas, seus benefícios são:

- **Integração**: são utilizadas para integrar sistemas de *software* já existentes e ganham em velocidade, já que cada funcionalidade não precisa ser escrita zero;
- **Expansão**: devido a demanda, as empresas podem adaptar uma *API REST* a diferentes plataformas, assim atendendo a muitos clientes, por exemplo, uma *API* de mapas pode permitir a integração com sistemas *web* e *mobile*.

[3] Um *endpoint* é uma *URL* que representa uma entidade ou um recurso dentro da *API*.

Figura 3 – Fluxo de uma *API REST*



Fonte: Fonseca (2021)

Conforme ilustrado na Figura 3, O Fluxo de uma *API REST*, inicia quando um cliente por meio de um sistema *web*, *mobile* ou *desktop*, envia uma requisição *HTTP* formatada em *JSON* para o servidor, onde a *API REST* está localizada. A *API*, então, processa esses dados recebidos pelo cliente, aplicando as regras de negócio necessárias. Em seguida, a *API* se conecta ao banco de dados para persistir ou recuperar as informações fornecidas pelo cliente. Uma vez que os dados são armazenados ou recuperados com sucesso, a *API* envia de volta uma mensagem na representação *JSON* ou *XML* para o cliente, comunicando o resultado da requisição realizada.

3 TRABALHOS RELACIONADOS

Na seleção de projetos relacionados, foram estabelecidos critérios específicos com o intuito de direcionar a pesquisa de forma mais precisa. Esses critérios visam identificar trabalhos científicos relevantes no contexto de registro de denúncias em Instituições de Ensino (IEs). Os critérios estabelecidas para a escolha dos projetos incluíram:

- Trabalhos científicos, como artigos e TCC entre o período de 2018 a 2023 realizados no Brasil;
- Trabalhos que abordam sistemas de *APIS* para registro de denúncias em IEs;
- Os trabalhos encontrados possuírem funcionalidades para auxiliar no combate de *bullying* e assédio;

Com base nesses critérios acima, foram selecionados cinco trabalhos que abordam os critérios estabelecidos, porém entre eles, apenas dois trabalhos se destacam por serem relacionados à proposta da *NoBullying API*. Os trabalhos selecionados como referência na pesquisa são: “BEKID” um sistema proposto por Maia e Júnior (2022) e “PANDORA” um sistema proposto por MARINHO et al. (2018).

3.1 BEKID: UM AMBIENTE PARA AUXÍLIO NO COMBATE AO *BULLYING* NA ESCOLA

Maia e Júnior (2022) propuseram um sistema *web* de nome BEKID com o intuito de auxiliar as escolas a identificarem possíveis casos de *bullying*. Este sistema conectou várias funcionalidades importantes como o compartilhamento de informações sobre o *bullying*, o monitoramento dos alunos ao longo do tempo, a coleta de informações sobre as emoções, a geração de relatórios e plataforma de gerenciamento por parte da escola. Esse sistema *web* possibilita a aproximação entre escola e aluno, além de guardar o anonimato das denúncias de *bullying*.

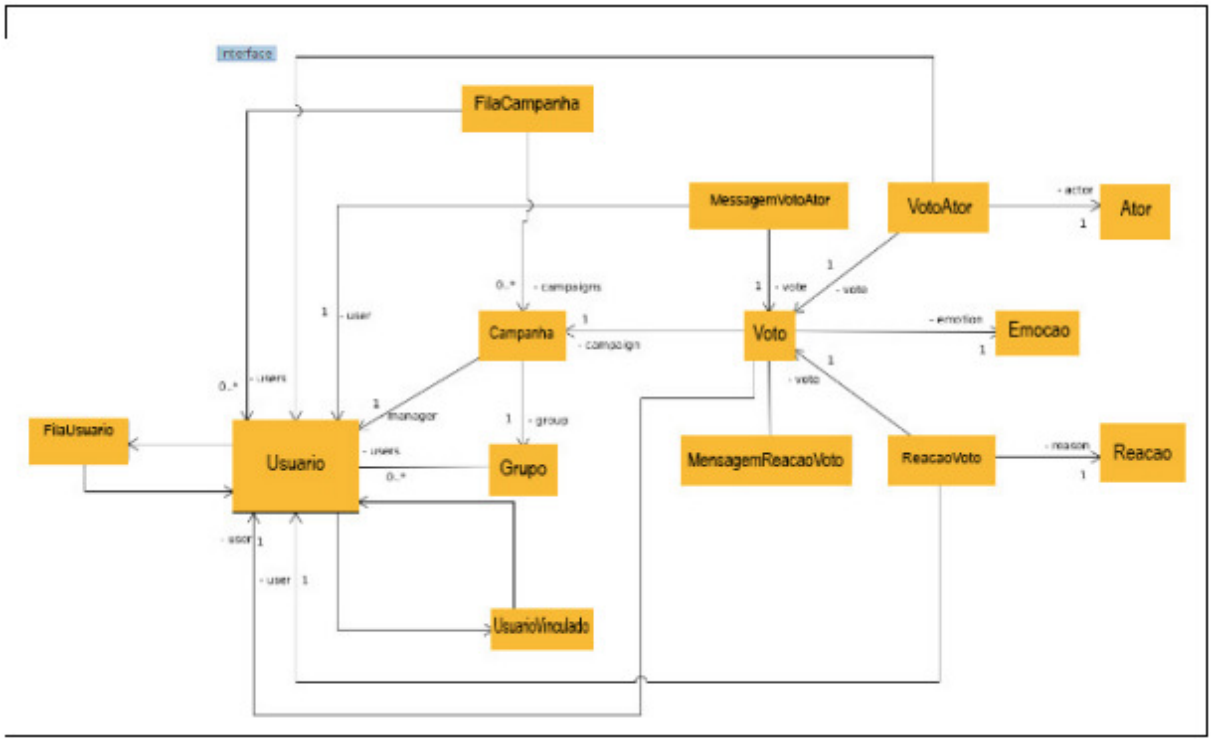
O sistema proposto leva em consideração duas versões, a primeira versão é por onde os dados que dizem respeito aos sentimentos dos alunos são coletados, esta versão é disponibilizada para os alunos no processo de criar um novo cadastro. A segunda versão é por onde a escola recebe os dados fornecidos pelos alunos, além de

ter permissão para criar grupos de alunos e campanhas, esta versão é disponibilizada para a escola no processo de criar um novo cadastro como usuário “Escola”.

A fins de identificar e prevenir possíveis casos de *bullying*, o sistema disponibiliza um módulo para, somente, o usuário registrado com “Escola” gerar um relatório contendo os dados informados pelos alunos e posteriormente conseguir identificar os comportamentos e sentimentos dos alunos no dia a dia na escola.

A Figura 4 ilustra todo o processo no qual o desenvolvimento do sistema é guiado. Observa-se os dois módulos que compõem o sistema BEKID, que são o módulo *front-end* e o módulo *back-end*. O *NodeJs*[4] foi escolhido para o desenvolvimento da *API REST*, pois é uma tecnologia que demanda um baixo nível de recursos de infraestrutura, além de ser um *software* de código aberto e que permite a execução de códigos *Javascript*[5] fora de um navegador *web*. Para a persistência de dados, foi escolhido o *PostgreSQL*[6]. Já para o *front-end* e consumo da *API*, foi escolhida a tecnologia *VueJs*[7] por se tratar de uma tecnologia de alto desempenho e baixa complexidade.

Figura 4 – BEKID: Diagrama de relacionamento do sistema



Fonte: Maia e Júnior (2022)

[4] https://nodejs.org/pt-br
[5] https://developer.mozilla.org/pt-BR/docs/Web/JavaScript
[6] https://www.postgresql.org/
[7] https://vuejs.org/

Por fim, os autores concluem afirmando que o trabalho desenvolvido contribui com a aproximação entre a escola e o aluno, além de guardar o anonimato e melhorar a condição para o relato de agressões e pedidos de ajuda, assim permitindo que os estudantes se sintam mais seguros em um ambiente seguro e favorável ao aprendizado. O autor complementa informando que o sistema foi apresentado e testado por alunos do Ensino Fundamental 2 (dois) em forma de uma simulação para a criação de campanha com o intuito de recolher dados a respeito dos sentimentos do aluno e os resultados obtidos foram positivos, indicando a utilidade, facilidade e importância do aplicativo no combate ao *bullying* na escola.

3.2 PANDORA: UMA PROPOSTA DE COMBATE A VIOLÊNCIA NAS ESCOLAS POR MEIO DA VIRTUALIZAÇÃO DE GRUPOS OPERATIVOS

MARINHO et al. (2018) propôs uma ferramenta de auxílio na prevenção e combate a violências por meio da virtualização de grupos operativos em uma plataforma online denominada de PANDORA. Primeiramente, o autor explica que foi realizada uma leitura de artigos que abordaram a temática da violência nas escolas e o processo de gestão de grupos operativos, e logo após o momento de entendimento de principais conceitos, foi iniciado o desenvolvimento do protótipo da aplicação *web* chamado de PANDORA.

A aplicação é composta por duas etapas, na qual a primeira está direcionada aos usuários, estes podem interagir com os grupos criados na aplicação. A segunda etapa pertence aos administradores, que podem criar e gerenciar esses grupos operativos.

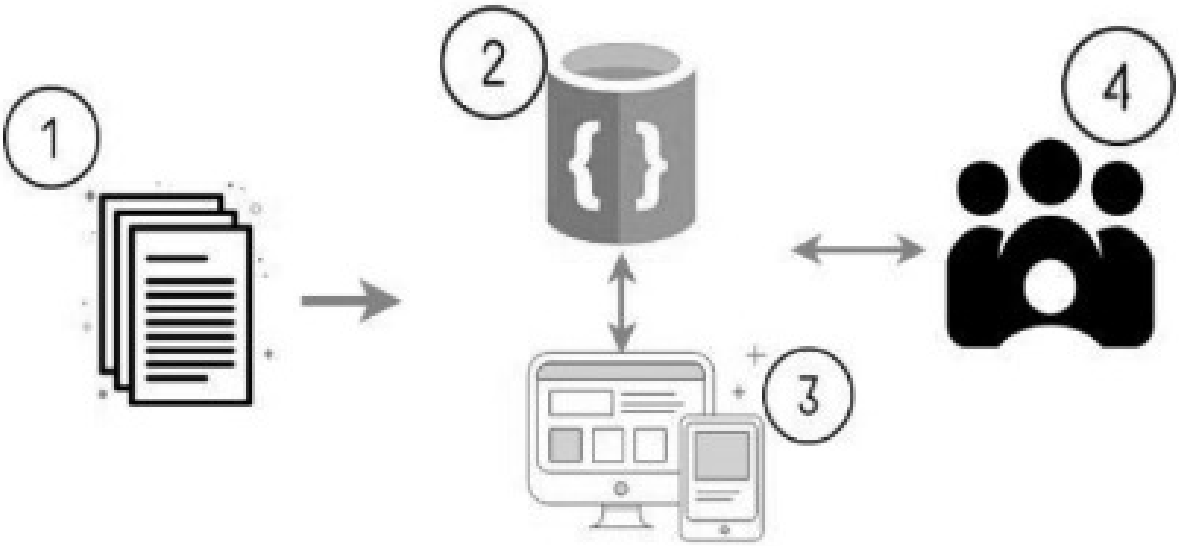
O autor afirma que esta ferramenta pode se tornar uma forma eficiente de guiar os funcionários do corpo docente de instituições de ensino a combater indícios de variados tipos de violência presentes no meio social de uma ambiente escolar. O sistema PANDORA foi desenvolvido utilizando as seguintes tecnologias: *Angular*[8] para o desenvolvimento e consumo da *API* por parte do sistema *web* e *JSON Server*[9] para o desenvolvimento da *API*.

A Figura 5 representa o fluxo de metodologia utilizado para o desenvolvimento do sistema PANDORA, no qual é dividida em quatro etapas de desenvolvimento:

[8] https://angular.io/
[9] https://www.npmjs.com/package/json-server

- A Primeira etapa (1): É destinada ao entendimento do escopo e definição de funcionalidades primárias a serem prototipadas;
- A Segunda etapa (2): É onde fica localizado o desenvolvimento da *API* utilizando o *JSON Server*, onde foram criadas tabelas responsáveis por armazenar os dados que são gerados pelos grupos;
- A Terceira etapa (3): É onde são criadas as interfaces do usuário com módulos criados com o *Angular*;
- A Quarta etapa (4): É onde são criados as interfaces administrativas que são responsáveis por criar e gerenciar os grupos operativos;

Figura 5 – PANDORA: Fluxo de metodologia



Fonte: MARINHO et al. (2018)

No fim, o autor afirma que a criação de uma ferramenta que é baseada na grupalidade e que tenha como foco o protagonismo estudantil pode auxiliar no combate e na prevenção da violência nas escolas, seja esta qual for, visto que a implementação desse sistema pode ajudar as demandas de cada escola que optar por implementá-lo.

4 *NOBULLYING API*

A *Nobullying API* é uma *API* desenvolvida e pensada para a prevenção e gerenciamento de denúncias de *bullying* e assédio no IFPB campus Cajazeiras. Essa *API* oferece uma variedade de serviços contendo funcionalidades que visam proporcionar um ambiente seguro e acolhedor para os discentes e servidores do IFPB Campus Cajazeiras. Com o intuito de auxiliar o NUCA a receber e acompanhar denúncias sobre *bullying* e assédios que ocorrem no campus.

Pensada para auxiliar o recebimento e gerenciamento de denúncias de *bullying* e assédio pelo NUCA, a *NoBullying API* é projetada para automatizar os processos manuais presentes no fluxograma de denúncias de *bullying* e assédio do NUCA e disponibilizar integrações com diferentes plataformas e sistemas. A *NoBullying API* atua entre o denunciante e o NUCA, assim automatizando o fluxo manual de registro de denúncias. Já que o denunciante pode registrar uma denúncia sem precisar ir presencialmente ao departamento do NUCA.

Nas Seções seguintes, são descritos os métodos de coleta de requisitos, além da análise dos mesmos e, logo após, serão apresentadas a arquitetura utilizada no desenvolvimento da *API* e uma análise comparativa do trabalho proposto com os trabalhos relacionados supracitados.

4.1 REQUISITOS FUNCIONAIS DO SISTEMA

Esta seção apresenta os requisitos funcionais da *NoBullying API*, uma *API* desenvolvida para auxiliar o NUCA do IFPB campus Cajazeiras a coletar e gerenciar denúncias de *bullying* e assédio. Os requisitos funcionais descrevem as funcionalidades específicas que a *API* oferece para atender as necessidades dos usuários, as quais estão listadas no Quadro 1. No Apêndice A, estão disponíveis informações detalhadas a respeito das *user stories*. Essas *user stories* descrevem os requisitos e funcionalidades específicas da *NoBullying API*, oferecendo uma visão abrangente das necessidades dos usuários e dos objetivos do sistema. Por meio dessas histórias de usuário, é possível compreender melhor como a *API* foi projetada para atender às demandas e proporcionar soluções eficazes no combate ao *bullying*.

Quadro 1 – Requisitos funcionais do sistema

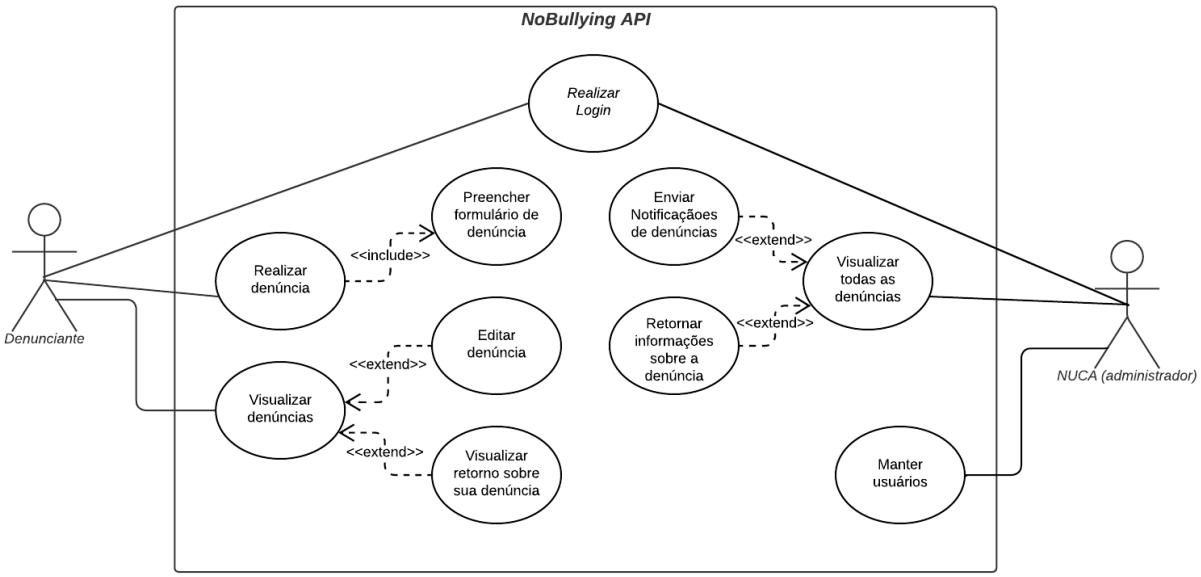
Requisitos Funcionais (RF)	Descrição
RF01: Realizar *login*	O sistema deve permitir que os usuários realizem *login* diferenciando um usuário comum e um usuário administrador.
RF02: Proteger *endpoints*	O sistema deve oferecer uma forma segura de autenticação, não permitindo que os usuários consigam realizar funções que não possuam acesso.
RF03: Criar denúncias	O sistema deve permitir que os usuários comuns possam registrar denúncias de *bullying* e assédio, sendo da escolha do usuário se a denúncia será anônima ou não.
RF04: Visualizar denúncias	O sistema proporcionará aos usuários comuns a capacidade de visualizar as denúncias que eles próprios registraram, enquanto os usuários administradores terão acesso à visualização de todas as denúncias registradas no sistema.
RF05: Editar denúncias	O sistema deve permitir que os usuários comuns possam editar suas denúncias.
RF06: Notificar	O sistema deve oferecer uma sistema de notificações, onde usuários comuns possam visualizar notificações direcionados a eles, enquanto usuários administradores possam ver, enviar, atualizar e excluir notificações.
RF07: Feedback	O sistema deve permitir que os usuários comuns possam receber atualizações a respeito das medidas tomadas sobre sua denúncia e usuários administradores possam enviar e atualizar essas respostas.
RF08: Manter Usuários	O sistema deve permitir que os usuários administradores possam adicionar, visualizar, atualizar e remover usuários do banco de dados.

Fonte: Elaborado pelo autor (2023).

4.2 ANÁLISE DE REQUISITOS

Após realizar o levantamento dos requisitos, foram construídos um Diagrama de Casos de Uso e um Diagrama de Classes. Os casos de uso, ilustrados na Figura 6, são representações detalhadas da interação do usuário e do sistema, descrevendo as funcionalidades e fluxo de trabalho. No Apêndice B está disponível um Diagrama de Classes que ilustra a estrutura e as relações entre as classes da *NoBullying API*. Esse Diagrama de Classes oferece uma representação visual das entidades e dos atributos presentes na *API*, facilitando a compreensão da organização do sistema.

Figura 6 – Diagrama de casos de uso



Fonte: Elaborado pelo autor (2023).

O diagrama de casos de uso, ilustrado pela Figura 6, apresenta um ator principal, chamado “Denunciante”, que realiza diversas interações com o sistema. Essas interações são representadas por casos de uso, que por sua vez, descrevem as funcionalidades que são realizadas pelo ator principal. Servirão como base para a construção dos requisitos funcionais do Quadro 1. Além dos casos de uso pertencentes ao ator principal (Denunciante), existem os casos de uso que pertencem ao ator nomeado “NUCA” (administradores do sistema). Estes atores atuam em algumas funções que são supracitadas no Quadro 1.

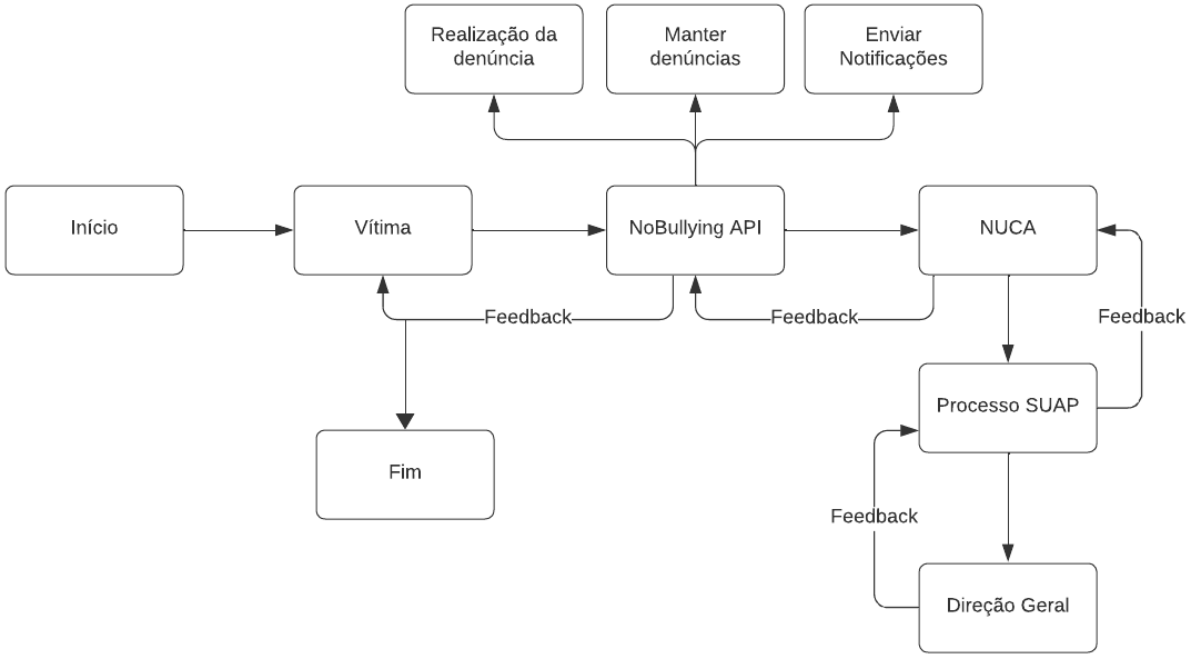
Essas interações representadas no Diagrama de Casos de Uso proporcionam uma visão geral das funcionalidades e dos fluxos de trabalho propostos no sistema, demonstrando as principais ações realizadas pelo Denunciante que pode realizar, editar e visualizar as denúncias. Também é possível ter uma visão geral das funcionalidade e do fluxo de trabalho para a realização de ações pelo Administrador para visualizar, enviar *feedback* de denúncias e enviar notificações, além de gerenciar os usuários registrados no sistema.

4.3 FLUXOGRAMA DE REALIZAÇÃO DE DENÚNCIAS DO NUCA UTILIZANDO A *NOBULLYING API*

Conforme supracitado e ilustrado na Figura 1, o NUCA possui um sistema de coleta de denúncias de forma inteiramente manual, em que o denunciante precisa se

locomover até o departamento do NUCA para conseguir realizar sua denúncia, e para o registro da denúncia, o denunciante precisa preencher de forma manual um documento de formulário. Utilizando a *NoBullying API* integrada a um sistema, todos os usuários poderão realizar denúncias sem ter a necessidade da locomoção até o departamento do NUCA e sem precisar preencher um documento de formulário de forma manual. A fim de ilustrar o fluxo de trabalho do NUCA entre usuários e o sistema contendo os serviços do *NoBullying API*, foi construído um diagrama de atividade. A ilustração da Figura 7 representa o diagrama de atividades para a realização de denúncias com o NUCA utilizando a *NoBullying API*.

Figura 7 – Fluxo de denúncias com o NUCA utilizando a *Nobullying API*



Fonte: Elaborado pelo autor (2023).

Observando a ilustração da Figura 7, a *Nobullying API* atua entre a vítima e o departamento do NUCA, assim facilitando a recepção de denúncias de *bullying* e assédio por parte dos discentes e membros do IFPB campus Cajazeiras. O fluxo de atividades para a realização das denúncias de *bullying* e assédio atua da seguinte forma:

1. A vítima, que deseja realizar uma denúncia, utiliza um sistema que integra a *NoBullying API*;
2. A *NoBullying API* fornece funcionalidades que ajudam a realizar uma denúncia,

como: A própria realização da denúncia, o sistema manter salva as denúncias realizadas pelos usuários e receber notificações sobre as denúncias registradas;

3. As denúncias realizadas através da *NoBullying API* e salvas no banco de dados, serão expostas para os administradores através de outro sistema que somente o departamento do NUCA tem acesso;

4. Os membros do NUCA podem enviar notificações para os usuários a respeito das medidas tomadas sobre as denúncias realizadas;

5. Logo após que a direção geral revisar o processo feito e retornar um resultado, o departamento do NUCA pode enviar para o denunciante informações sobre as medidas tomadas a respeito de sua denúncia;

O fluxo apresentado na Figura 7 oferece diversas vantagens em comparação com o fluxo manual mostrado na Figura 1. Primeiramente, a automatização do registro e gerenciamento das denúncias elimina a necessidade de realizar essas atividades de forma manual, tornando o processo mais eficiente e preciso. Além disso, o sistema permite que os usuários realizem suas denúncias de forma remota, sem precisar comparecer fisicamente ao departamento do NUCA. Isso proporciona maior comodidade e acesso facilitado para os denunciantes.

Outra vantagem é a facilidade dos membros do NUCA em gerenciar todas as denúncias registradas pelos usuários comuns. O sistema centraliza as informações e fornece ferramentas de acompanhamento e análise das denúncias, além disso, a função de envio de notificaçõs facilita a comunicação entre os integrandes do NUCA e os usuários que utilizam o sistema. Portanto, a adoção da *NoBullying API* traz inúmeras vantagens em relação ao sistema manual anteriormente utilizado pelo NUCA.

4.4 INFRAESTRUTURA DO SISTEMA

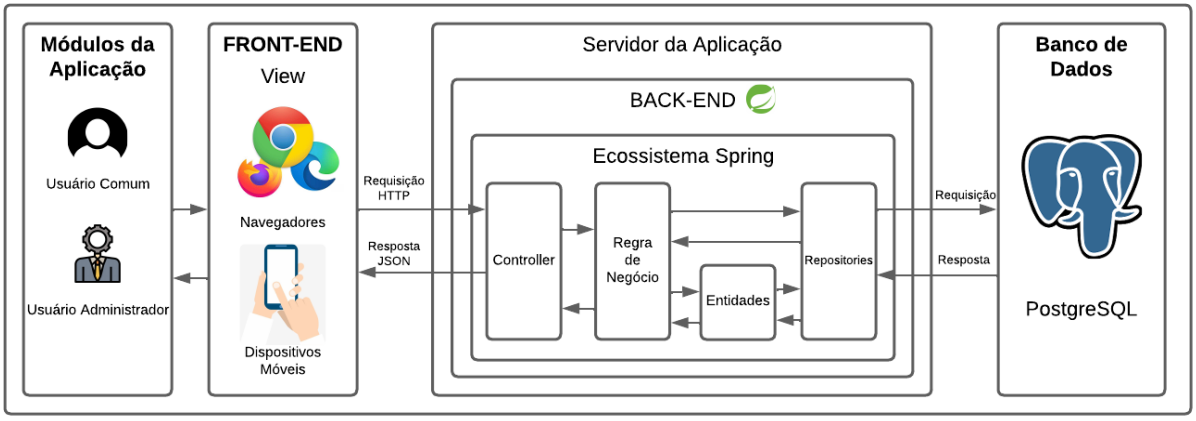
A infraestrutura do sistema é dividida em camadas. A Figura 8 ilustra a infraestrutura da solução proposta para o desenvolvimento da *NoBullying API*, mostrando as tecnologias abordadas. A *NoBullying API* é desenvolvida utilizando a linguagem de programação *Java*[10] na sua versão 17 juntamente com o *Spring Framework*[11], dependências do ecossistema do *Spring* e o gerenciador de dependências *Apache Maven*[12]. A escolha do *Spring Boot* oferece vantagens para o início e o decorrer do projeto, vantagens estas que são a facilidade no início de um projeto. Pois, algumas

[10] https://www.java.com/pt-BR/
[11] https://spring.io/
[12] https://maven.apache.org/

configurações iniciais já vem pré-configuradas, ou seja, não necessita de o desenvolvedor gastar tempo começando uma *API* do zero, já que o *Spring Boot* provém muitos recursos na sua inicialização. O *Spring Framework* contribui com projetos de forma ágil e organizada, além da facilidade de criação de *APIS RESTful*. Inicialmente a *NoBullying API* utiliza dependências do próprio ecossistema *Spring*, como o *Spring Boot*, *Spring Security*, *Spring Data JPA*, *Spring MVC* entre outros.

O banco de dados utilizado no projeto é o *PostgreSQL*[13], no Apêndice B está disponível o projeto de modelagem do banco de dados. O *PostgreSQL* oferece uma ampla gama de funcionalidades que contribuem para o sucesso da *NoBullying API*. Uma das principais funcionalidades é a capacidade de armazenar e manipular dados de forma estruturada, seguindo um modelo relacional. Permitindo uma organização eficiente das informações, facilitando consultas e análises complexas.

Figura 8 – Infraestrutura do sistema *NoBullying API*



Fonte: Elaborado pelo autor (2023).

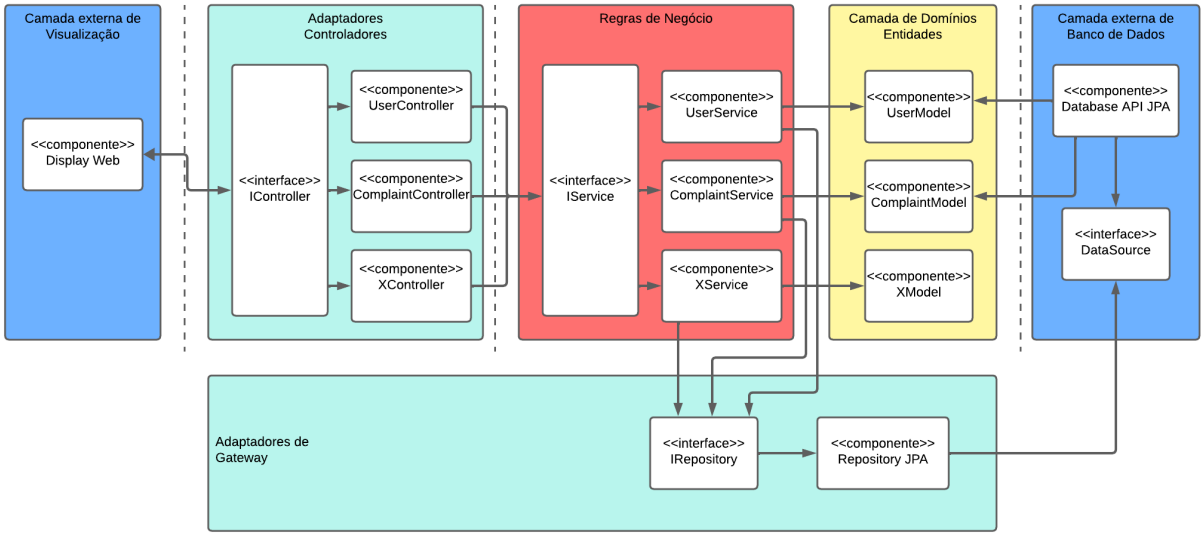
Analisando a ilustração da Figura 8, é possível observar a transmissão de informações entre a aplicação *front-end* com a *API*, ou seja, o *back-end* que é o objetivo da *NoBullying API*. Os módulos da aplicação são as diferentes plataformas que consomem a *NoBullying API*, esses módulos são separados em módulo de usuário comum e módulo de usuário administrador. Seguindo, os módulos da aplicação através de uma aplicação *front-end* fazem requisições *HTTP* com o corpo da requisição no formato *JSON* para o *back-end* que, por sua vez, recebe os dados pelo controlador, passa por um conjunto de regras de negócio e com a dependência do *Spring Data JPA*, acessa o repositório e persiste os dados no banco de dados *PostgreSQL*.

[13] https://www.postgresql.org/

4.5 ARQUITETURA DO SISTEMA

A ilustração da Figura 9 apresenta a representação visual da organização e atuação da *NoBullying API* com base no padrão de projeto *Clean Architecture*. Essa representação possibilita a separação clara das responsabilidades do sistema, assim possibilitando a identificação do fluxo de dados por trás da *NoBullying API*, além de facilitar a manutenção e expansão do sistema.

Figura 9 – Arquitetura do sistema *NoBullying API*



Fonte: Elaborado pelo autor (2024).

Ao examinar a Figura 9, torna-se evidente o trajeto dos dados dentro da camada de *backend* da aplicação *NoBullying API*, proporcionando uma compreensão clara do fluxo no contexto do projeto baseado na arquitetura *Clean Architecture*. Essa implementação da *Clean Architecture* possibilitou a organização dos serviços da aplicação em camadas, viabilizando a separação de responsabilidades, como a distinção entre lógica de negócios, apresentação e controle de fluxo. O fluxo de dados apresenta-se da seguinte forma:

1. O usuário realiza uma solicitação aos serviços da *NoBullying API* por meio de uma requisição *HTTP* através de alguma aplicação *web* ou *mobile* que tenha integrado a API (Camada externa de Visualização);
2. A *NoBullying API* recebe a solicitação do controlador (interface *IController*) ao qual foi encaminhada. Os controladores são encarregados de disponibilizar *endpoints* para serviços específicos, como salvar uma nova denúncia, e aplicar validações

de informações utilizando *DTO's*[14]. Posteriormente, os controladores encaminham os dados recebidos na solicitação para a camada de serviços (interface *IService*);

3. A camada de serviços (Regras de Negócio) é responsável por aplicar as regras necessárias nas informações provenientes dos controladores. Essa camada pode utilizar entidades (Camada de Domínios) para construir *DTO's*. Após a aplicação das regras de negócio, os dados são enviados para o repositório (interface *IRepository*) associado ao método *HTTP* da solicitação;

4. O repositório é responsável pelas operações de *CRUD* conforme informado pela camada de serviços, estabelecendo uma conexão direta com o banco de dados (componente *Repository JPA*) para executar essas operações;

5. Subsequentemente, é gerada uma resposta que percorre todas as camadas até alcançar o controlador, que envia uma resposta no formato *JSON* para a aplicação que o usuário está utilizando;

4.6 ANÁLISE COMPARATIVA DOS PROJETOS

Com o objetivo de ressaltar as particularidades de cada aplicação, será conduzida uma análise comparativa com estudos anteriores. Essa análise levará em consideração as características das aplicações semelhantes em relação à aplicação desenvolvida neste trabalho. Com o objetivo de organizar melhor as informações no Quadro 2, atribuiu-se as seguintes nomenclaturas às aplicações similares:

- **APP 01**: BEKID - Um Ambiente Para Auxílio No Combate Ao Bullying Na Escola;
- **APP 02** - PANDORA: Uma Proposta de Combate a Violência nas Escolas por meio da Virtualização de Grupos Operativos;

[14] Objeto de Transferência de Dados, é um padrão de projeto de software usado para transferir dados entre subsistemas de um software

Quadro 2 – Comparação entre características dos trabalhos relacionados.

Características	**APP 01**	**APP 02**	***NoBullying API***
Tecnologia	*NodeJs* e *VueJs*	*Angular* e *JSON Server*	*Java* com o *framework Spring*
Multiplataforma	Não	Não	Sim
Arquitetura do sistema	*Domain Driven Design*	Arquitetura de Componentes	*Clean Architecture*
Serviço de Denúncias anônimas	Sim	Não	Sim
Automatização de processos manuais de uma Instituição de Ensino	Não	Não	Sim
Tipos de denúncias	*Bullying*	*Bullying*	*Bullying* e assédios
Serviços de acompanhamento	Sim	Não	Sim
Serviço de notificação	Sim	Não	Sim
Público alvo	Alunos, professores e direção geral escolar	Alunos e direção geral escolas	Alunos, professores, servidores e departamento do NUCA do IFPB

Fonte: Elaborado pelo autor (2023).

Ao analisar o Quadro 2, é possível identificar pontos de diferenciação quanto a pontos de semelhança. Desde o início deste trabalho, se destaca que a *NoBullying API* auxilia o NUCA no combate e prevenção de causas de *bullying* e assédio dentro do IFPB campus Cajazeiras. No APP 01 foi utilizada uma abordagem baseada em campanhas, ou seja, grupos que o usuário “Escola” cria a campanha para recolher dados dos sentimentos dos alunos. No entanto, este artigo não possui a automação de coleta de denúncias de *bullying* e assédio a todo momento, necessitando da criação de campanhas para poder coletar suas informações, além de não ter retorno para o denunciante sobre que fim levou sua denúncia.

No APP 02, foi apresentado uma proposta de auxílio e combate a violências por meio da virtualização de grupos operativo em uma plataforma *web*. Apesar do auxílio e o combate que este trabalho trás, é algo voltado às oficinas que contém agendamento de reuniões para debater sobre diferentes temas, incluindo *bullying*.

Portanto, ao analisar o Quadro 2, fica evidente a falta de adoção da *Clean*

Architecture e a ausência de disponibilidade multiplataforma nos aplicativos APP 01 e APP 02. Essas falhas resultam em deficiências no desenvolvimento, como a falta de escalabilidade ao longo do tempo e a dificuldade na manutenção, devido à ausência da abordagem da *Clean Architecture*. Além disso, esses aplicativos apresentam falhas de disponibilidade, uma vez que não oferecem suporte para diferentes plataformas. Em contraste, a *NoBullying API* adota a *Clean Architecture* e oferece suporte multiplataforma em seu desenvolvimento. Essa abordagem permite que o sistema seja escalável ao longo do tempo e facilmente mantido, devido à separação de seus módulos, além de proporcionar suporte para diferentes plataformas.

4.7 IMPLEMENTAÇÃO DA *NOBULLYING API*

Após a definição das tecnologias e a modelagem do banco de dados, a *NoBullying API* foi desenvolvida e está agora em sua versão final. Durante todo o ciclo de desenvolvimento, a *IDE* escolhida foi o *IntelliJ IDEA*[15], proporcionando um ambiente de desenvolvimento robusto e eficiente. Consequentemente, o *Insomnia*[16] (aplicativo utilizado para realizar requisições *HTTP* para *endpoints* de uma *API*) foi utilizado de maneira integrada para facilitar a realização de requisições *HTTP* aos diversos *endpoints* da *API*, agilizando o processo de teste e validação.

A documentação da *API* foi cuidadosamente preparada, incluindo representações visuais, a fim de proporcionar uma compreensão clara e abrangente das funcionalidades disponíveis. Esse cuidado não apenas facilita a integração por parte de desenvolvedores, mas também contribui para uma experiência mais intuitiva no uso da *NoBullying API*.

Os *endpoints* foram projetados com a finalidade de atender de forma eficaz todas as necessidades identificadas nos casos de uso apresentados na Figura 6. Cada *endpoint* foi implementado considerando os requisitos funcionais do sistema, conforme detalhado no Quadro 1. Dessa forma, a *NoBullying API* oferece uma solução completa e alinhada às expectativas, proporcionando um ambiente seguro e eficiente para a gestão de informações relacionadas ao combate ao bullying.

[15] https://www.jetbrains.com/pt-br/idea/
[16] https://insomnia.rest/

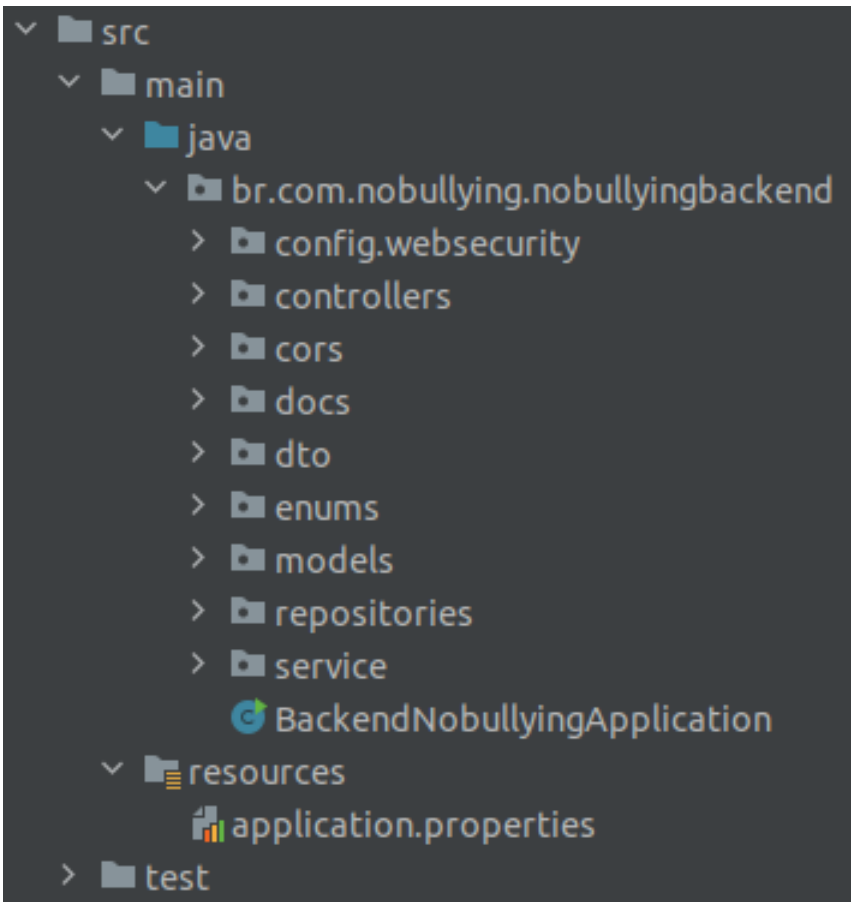
5 DISCUSSÕES E RESULTADOS

Com o intuito de apresentar os resultados alcançados, esta subseção tem como objetivo destacar não apenas as funcionalidades principais da *NoBullying API*, mas também proporcionar demonstrações práticas das funcionalidades relacionadas aos casos de uso e requisitos funcionais previamente mencionados, além de explanar as dificuldades enfrentadas ao longo do caminho de desenvolvimento da aplicação.

5.1 ARQUITETURA DE DIRETÓRIOS

Seguindo a risca a implementação da *Clean Architecture* juntamente com o *Spring MVC* foi possível criar uma arquitetura de diretório de fácil entendimento e de alta precisão, onde cada diretório armazena classes *Java* de configuração e implementação, conforme a figura a seguir:

Figura 10 – Arquitetura de Diretórios da *NoBullying API*



Fonte: Elaborado pelo autor (2024).

Como ilustrado na Figura 10, a implementação da *Clean Architecture* foi fundamental durante o desenvolvimento da *API*, com ela foi possível criar e ordenar devidas responsabilidades para diferentes pastas e classes *Java*. Estes diretórios são organizados da seguinte forma:

- ***config.websecurity***: Estes diretórios são responsáveis por armazenar as configurações de segurança do sistema, conténdo classes *Java* que tem a função de criar a autenticação e autorização utilizando um *token JWT*, além de bloquear *endpoints* para garantir que somente usuários autenticados e autorizados possam mandar requisições *HTTP* para o sistema;
- ***controllers***: Este diretório é responsável por armazenar todas as classes *Java* que criam os *endpoints* que permitem a interação de usuários com o sistema;
- ***cors***: Este diretório é responsável por armazenar as configurações que permitem algumas origens a fazerem requisições *HTTP* para a aplicação, ou seja, a configuração *CORS*[17] permite que as aplicações *web* possam acessar recursos de fora de seu próprio domínio;
- ***docs***: Este diretório é responsável por armazenar as configurações sobre a documentação do sistema utilizando de uma ferramenta chamada *Swagger* que permite a criação de uma página *web* contendo todos os *endpoints* mapeados, assim facilidando a visualização;
- ***dto***: Aqui estão armazenados todos os *DTO's* do sistema, sendo utilizandos para moldar as solicitações aos *endpoints* e utilizados em alguns retornos dos próprios *endpoints*;
- ***enums***: Este diretório armazena todos os arquivos de enumeração do sistema, como por exemplo, as permissões dos usuários, os tipos de categorias das denúncias e o *status* da denúncia;
- ***models***: Este diretório é responsável por armazenar todas a abstrações do mundo real para dentro do sistema, contendo classes *Java* onde representam essas abstrações, como por exemplo, os usuários que utilizam o sistema;
- ***repositories***: Este diretório armazena configurações relacionadas ao banco de dados, assim fazendo a ligação direta com o mesmo;

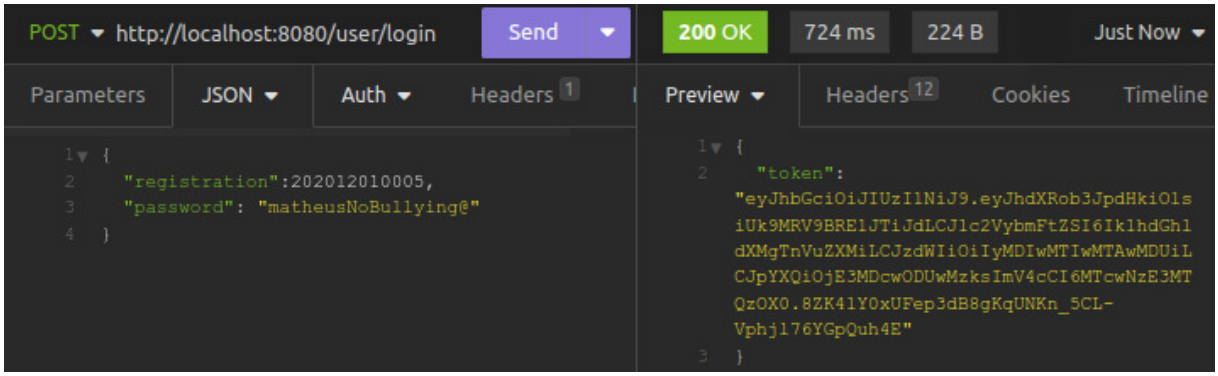
[17] *Cross-Origin Resource Sharing* é um mecanismo que permite que recursos restritos em uma página da web sejam recuperados por outro domínio fora do domínio ao qual pertence o recurso que será recuperado.

- ***service***: Este diretório armazena todos as configurações que dizem respeito as regras de negócio da aplicação;
- ***test***: Aqui são armazenados todos os testes unitários automatizados da aplicação;

5.2 REQUISIÇÕES *HTTP* COM *INSOMNIA*

Ao utilizar a ferramenta de requisições *HTTP* conhecida como *Insomnia* para testar as funcionalidades ativas nos diversos *endpoints* da *NoBullying API*, foi possível realizar uma variedade de requisições. O principal objetivo foi observar o desempenho das funcionalidades implementadas, bem como analisar as respostas obtidas no formato *JSON* com as informações solicitadas. A seguir, observa-se algumas das requisições *HTTP* e respostas encaminhadas aos serviços da *API*:

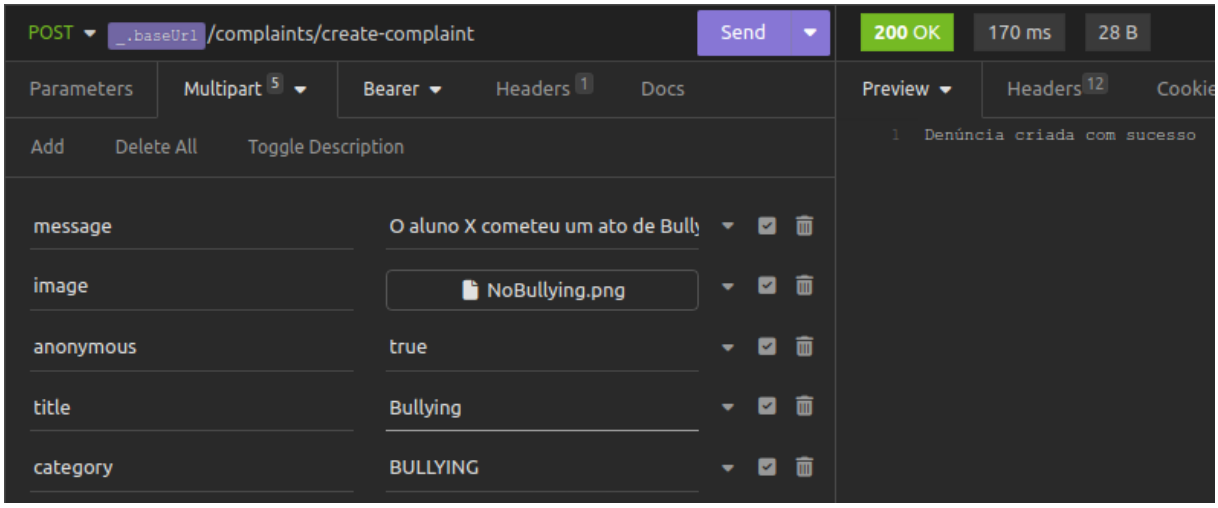
Figura 11 – *Login* com *Insomnia*



Fonte: Elaborado pelo autor (2024).

Observando a Figura 11, pode-se perceber o corpo da requisição e a resposta do *endpoint* de *login*. Esta funcionalidade não tem restrições por usuário, ou seja, independente da permissão que o usuário possuir, ele poderá acessar normalmente este *endpoint*. Para utilizar essa funcionalidade é necessário enviar na requisição um corpo contendo as informações válidas de *registration* (matrícula) e *password* (senha), se as informações estiverem corretas conforme uma pesquisa no banco de dados, a funcionalidade retorna um *token JWT* contendo as permissões do usuário que efetuou o *login* juntamente com outras informações em sua composição.

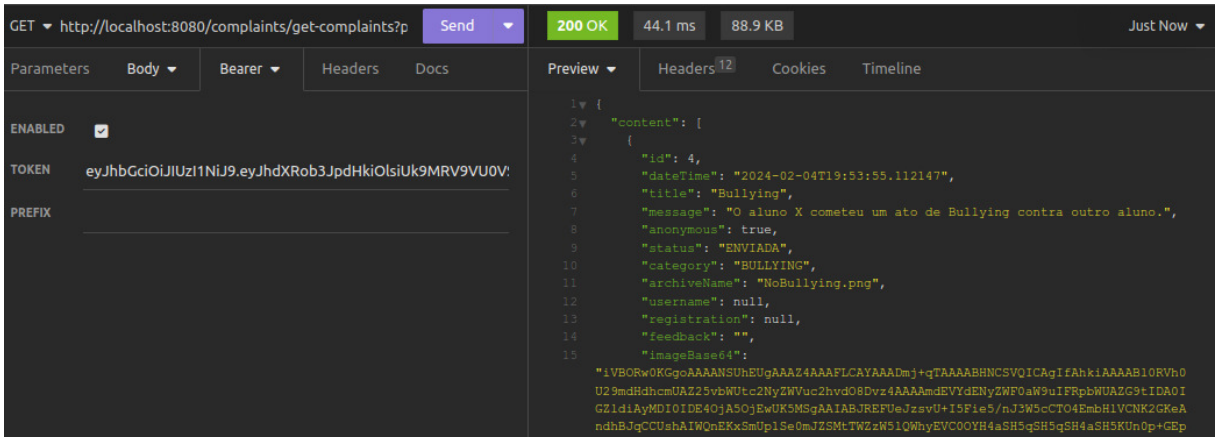
Figura 12 – Denúncia criada a partir do *Insomnia*



Fonte: Elaborado pelo autor (2024).

A Figura 12 representa a criação de uma nova denúncia no sistema da *No-Bullying API*. Observa-se o corpo da requisição, que possui um formato diferente da requisição de *login*, sendo necessário informar alguns campos importantes para a criação de uma nova denúncia, vale ressaltar que todos os *endpoints*, com a exceção de *login*, são seguros e só podem ser acessados por usuários autenticados no sistema, portanto é necessário enviar na requisição o *token JWT* para a autenticação e autorização da funcionalidade solicitada.

Figura 13 – Busca denúncia de usuário com *Insomnia*



Fonte: Elaborado pelo autor (2024).

A Figura 13 representa uma solicitação *HTTP* de busca, onde o *endpoint* retorna todas as denúncias realizadas pelo usuário cujo *token JWT* foi informado, ou

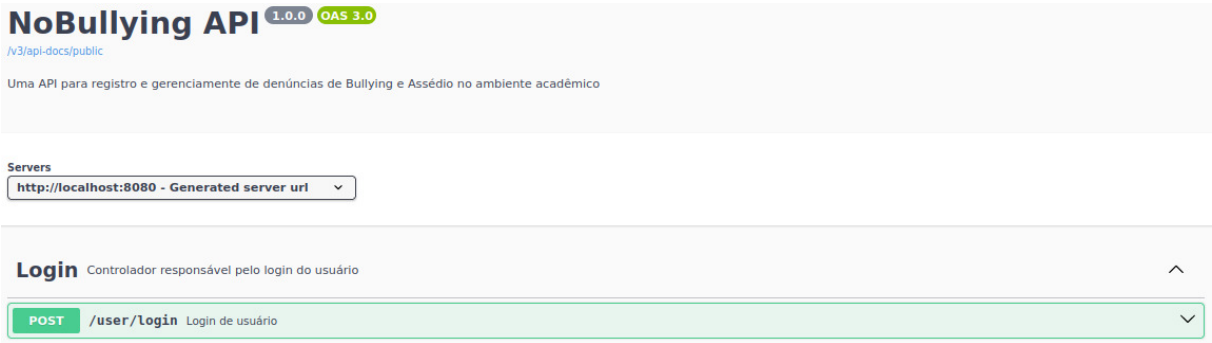
seja, a resposta contendo as denúncias realizadas são retornadas de acordo com o usuárioo que efetuou o *login*. É importante observar que no retorno da funcionalidade, os campos "*username*"(nome do usuário) e "*registration*"(matrícula do usuário) são inserido ou não com base no valor do campo "*anonymous*"(informa se a denúncia é anonima ou não), essa estratégia foi pensada para garantir a segurança dos usuários que realizam denúncias e não querem ser identificados.

5.3 DOCUMENTAÇÃO DA *NOBULLYING API*

A documentação de uma API é uma peça fundamental para facilitar a compreensão e integração eficaz por parte dos desenvolvedores. Neste contexto, a *NoBullying API* adota uma abordagem transparente e acessível para a documentação, utilizando a ferramenta *Swagger*. O *Swagger*[18] não apenas simplifica a visualização das funcionalidades e *endpoints* disponíveis, mas também oferece uma interface interativa, proporcionando uma experiência aprimorada durante o processo de desenvolvimento e integração.

A ilustração da Figura 14 e 15 explanam a documentação dos endpoints públicos e privados, respectivamente, criados com a ferramenta *Swagger* com base nos casos de uso e requisitos funcionais do sistema, supracitados. O detalhamento dos *endpoints* ilustrados na Figura 14 e 15 podem ser encontrados no Apêndice C.

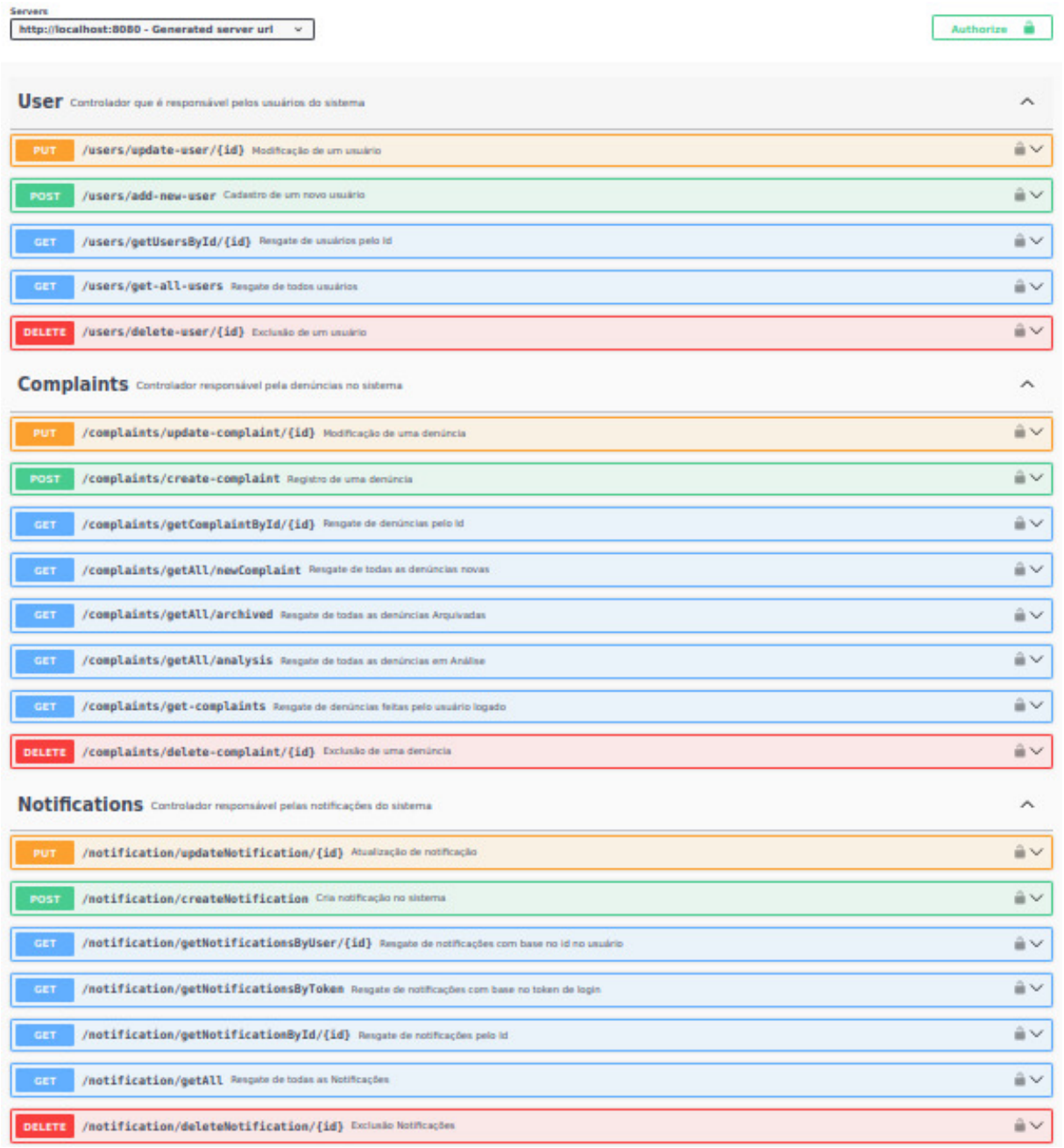
Figura 14 – Documentação de *endpoints* públicos da *NoBullying API*



Fonte: Elaborado pelo autor (2024).

[18] https://swagger.io/

Figura 15 – Documentação de *endpoints* privados da *NoBullying API*



Fonte: Elaborado pelo autor (2024).

6 CONSIDERAÇÕES FINAIS

Considerando o desenvolvimento concluído e a proposta da *NoBullying API*, pode-se afirmar que a implementação de uma *API* voltada para o combate ao bullying é uma iniciativa relevante e promissora. Por meio dessa aplicação, é possível fornecer serviços contendo funcionalidades que auxiliam no enfrentamento e prevenção dessa forma de violência nas escolas. A *NoBullying API* foi construída com base nos requisitos funcionais coletados por meio de entrevistas com membros do NUCA do IFPB Campus Cajazeiras. A implementação de serviços de denúncias anônimas, fornecidos pela *API*, desempenha um papel crucial no incentivo à participação e na proteção da privacidade dos usuários, promovendo, assim, um ambiente de confiança para a denúncia e o combate a essas práticas prejudiciais.

Uma das principais contribuições técnicas do projeto é a adoção da *Clean Architecture*, que proporciona uma estrutura sólida e modular para o desenvolvimento da *API*. Essa arquitetura permite a separação clara das responsabilidades, facilitando a manutenção, escalabilidade e testabilidade do sistema. Outra contribuição técnica muito importante foi a adoção do uso do *Spring Framework*. O *Spring* oferece um conjunto de módulos que abordam diferentes aspectos do desenvolvimento de aplicativos, como persistência de dados, segurança e criação de *APIS RESTful*.

Além disso, a *NoBullying API* representa não apenas uma conquista tecnológica, mas também um compromisso ético e social. Ao oferecer funcionalidades que incentivam a denúncia e promovem a transparência no enfrentamento ao bullying, a aplicação contribui para a construção de uma cultura escolar mais segura e inclusiva. A utilização da arquitetura *Clean Architecture* não só facilitou o desenvolvimento e manutenção do sistema, mas também enfatiza a preocupação com a escalabilidade e a adaptabilidade da *NoBullying API* às necessidades futuras.

Em suma, a conclusão do desenvolvimento da *NoBullying API* não apenas prova sua viabilidade técnica, mas também ressalta seu impacto potencial na promoção de ambientes educacionais mais seguros e saudáveis. O compromisso com a causa, aliado às escolhas arquiteturais e tecnológicas, torna a *NoBullying API* uma contribuição significativa para a abolição do *bullying* e assédio nas instituições de ensino. Este projeto não apenas resulta em uma solução tecnológica eficiente, mas também representa um passo importante em direção à construção de comunidades educacionais mais empáticas e livres de violência.

REFERÊNCIAS

ALMEIDA, K. L.; CAVALCANTE, A.; SILVA, J. S. C. Importância da identificação precoce da ocorrência do bullying: uma revisão de literatura the importance of early identification of bullying: a review of the literature. **Rev Pediatr**, v. 9, n. 1, p. 8–16, 2008.

AQUINO, R. de; MACIEL, R. H.; GONÇALVES, R. C. Prevenção e combate ao assédio moral entre servidores públicos do estado do ceará. **Revista Brasileira de Saúde Ocupacional**, SciELO Brasil, v. 37, p. 243–255, 2012.

AWS, A. W. S. **O que é uma API? – Guia de APIs para iniciantes – AWS**. Disponível em: <https://aws.amazon.com/pt/what-is/api/>.

BARBIERI, J. M. d. O. **Bullying, Assédio moral e mobbing: Você Sabe Diferenciá-Los?** Jus Navigandi, 2019. Disponível em: <https://jus.com.br/artigos/70734/bullying-assedio-moral-e-mobbing>.

BERENGUEL, A. L. A.; QUEIRÓS, L. R.; SOUZA, M. I. F.; ALVES, M. d. D. R. Arquitetura aaa em sistemas web baseados em rest. **Global Science and Technology**, v. 1, n. 1, 2008.

BRAGA, M. O mito do anjo vingador: Influência de columbine na cobertura dos massacres de realengo e suzano do jornal o globo. 2022.

BUENO, C. E. d. O. Desenvolvimento de um aplicativo utilizando o framework flutter e arquitetura limpa. Pontifícia Universidade Católica de Goiás, 2021.

CHALITA, G. **Pedagogia da amizade**. [S.l.]: Editora Gente, 2008.

DUQUE, D. Bullying: a violência invisível. **Revista Dimensão**, v. 49, p. 24–25, 2007.

FANTE, C. **Fenômeno bullying: como prevenir a violência nas escolas e educar para a paz**. [S.l.]: Verus Editora, 2005.

FANTE, C.; PEDRA, J. A. **Bullying escolar: perguntas e respostas**. [S.l.]: Artmed, 2008.

FANTE, C. A. Z. O fenômeno bullying e suas conseqüências psicológicas. **São Paulo**, 2002.

FONSECA, C. D. **O que uma boa API Rest deve ter e porquê?** 2021. Disponível em: <https://dev.to/diariodeumacdf/o-que-uma-boa-api-rest-deve-ter-e-porque-4k14>.

GONÇALVES, B. P.; MENDES, O. L. Princípio da inversão de dependência na qualidade de software: Aplicação da injeção de dependência no desenvolvimento de software. **Revista Interface Tecnológica**, v. 19, n. 1, p. 34–46, 2022.

HIRIGOYEN, M.-F. **Mal-estar no trabalho: redefinindo o assédio moral**. [S.l.]: Bertrand Brasil, 2002.

IFPB. **Eventos de apresentação do NUCA problematizaram os tipos de assédio na escola**. 2019. Disponível em: <https://www.ifpb.edu.br/cajazeiras/noticias/2019/08/eventos-de-apresentacao-do-nuca-problematizaram-os-tipos-de-assedio-na-escola>.

____. **Uma Corrente do Bem no combate ao bullying**. 2019. Disponível em: <https://www.ifpb.edu.br/joaopessoa/noticias/2019/04/um-corrente-do-bem-para-combater-o-bullying#wrapper>.

____. **IFPB**. 2021. Disponível em: <https://www.ifpb.edu.br/prae/rede-de-combate-ao-assedio/legislacao-e-documentos-pertinentes/resolucao-60-2021-consuper-daaoc-reitoria-ifpb.pdf/view>.

LIMA, D. S. de; PEREIRA, R. A. T.; FRANCISCO, M. V. Notas sobre os programas e leis de enfrentamento ao bullying escolar.

MAIA, J. d. S.; JÚNIOR, E. E. d. A. **Concepção e implantação da plataforma bekid: um ambiente para auxílio no combate ao bullying na escola**. Dissertação (B.S. thesis), 2022.

MARCOLINO, E. d. C.; CAVALCANTI, A. L.; PADILHA, W. W. N.; MIRANDA, F. A. N. d.; CLEMENTINO, F. d. S. Bullying: prevalência e fatores associados à vitimização e à agressão no cotidiano escolar. **Texto & Contexto-Enfermagem**, SciELO Brasil, v. 27, p. e5500016, 2018.

MARINHO, D.; SILVA, E.; OLIVEIRA, T.; FAGUNDES, F. Pandora: Uma proposta de combate a violência nas escolas por meio da virtualização de grupos operativos[1]. 10 2018.

MARTIN, R. C. **Arquitetura limpa: o guia do artesão para estrutura e design de software**. [S.l.]: Alta Books Editora, 2019.

MELO, D. **O que é uma API? [Guia para iniciantes]**. 2021. Disponível em: <https://tecnoblog.net/responde/o-que-e-uma-api-guia-para-iniciantes/>.

MOREIRA, F. M. et al. Violência de gênero na escola: abuso/assédio sexual e relações de poder. Florianópolis, SC, 2016.

NORDESTE, D. do. **Menina com deficiência de Aprendizagem Sofre bullying em Escola no RJ e caso Gera Comoção na web**. Diário do Nordeste, 2023. Disponível em: <https://diariodonordeste.verdesmares.com.br/ultima-hora/pais/menina-autista-sofre-bullying-em-escola-no-rj-e-caso-gera-comocao-na-web-1.3352973>.

RUSSO, L. X. Associação entre vitimização por bullying e índice de massa corporal em escolares. **Cadernos de Saúde Pública**, SciELO Public Health, v. 36, p. e00182819, 2020.

SAMPAIO, J. **Clean Architecture: descubra o que é e onde aplicar Arquitetura Limpa**. 2021. Disponível em: <https://www.zup.com.br/blog/clean-architecture-arquitetura-limpa>.

SANTOS, M. M.; KIENEN, N. Características do bullying na percepção de alunos e professores de uma escola de ensino fundamental. **Temas em psicologia**, Sociedade Brasileira de Psicologia, v. 22, n. 1, p. 161–178, 2014.

SANTOS, M. M.; PERKOSKI, I. R.; KIENEN, N. Bullying: atitudes, consequências e medidas preventivas na percepção de professores e alunos do ensino fundamental. **Temas em Psicologia**, Sociedade Brasileira de Psicologia, v. 23, n. 4, p. 1017–1033, 2015.

SCHULTZ, N. C. W.; DUQUE, D. F.; SILVA, C. F. d.; SOUZA, C. D. d.; ASSINI, L. C.; CARNEIRO, M. d. G. d. M. A compreensão sistêmica do bullying. **Psicologia em Estudo**, SciELO Brasil, v. 17, p. 247–254, 2012.

SOUSA, I. de. **Saiba o que é REST (Representational State Transfer) e como usá-lo**. 2020. Disponível em: <https://rockcontent.com/br/blog/rest/>.

APÊNDICE A – *USER STORIES*

O *User Stories* no desenvolvimento de um sistema é uma prática ágil que busca abordar as necessidades e perspectivas do usuário final. No contexto da *NoBullying API*, os *user stories* desempenham um importante papel na definição dos requisitos e funcionalidades, ao representar as diversas tarefas e objetivos que os alunos e membros do NUCA almejam alcançar ao utilizar a aplicação. Ao criar essas histórias, é possível compreender melhor as expectativas dos usuários, identificar os recursos necessários e priorizar as funcionalidades mais relevantes.

1. Como usuário, eu quero poder criar uma denúncia anônima de *bullying* ou assédio sem precisar me locomover até o departamento do NUCA;
2. Como usuário, eu quero poder observar as denúncias que eu mesmo criei;
3. Como usuário, eu quero ter um retorno e notificações sobre as denúncias que eu realizei;
4. Como usuário, eu quero poder editar uma denúncia feita para o caso de adicionar mais provas a respeito da denúncia ou correção de textos;
5. Como membro do NUCA, eu quero poder ter acesso a todas as denúncias realizadas no sistema;
6. Como membro do NUCA, eu quero poder adicionar novos usuários no sistema e atribuí-los permissões que forem necessárias;
7. Como membro do NUCA, eu quero poder ter acesso a todos os usuários registrados no sistema;
8. Como membro do NUCA, eu quero poder modificar o status de progresso de uma denúncia;
9. Como membro do NUCA, eu quero poder enviar um retorno e notificações sobre as medidas tomadas a respeito de uma denúncia específica;

APÊNDICE B – ARQUITETURA DO BANCO DE DADOS DO SISTEMA

Descrição do Mini-Mundo

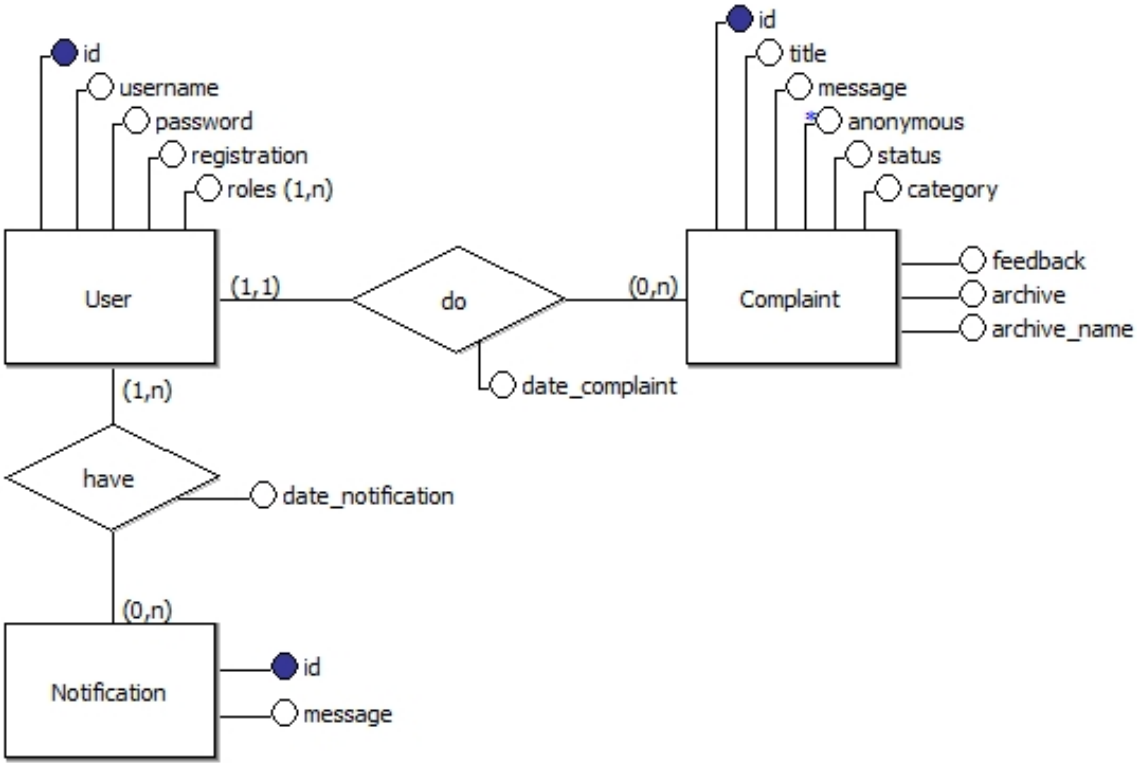
A base de dados irá armazenar informações a respeito de denúncias de *bullying* e assédio. Essas informações serão divididas em três entidades do banco de dados relacional, são elas:

1. *User* (Usuário): Será responsável por armazenar dados sobre os usuários do sistema. Esses dados são: *id* (identificador), *username* (nome), *registration* (matrícula), *password* (senha) e *roles* (representam as permissões que o usuário possui no sistema);
2. *Complaint* (Denúncia): Será responsável por armazenar dados sobre as denúncias realizadas por cada usuário. Esses dados são: *id* (identificador), *message* (descrição do ocorrido), *title* (título da denúncia), *anonymous* (campo de escolha do usuário se a denúncia será anônima ou não), *date_complaint* (data da denúncia), *status* (campo que representa a situação de revisão da denúncia, modificado apenas por administradores), *archive* (arquivo de prova), *archive_name* (nome do arquivo), *category* (categoria da denúncia) e *feedback* (retorno dos administradores a respeito da denúncia feita);
3. *Notification* (Notificações): Será responsável por armazenar dados sobre as notificações que os usuário pode receber dos usuários administradores. Esses dados são: *id* (identificador), *message* (representa a mensagem que o usuário administrador pode enviar) e *date_notification* (representa a data que a notificação foi criada);

A entidade *User* deve possuir um relacionamento com a entidade *Complaint* de um para muitos(1,N ou *One To Many*), onde cada usuário pode realizar muitas instâncias da entidade *Complaint*. A entidade *User* deve possuir também um relacionamento de um para muitos com a entidade *Notification*, onde cada usuário podem possuir uma ou muitas notificações atreladas a ele.

Modelo Entidade Relacionamento (MER)

Figura 16 – Diagrama do Modelo Entidade Relacionamento



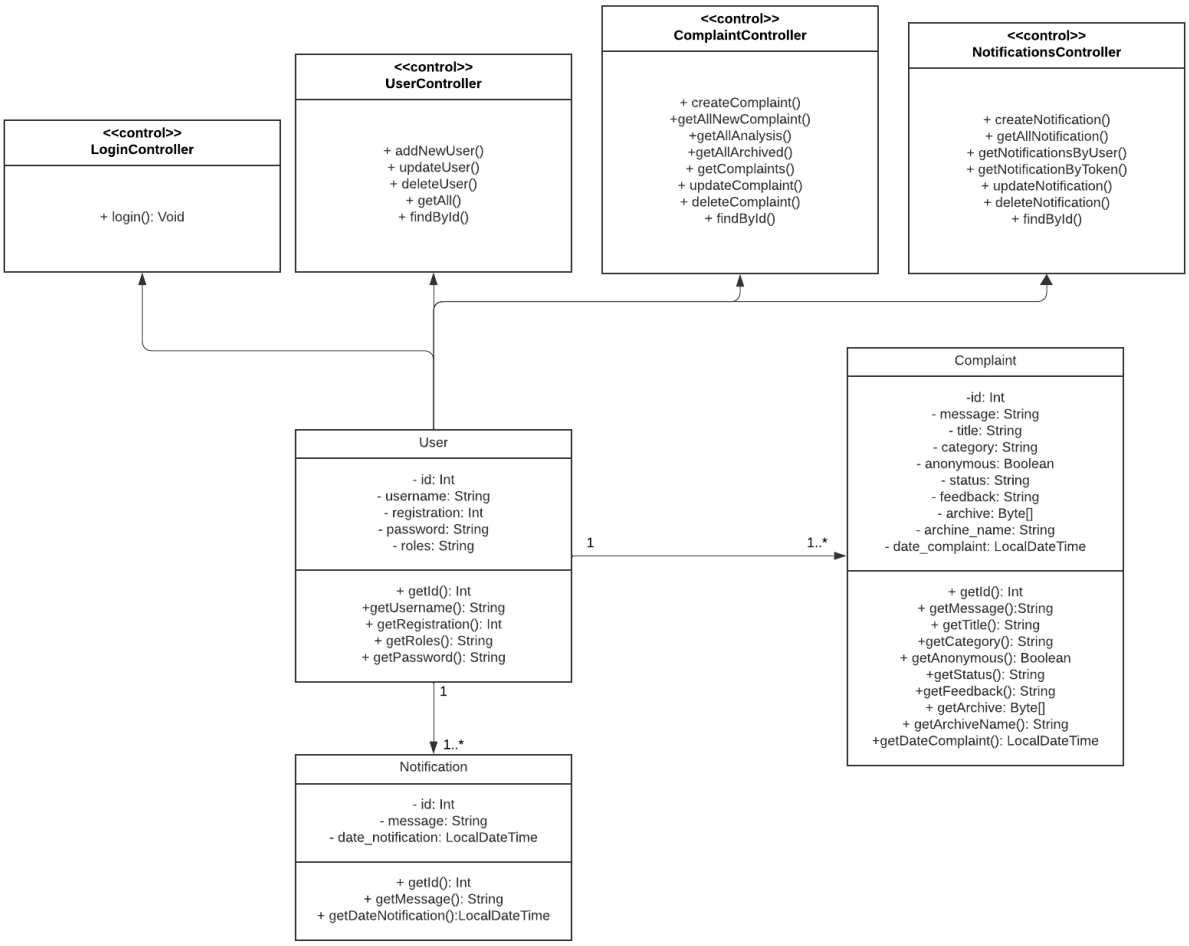
Fonte: Elaborado pelo autor (2023).

A Figura 16 representa o Modelo Entidade Relacionamento que foi utilizado para a criação da estrutura de armazenamento de dados com base na descrição do Mini-Mundo.

Diagrama de Classes

Um diagrama de classes é uma representação visual das classes de um sistema e das relações entre elas. O diagrama de classes concede uma visão geral da arquitetura do sistema, permitindo uma compreensão clara da *API*. Nesta seção será apresentado o diagrama de classes da *NoBullying API*, que representa a estrutura e interações entre as principais entidades e controladores.

Figura 17 – Diagrama de Classes *UML*



Fonte: Elaborado pelo autor (2023).

A Figura 17 representa o diagrama de classes da *NoBullying API*. Este diagrama ilustra todas as classes presentes na *API*, juntamente com seus atributos correspondentes:

1. *User*: Representa a entidade do usuário, contendo seus atributos como: *id* (identificador), *username* (nome do usuário), *registration* (número da matrícula do usuário), *password* (senha do usuário) e *roles* (permissões que o usuário possui no sistema). Esta classe usuário possui associação simples com as classes “*LoginController*”, “*UserController*”, “*ComplaintController*” e "*NotificationsController*", essas classes possuem funções que o usuário pode utilizar, essas funções são: *login*, *CRUD* de usuário (funções de *create*, *delete* e *update* são exclusivas de administradores), *CRUD* de denúncias (funções de *delete* são exclusivas de administradores) e *CRUD* de notificações (todas as funções são exclusivas de usuários administradores);

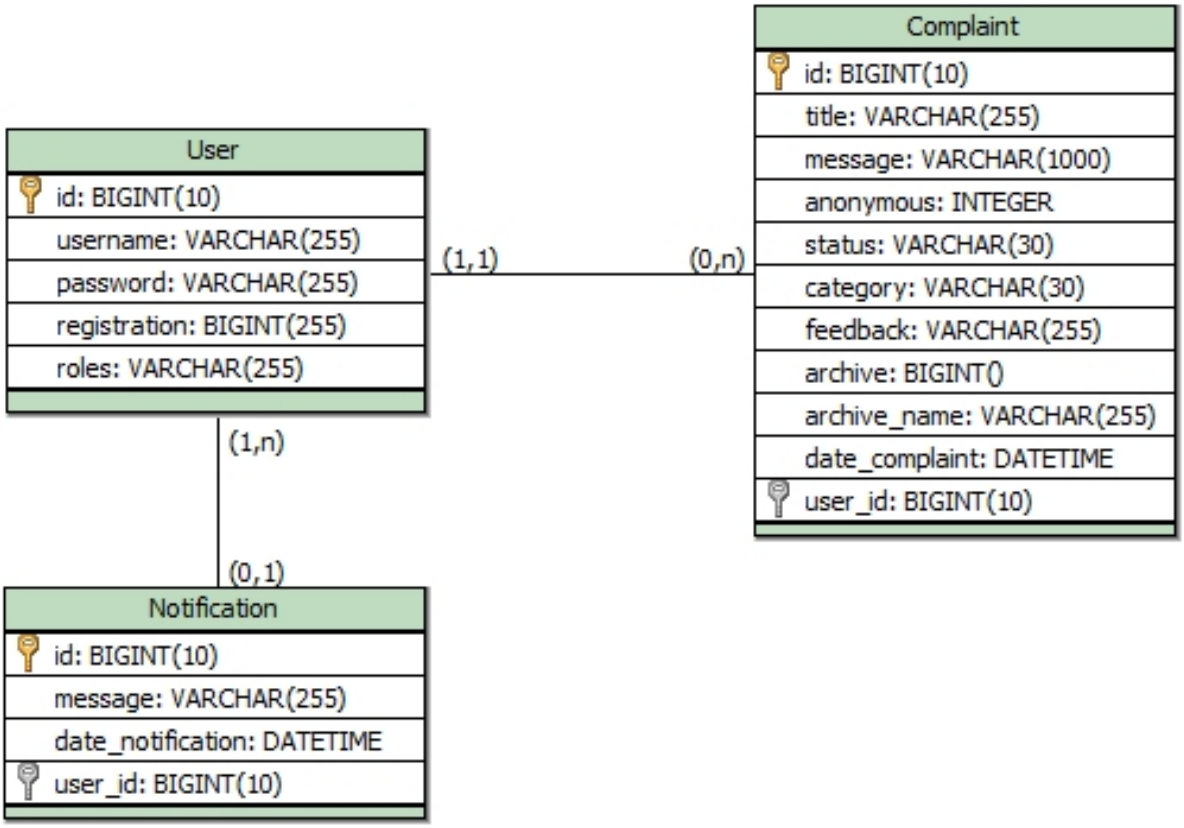
2. *Complaint*: Representa a entidade da denúncia, contendo seus atributos como: *id* (identificador), *message* (descrição do ocorrido), *title* (título da denúncia), *category* (categoria da denúncia), *anonymous* (anonimato de escolha do usuário), *status* (informação sobre a situação da denúncia), *feedback* (retorno dos administradores sobre a denúncia), *archive* (arquivo enviado como prova), *archive_name* (nome do arquivo) e *date_complaint* (data da denúncia);

3. *Notification*: Representa a entidade de notificações, ou seja, as notificações criadas pelos usuários administradores a respeito das denúncias e enviadas para os usuários comuns. Essa entidade possui atributos como: *id* (identificador), *message* (mensagem de conteúdo da notificação) e *date_notification* (data em que a notificação foi criada);

Modelo Lógico

A representação lógica do banco de dados apresenta um modelo mais próximo da representação física do banco de dados.

Figura 18 – Modelo lógico do banco de dados



Fonte: Elaborado pelo autor (2023).

A Figura 18 apresenta o modelo lógico da base de dados que será responsável por armazenar os dados da aplicação e consiste na representação de todas as tabelas a serem criadas na base de dados juntamente com os atributos que são gerados a partir de relacionamentos entre entidades.

Modelo Físico

Esta seção apresenta o modelo físico da base de dados gerado a partir do modelo lógico que se encontra na seção anterior. O código gerado segue os comandos e regras utilizados pelo *PostgreSQL*, uma vez que é a ferramenta utilizada na aplicação para a permanência dos dados.

Algoritmo 1 – Código *SQL* do banco de dados

```
CREATE TABLE tb_User (
  id BIGINT PRIMARY KEY,
  username VARCHAR(255) NOT NULL,
  password VARCHAR(255) NOT NULL,
  registration BIGINT NOT NULL,
  roles VARCHAR(255) NOT NULL
);

CREATE TABLE Complaint (
  id BIGINT PRIMARY KEY,
  title VARCHAR(255) NOT NULL,
  message VARCHAR(1000) NOT NULL,
  anonymous INTEGER NOT NULL,
  status VARCHAR(30) NOT NULL,
  category VARCHAR(30) NOT NULL,
  feedback VARCHAR(255) NOT NULL,
  archive BIGINT,
  archive_name VARCHAR(255),
  date_complaint TIMESTAMP NOT NULL,
  user_id BIGINT,
  FOREIGN KEY (user_id) REFERENCES tb_User(id)
);

CREATE TABLE Notification (
  id BIGINT PRIMARY KEY,
  message VARCHAR(255) NOT NULL,
  date_notification TIMESTAMP NOT NULL,
  user_id BIGINT,
  FOREIGN KEY (user_id) REFERENCES tb_User(id)
);
```

Fonte: Elaborado pelo autor (2023)

APÊNDICE C – DOCUMENTAÇÃO DOS *ENDPOINTS*

O Quadro 3 apresenta a documentação para os *endpoints* responsáveis pelos usuários registrados no sistema, todos os *endpoints* são protegidos e só podem ser acessados por um usuário que tenha o *token JWT* com as permissões de administradores.

Quadro 3 – *UserController*

User-Controller			
Método *HTTP*	***Endpoint***	**Parâmetros**	**Descrição**
PUT	*/users/update-user/{id}*	*id: Integer* *JSON representado pelo Algoritmo 2.*	Esse *endpoint* é responsável por atualizar um usuário já registrado no sistema.
POST	*/users/add-new-user*	*JSON* representado pelo Algoritmo 2.	Esse *endpoint* é responsável por registrar um novo usuário no sistema.
GET	*/users/get-all-users*	*page?: Integer* *size?: Integer*	Esse *endpoint* é responsável por resgatar todos os usuários cadastrados no sistema.
GET	*/users/getUsersById/{id}*	*id: Integer*	Esse *endpoint* é responsável por resgatar um usuário pelo número de identificação.
DELETE	*/users/delete-user/{id}*	*id: Integer*	Esse *endpoint* é responsável por deletar um usuário do sistema.

Fonte: Elaborado pelo autor (2024).

Observando o Quadro 3, os *endpoints* de cujos métodos *HTTP* são *PUT* e *POST* precisam do parâmetro representado no Algoritmo 2, que simboliza o corpo da requisição *HTTP* necessária para o funcionamento correto dos métodos. Já o *endpoint* cujo método *HTTP* é *GET*, possui dois parâmetros que são opcionais e que podem ser informados como "*query params*"[1] no corpo do *endpoint*, por padrão o parâmetro

[1] *Query Params* são parâmetros da *URL* que tem o objetivo extender informações de um seguimento

“*page*” possui o valor zero e o parâmetro “*size*” possui o valor dez.

Quadro 4 – *ComplaintController*

Complaint-Controller			
Método *HTTP*	***Endpoint***	**Parâmetros**	**Descrição**
PUT	*/complaints /update-complaint/{id}*	*id: Integer JSON representado pelo Algoritmo 3.*	Esse *endpoint* é responsável por atualizar uma denúncia registrada no sistema.
POST	*/complaints /create-complaint*	*JSON* representado pelo Algoritmo 4.	Esse *endpoint* é responsável por registrar uma nova denúncia no sistema.
GET	*/complaints /getAll/newComplaint*	*page?: Integer size?: Integer*	Esse *endpoint* é responsável por resgatar todos as novas denúncias registradas no sistema.
GET	*/complaints /getAll/analysis*	*page?: Integer size?: Integer*	Esse *endpoint* é responsável por resgatar todos as denúncias em análise registradas no sistema.
GET	*/complaints /getAll/archived*	*page?: Integer size?: Integer*	Esse *endpoint* é responsável por resgatar todos as denúncias arquivadas registradas no sistema.
GET	*/complaints /getComplaintById/{id}*	*id: Integer*	Esse *endpoint* é responsável por resgatar uma denúncia pelo número de identificação.
GET	*/complaints /get-complaints*	*page?: Integer size?: Integer*	Esse *endpoint* é responsável por resgatar as denúncias do usuário logado.
DELETE	*/complaints /delete-complaint/{id}*	*id: Integer*	Esse *endpoint* é responsável por deletar uma denúncia do sistema.

Fonte: Elaborado pelo autor (2024).

da *URL* afim de adicionar informações extras

O Quadro 4 apresenta a documentação dos *endpoints* responsáveis pelas denúncias no sistema, todos esses *endpoints* são protegidos. Usuários com as permissões de usuários comuns podem ter acesso somente aos *endpoints* cujos métodos *HTTP* são *POST*, o método *GET* de *endpoint* "*/complaints/get-complaints*", e o método *PUT*. Os demais métodos são exclusivos para usuários com permissões de administradores.

Quadro 5 – *NotificationsController*

Notifications-Controller			
Método *HTTP*	***Endpoint***	**Parâmetros**	**Descrição**
PUT	*/notificaiton /updateNotification/{id}*	*id: Integer JSON representado pelo Algoritmo*	Esse *endpoint* é responsável por atualizar uma notificação registrada no sistema.
POST	*/notification /createNotification*	*JSON* representado pelo Algoritmo	Esse *endpoint* é responsável por registrar uma nova notificação no sistema.
GET	*/notification /getNotificationsByUser /{id}*	*page?: Integer size?: Integer*	Esse *endpoint* é responsável por resgatar todas as notificações de um determinado usuário.
GET	*/notification /getNotificationsByToken*	*page?: Integer size?: Integer*	Esse *endpoint* é responsável por resgatar todas as notificações do usuário logado.
GET	*/notification /getNotificationById/{id}*	*id: Integer*	Esse *endpoint* é responsável por resgatar uma notificação com base no número de identificação.
GET	*/notification/getAll*	*page?: Integer size?: Integer*	Esse *endpoint* é responsável por resgatar todas as notificações registradas no sistema.
DELETE	*/notification /deleteNotification/{id}*	*id: Integer*	Esse *endpoint* é responsável por deletar uma notificação do sistema.

Fonte: Elaborado pelo autor (2024).

O Quadro 5 apresenta a documentação dos *endpoints* responsáveis pelas notificações do sistema. Todos esses *endpoints* são exclusivos para usuários com permissões de administradores, com exceção do método *GET* de *endpoint* "*/notification/getNotificationByUser*".

Quadro 6 – *LoginController*

Login-Controller			
Método *HTTP*	***Endpoint***	**Parâmetros**	**Descrição**
POST	*/user/login*	*JSON* representado pelo Algoritmo	Esse *endpoint* é responsável pela autenticação dos usuários no sistema.

Fonte: Elaborado pelo autor (2024).

O Quadro 6 apresenta a documentação do *endpoint* responsável pela autenticação dos usuários. Esse *endpoint* não é protegido por autenticação nenhuma e pode ser acessado por todos os usuários do sistema.

Algoritmo 2 – Corpo da Requisição *POST* e *PUT* no formato *JSON* para o *UserController*

```
{
    "username": "String",
    "password":"Number",
    "registration": "Number",
    "roles":[
        "String"
    ]
}
```

Fonte: Elaborado pelo autor (2024)

O Algoritmo 2 descreve o corpo das requisições associadas aos métodos *HTTP POST* e *PUT*. Este corpo de requisição é utilizado tanto para a adição de um novo usuário ao sistema quanto para a atualização de um usuário já existente. Os *endpoints* correspondentes são destinados exclusivamente a usuários com privilégios administrativos.

Algoritmo 3 – Corpo da Requisição *PUT* no formato *JSON* para o *ComplaintController*

```
{
    "message": "String",
    "title":"String",
    "category": "String",
    "anonymous": "Boolean",
    "image": "Byte",
    "status": "String",
    "feedback": "String"
}
```

Fonte: Elaborado pelo autor (2024)

O Algoritmo 3 descreve o conteúdo da requisição associada ao método *HTTP PUT*. Este corpo de requisição inclui propriedades adicionais que apenas usuários administradores têm permissão para modificar, sendo utilizado para a atualização de uma denúncia já existente no sistema. Esse corpo de requisição precisa estar no formato *"multipart/form-data"*[2]

Algoritmo 4 – Corpo da Requisição *POST* no formato *JSON* para o *ComplaintController*

```
{
    "message": "String",
    "title":"String",
    "category": "String",
    "anonymous": "Boolean",
    "image": "Byte"
}
```

Fonte: Elaborado pelo autor (2024)

O Algoritmo 4 descreve o conteúdo da requisição associada ao método *HTTP POST*. Este corpo de requisição é utilizada para adicionar uma nova denúncia no sistema. De acordo com o Algoritmo 3, essa requisição tabmém precisa estar no formato *multipart/form-data*.

[2] O formato *multipart/form-data* é um tipo de codificação de dados utilizado em requisições HTTP para o envio de arquivos e dados binários.

Algoritmo 5 – Corpo da Requisição *PUT* e *POST* no formato *JSON* para o *NotificationsController*

```
{
    "message": "String",
    "userId": {
        "id": "Number"
    }
}
```

Fonte: Elaborado pelo autor (2024)

O Algoritmo 5 contém o corpo da requisição associada aos métodos *HTTP POST* e *PUT*. Este corpo de requisição é utilizado para adicionar uma nova notificação no sistema e atualizar uma notificação já existente.

Algoritmo 6 – Corpo da Requisição *POST* no formato *JSON* para o *LoginController*

```
{
    "registration": "Number",
    "password": "String"
}
```

Fonte: Elaborado pelo autor (2024)

O Algoritmo 6 descreve o conteúdo da requisição relacionada ao método *HTTP POST*. Esse corpo é empregado para efetuar o processo de *login* na aplicação.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DA PARAÍBA
Campus Cajazeiras - Código INEP: 25008978
Rua José Antônio da Silva, 300, Jardim Oásis, CEP 58.900-000, Cajazeiras (PB)
CNPJ: 10.783.898/0005-07 - Telefone: (83) 3532-4100

Documento Digitalizado Ostensivo (Público)

Entrega de TCC

Assunto:	Entrega de TCC
Assinado por:	Matheus Miguel
Tipo do Documento:	Anexo
Situação:	Finalizado
Nível de Acesso:	Ostensivo (Público)
Tipo do Conferência:	Cópia Simples

Documento assinado eletronicamente por:

- **Matheus Nunes Miguel**, **ALUNO (202012010005) DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS - CAJAZEIRAS**, em 08/03/2024 21:22:37.

Este documento foi armazenado no SUAP em 08/03/2024. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifpb.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 1109250
Código de Autenticação: 1956fc2e98