Desenvolvimento de uma Solução para Identificação de Assédio Moral em Dados Conversacionais Ubíquos

Gabriel Valentim[1]

[1]Universidade Federal de Santa Maria (UFSM)
97105-900 – Santa Maria – RS – Brazil

Abstract. *This study presents the development and evaluation of a moral harassment detection system focusing on mobile and pervasive computing, leveraging artificial intelligence, textual similarity analysis, and ubiquitous data generated from recorded audio. Implemented as a mobile application, the system allows users to record audio and identify inappropriate behaviors using models like Mistral AI and Cohere, while integrating a collaborative database that evolves with user contributions. Tests conducted ranged from simple phrases to complex dialogues and colloquial expressions, demonstrating the hybrid approach's effectiveness in capturing cultural and linguistic nuances. By combining advanced technologies and user participation, the system adaptively identifies moral harassment, enhancing detection accuracy and continuous learning. This work underscores the potential of mobile devices and pervasive systems to monitor daily interactions in real-time, contributing to moral harassment prevention, fostering ethical environments, and advancing the innovative use of ubiquitous data for social well-being.*

Resumo. *Este trabalho apresenta o desenvolvimento e avaliação de um sistema de detecção de assédio moral, com foco em computação móvel e pervasiva, utilizando inteligência artificial, análise de similaridade textual e dados ubíquos gerados por áudios gravados. Implementado em um aplicativo, o sistema permite gravar áudios e identificar comportamentos inadequados por meio de modelos como Mistral AI e Cohere, além de integrar uma base de dados colaborativa que evolui com contribuições dos usuários. Os testes realizados avaliaram desde frases simples até diálogos complexos e expressões coloquiais, demonstrando a eficácia da abordagem híbrida para capturar nuances culturais e linguísticas. A combinação de tecnologias avançadas e participação dos usuários possibilitou identificar assédio moral de forma adaptável, promovendo maior precisão e aprendizado contínuo. Este estudo destaca o potencial de dispositivos móveis e sistemas pervasivos para monitorar interações cotidianas em tempo real, contribuindo para a prevenção de assédio moral e a criação de ambientes mais éticos, ao mesmo tempo em que fomenta a inovação no uso de dados ubíquos para o bem-estar social.*

1. Introdução

A evolução constante das tecnologias de conectividade, incluindo conexões remotas e sem fio, tem transformado significativamente o desenvolvimento e o uso de dispositivos ubíquos, como cartões inteligentes, sensores e outros. Esses dispositivos permeiam diversas áreas da sociedade, desempenhando papéis fundamentais na vida cotidiana e no avanço de soluções tecnológicas inovadoras [Barros 2008].

Historicamente, a computação apresentou três grandes tendências que moldaram o relacionamento entre usuários e tecnologias. A primeira tendência, conhecida como mainframe, caracteriza-se pelo modelo ”um computador, vários usuários”. Em seguida, a era dos computadores pessoais consolidou-se com o conceito ”um usuário, um computador”. A terceira tendência surgiu com o avanço da internet, destacando o paradigma ”um usuário, muitos computadores”. Essa última tendência, que se alinha à computação distribuída e móvel, está intimamente conectada ao conceito de Computação Ubíqua [Santos 2014]. Nesse contexto, a presença de centenas de computadores em uma sala, que inicialmente poderia parecer intimidadora, tende a tornar-se invisível à percepção consciente, assim como os fios elétricos ocultos nas paredes de um ambiente. A ubiquidade computacional busca atingir essa naturalidade, integrando a tecnologia de forma imperceptível às atividades humanas cotidianas [Weiser 1995].

A computação ubíqua enfatiza a utilização de ferramentas ”invisíveis”, ou seja, dispositivos que realizam suas funções de forma eficiente sem demandar a atenção total do usuário. Exemplos disso incluem óculos que, embora posicionados no rosto, não desviam o foco visual de quem os utiliza, e dispositivos de captação de áudio que podem registrar conversas em ambientes hostis para análise de dados posteriormente [Weiser 1994].

Embora tais dispositivos utilizem diferentes mídias — como som, vídeo, texto e gráficos —, seu objetivo não é agir como ”dispositivos multimídia”. Ao contrário das máquinas multimídia tradicionais, que exigem a atenção direta do usuário, os dispositivos ubíquos são projetados para desaparecer no plano de fundo, permitindo que os usuários se concentrem nas tarefas essenciais e na interação com o ambiente [Weiser 1995].

1.1. Problema

O crescente uso de dispositivos ubíquos em ambientes cotidianos tem ampliado as possibilidades de monitoramento e análise das interações humanas, incluindo aquelas que ocorrem em contextos adversos, como o assédio moral. Apesar dos avanços significativos proporcionados pela computação ubíqua, ainda existem lacunas importantes a serem exploradas para aprimorar as tecnologias de detecção de comportamentos abusivos, especialmente no campo da Interação Humano-Computador. Este estudo busca investigar como dispositivos ubíquos, particularmente aqueles baseados em captação de áudio e análise de dados, podem ser empregados na identificação de práticas prejudiciais, como o assédio moral, durante as interações mediadas por tecnologia. O assédio moral, caracterizado por condutas abusivas e repetitivas que afetam a integridade psíquica, social e/ou física de uma pessoa [DE SOUZA 2018], pode ocorrer em diversos ambientes sociais, sendo especialmente preocupante no contexto laboral e educacional, onde compromete o bem-estar dos indivíduos e desestabiliza os ambientes de trabalho e de ensino.

1.2. Objeto

Este estudo tem como objetivo o desenvolvimento de um sistema inteligente para monitoramento e detecção precisa de situações de assédio moral em dados gerados pela interação Humano-Computador, especificamente por dispositivos ubíquos em ambientes de conversa. Utilizando computação pervasiva e técnicas de processamento de linguagem natural (PLN), o sistema analisará as interações, com foco na identificação de padrões de comportamento que possam indicar práticas de assédio. A proposta envolve o desenvolvimento e a implementação de sistemas centrados no usuário, integrados a dispositivos portáteis, como smartphones, que captam conversas de maneira eficiente, com o objetivo de fornecer uma experiência personalizada de monitoramento e detecção.

A análise será conduzida com base em metodologias de inteligência artificial e aprendizado de máquina, permitindo a identificação, classificação e fornecimento de feedback sobre os dados coletados, a fim de identificar comportamentos de assédio moral de forma automatizada e precisa.

1.3. Objetivo

O objetivo deste trabalho é o desenvolvimento de um sistema inteligente para a detecção de assédio moral em dados gerados por dispositivos ubíquos, especificamente em ambientes de conversa mediados pela interação Humano-Computador. O foco principal estará no processamento e análise dos dados de áudio capturados, utilizando metodologias avançadas de Inteligência Artificial e aprendizado de máquina. Para a geração dos áudios, será criado um aplicativo que utilizará o microfone integrado dos smartphones. Este objetivo será desdobrado nas seguintes metas:

1. Desenvolver um sistema composto por duas APIs REST para o tratamento e análise dos dados de áudio gerados em interações Humano-Computador.
2. Implementar uma detecção de assédio moral utilizando inteligências artificiais amplamente disponíveis no mercado.
3. Criar uma detecção baseada em consultas SQL, utilizando dados pré-estabelecidos armazenados no banco de dados.
4. Desenvolver um método de detecção por similaridade, comparando novos dados com informações previamente armazenadas no banco de dados.
5. Implementar um sistema de aprendizado baseado no feedback dos usuários, para aprimorar a detecção de assédio moral.
6. Analisar as características dos dados gerados por dispositivos ubíquos, com ênfase nas interações de áudio, para identificar variáveis relevantes na detecção do assédio moral.
7. Desenvolver um aplicativo que funcione como cliente para a captura e gravação de arquivos de áudio, permitindo a interação com o sistema de detecção de assédio.

1.4. Justificativa

A crescente adoção de tecnologias de computação pervasiva e a evolução das práticas de Interação Humano-Computador oferecem uma oportunidade única para utilizar os dados gerados por dispositivos ubíquos de maneira inteligente e eficaz. Esses dados podem ser fundamentais para criar ambientes mais seguros, inclusivos e respeitosos, permitindo, por exemplo, o desenvolvimento de sistemas centrados no usuário capazes de identificar

comportamentos prejudiciais, como o assédio moral. A implementação de uma solução baseada em aprendizado de máquina para detectar esses comportamentos é uma inovação significativa, com potencial para promover um impacto positivo na sociedade, ao proporcionar um ambiente mais saudável, tanto em contextos profissionais quanto educacionais.

O assédio moral, especialmente em instituições de ensino superior, é um fenômeno que, apesar de suas repercussões significativas, ainda não recebe a atenção devida na literatura acadêmica. Caracterizado por práticas abusivas e repetitivas, com o objetivo de humilhar e desestabilizar psicologicamente a vítima, o assédio moral pode resultar em impactos devastadores, tanto na vida pessoal dos envolvidos quanto no ambiente de trabalho e no contexto educacional. No âmbito educacional, o assédio moral compromete o bem-estar de docentes e discentes, prejudicando a dinâmica organizacional e afetando diretamente a produtividade e a qualidade do ensino. Assim, é fundamental que o assédio moral nas instituições de ensino seja discutido de forma mais aprofundada, com a exploração de soluções tecnológicas para sua detecção e prevenção [DE SOUZA 2018].

2. Metodologia

A metodologia adotada para esta pesquisa segue uma abordagem prática e aplicada, com ênfase na Interação Humano-Computador (IHC) e nos Sistemas Centrados no Usuário. O foco principal é o desenvolvimento de uma aplicação móvel voltada para a detecção de assédio moral em interações baseadas em áudio, geradas por dispositivos ubíquos. O processo envolve a criação de um aplicativo mobile, utilizando as tecnologias TypeScript, React Native e Expo, permitindo a gravação e o envio de dados de áudio para posterior análise. O sistema de backend será composto por duas REST APIs: uma construída com TypeScript e Nest.js e outra com Python, para processamento e análise dos dados capturados.

2.1. Procedimentos e Recursos

Nesta seção, são detalhados os procedimentos adotados para o desenvolvimento do aplicativo e do sistema de análise, assim como os recursos tecnológicos empregados. O processo de detecção de assédio moral é centrado nas interações usuário-dispositivo, com o objetivo de identificar comportamentos abusivos por meio da análise das gravações de áudio realizadas pelos usuários. O enfoque está em compreender e interpretar padrões de comunicação que possam indicar assédio moral, considerando a dinâmica da interação humano-computador e como os dispositivos ubíquos podem ser usados de forma invisível, mas eficaz, para o monitoramento e detecção de comportamentos inadequados.

2.1.1. Desenvolvimento do Sistema

O desenvolvimento do sistema foi projetado para lidar com dados ubíquos em formato de áudio, alinhando-se aos princípios da computação pervasiva e Interação Humano-Computador (IHC). O backend da aplicação é composto por duas APIs REST, cada uma desempenhando um papel específico no processamento e análise dos dados:

1. API 1: Desenvolvida em TypeScript utilizando o framework Nest.js, em conjunto com o Prisma ORM, garantindo uma estrutura robusta para o gerenciamento de dados e integração com o banco de dados PostgreSQL.

2. API 2: Implementada em Python com a biblioteca FastAPI, voltada para o processamento dos áudios e análises mais complexas, incluindo a integração com modelos de aprendizado de máquina.

O cliente responsável pela coleta de dados, ou seja, a gravação de conversas que gerarão os áudios para análise, será desenvolvido como um aplicativo móvel utilizando as tecnologias TypeScript, React Native e Expo. A interface deste aplicativo será projetada com base nos princípios de Sistemas Centrados no Usuário, garantindo facilidade de uso e uma experiência amigável durante a interação.

2.1.2. Fontes de Informação

A construção do sistema baseia-se em fontes confiáveis que suportam o desenvolvimento tecnológico e o embasamento teórico da pesquisa. Essas fontes incluem:

- Documentação técnica: Manuais e guias das tecnologias empregadas, como TypeScript, Nest.js, Prisma, Python, FastAPI, PostgreSQL, React Native e Expo, que fornecem informações detalhadas sobre suas funcionalidades e melhores práticas.
- Estudos acadêmicos: Artigos científicos e estudos de caso relacionados ao desenvolvimento de aplicativos, computação pervasiva, dados ubíquos e assédio moral, utilizados para fundamentar teoricamente as escolhas metodológicas.

3. Termos técnicos

Esta seção abordará termos técnicos essenciais para uma compreensão geral da área de desenvolvimento de software. Entre os conceitos discutidos estão linguagens de programação, frontend, backend, bancos de dados, entre outros. A familiarização com esses termos é fundamental para entender as dinâmicas e práticas utilizadas.

3.1. Linguagem de programação

Uma linguagem de programação é um sistema estruturado que permite a comunicação de instruções a um computador. Ela segue um conjunto de regras de sintaxe e semântica para criar programas. As regras de sintaxe definem a forma correta de escrever o código, enquanto as regras semânticas determinam o significado do código escrito [Gotardo 2015].

Ao usar uma linguagem de programação, você define como os dados serão manipulados, armazenados e transmitidos, e quais ações o computador deve executar em resposta a diferentes condições. O código gerado a partir dessas instruções é conhecido como ”código-fonte”, e deve seguir as normas específicas da linguagem utilizada, como Python, JavaScript, Java, PHP, TypeScript, C, C++, C#, Rust, Ruby, Dart, Pascal, Lua ou Go.

3.1.1. JavaScript e TypeScript

Apesar de sua ampla adoção, o JavaScript apresenta limitações significativas para o desenvolvimento e manutenção de aplicações de grande porte. Para superar essas restrições, foi introduzido o TypeScript, uma extensão que aprimora o JavaScript ao adicionar recursos avançados. O TypeScript é definido como um superconjunto do EcmaScript 5, o que

implica que qualquer código JavaScript válido também é compatível com TypeScript. Essa extensão enriquece a linguagem original ao incluir suporte para módulos, classes, interfaces e um robusto sistema de tipos estático.

O sistema de tipos do TypeScript é projetado para ser leve e intuitivo, permitindo aos desenvolvedores adotá-lo sem grandes esforços. Ele oferece suporte às práticas comuns de programação em JavaScript, além de habilitar ferramentas e experiências em ambientes de desenvolvimento integrados (IDEs), que anteriormente eram características de linguagens como C e Java. Por exemplo, o uso de tipos estáticos possibilita a detecção de erros durante o desenvolvimento, além de fornecer recursos como a sugestão de métodos aplicáveis a objetos. Adicionalmente, o suporte a classes do TypeScript está em conformidade com as propostas de padronização do EcmaScript 6 [Bierman et al. 2014].

Em síntese, o TypeScript pode ser considerado uma evolução do JavaScript, trazendo melhorias significativas para o desenvolvimento moderno. Atualmente, encontra-se na versão 5.7, com atualizações constantes que visam tornar o desenvolvimento de soluções mais ágil e eficiente para os programadores que utilizam essa tecnologia [TypeScript Team 2024].

3.1.2. Python

Python é uma das linguagens de programação mais populares e versáteis, conhecida por sua simplicidade e legibilidade. Embora cada linguagem tenha suas especialidades, como a portabilidade em Java ou a integração de bancos de dados em PHP, todas elas compartilham conceitos fundamentais, como variáveis para armazenar dados e funções (ou métodos) para manipulá-los. Algumas linguagens, como o LISP, são ainda mais flexíveis, permitindo tratar funções como variáveis, o que abre novas possibilidades de programação [Why Python 2021].

A linguagem Python se destaca por ser poderosa e elegante, com uma sintaxe intuitiva que facilita a leitura e compreensão do código. Ela integra muitos dos conceitos comuns a outras linguagens, ao mesmo tempo em que é altamente aplicável em cenários do mundo real. Python é um software livre com uma implementação padrão e possui uma comunidade ativa e acolhedora de desenvolvedores. Uma das grandes vantagens de aprender Python é que ela proporciona uma base sólida que torna o aprendizado de outras linguagens de programação mais fácil e natural [Why Python 2021].

Atualmente, Python está em sua terceira versão (Python 3), que é amplamente utilizada devido à sua vasta gama de bibliotecas e recursos. A linguagem é ideal para construir executáveis, desenvolver APIs, trabalhar com aprendizado de máquina e explorar inteligência artificial. Python segue em constante evolução e está atualmente na versão 3.13.1, com melhorias contínuas para atender às demandas dos desenvolvedores e expandir suas capacidades [Python Software Foundation 2024].

3.2. Backend e Frontend

O Backend refere-se à parte do servidor de uma aplicação, que lida com o armazenamento e gerenciamento de dados, além das regras de negócios e APIs. Esta camada opera em segundo plano e não tem interação direta com o usuário; em vez disso, os dados gerados e

processados são acessados através do Frontend da aplicação. Por outro lado, o Frontend é a parte da aplicação com a qual o usuário interage diretamente. Ele oferece uma interface visual e experiências de usuário, conectando-se ao Backend por meio de requisições e envios de dados para funcionar de maneira integrada [Júnior et al. 2022].

3.2.1. REST API

Os serviços web são servidores que fornecem funcionalidades essenciais para sites ou aplicativos, atendendo às suas necessidades específicas. Para facilitar a comunicação entre diferentes sistemas, os programas clientes utilizam interfaces de programação de aplicativos (APIs). De maneira geral, uma API permite que diferentes programas compartilhem dados e executem funções de forma eficaz, simplificando a interação entre sistemas computacionais [Masse 2011].

O estilo arquitetural REST é amplamente adotado para o desenvolvimento de APIs em serviços web modernos. Quando uma API segue os princípios do REST, ela é chamada de API REST. Essas APIs são compostas por uma coleção de recursos interconectados, formando o modelo de recursos da API REST. Um design bem estruturado de uma API REST não só facilita a utilização dos serviços web, mas também atrai desenvolvedores a utilizá-los. Em um mercado competitivo, onde diversos serviços web disputam a atenção dos usuários, um design eficiente e bem planejado de API REST torna-se um diferencial importante para o sucesso do serviço [Masse 2011]

3.2.2. Nest.js

O NestJS é um framework moderno e progressivo para o desenvolvimento de aplicações no lado do servidor com Node.js. Ele amplia frameworks como Express e Fastify, proporcionando uma estrutura modular e a integração de diversas bibliotecas que automatizam tarefas recorrentes. Sendo open-source e baseado em TypeScript, o NestJS se destaca pela sua flexibilidade e capacidade de organização, permitindo a criação de sistemas backend robustos e escaláveis [Saepuloh and Ginting 2022].

Este framework é amplamente utilizado em diversos cenários de desenvolvimento, incluindo a construção de sistemas backend como o utilizado neste trabalho. Atualmente, o NestJS está na versão 10.4.13 [NestJS Team 2024], e continua a evoluir para melhorar a eficiência e facilitar a implementação de aplicações complexas.

3.2.3. FastAPI

O FastAPI é um framework moderno e de alto desempenho para Python, amplamente utilizado para a construção de APIs REST. Ele é baseado em sugestões de tipo padrão do Python e se destaca por sua alta performance, comparável a frameworks como NodeJS e Go, graças ao uso de Starlette e Pydantic. Essa combinação torna o FastAPI um dos frameworks mais rápidos disponíveis para Python [Sebastián Ramírez 2024].

Entre suas principais características estão:

- **Rápido**: Apresenta um desempenho excepcional, comparável a NodeJS e Go, sendo um dos frameworks Python mais rápidos.
- **Velocidade no desenvolvimento**: Acelera o desenvolvimento de funcionalidades entre 200% e 300%, proporcionando uma experiência mais ágil para os desenvolvedores.
- **Menos erros**: Reduz em cerca de 40% os erros humanos causados durante o desenvolvimento.
- **Intuitivo**: Oferece excelente suporte ao editor, com sugestões de código em tempo real, o que reduz significativamente o tempo de depuração.
- **Fácil de aprender e usar**: Projetado para ser simples de entender e utilizar, minimizando a necessidade de leitura extensiva de documentação.
- **Código conciso**: Minimiza a duplicação de código e utiliza parâmetros de forma eficiente, reduzindo a quantidade de erros.
- **Robusto**: Gera código pronto para produção, com documentação interativa automática.
- **Padrões abertos**: Baseado em padrões abertos para APIs, como OpenAPI (anteriormente conhecido como Swagger) e JSON Schema, garantindo total compatibilidade com esses padrões.

  Atualmente, o FastAPI está na versão 0.115.6 [Sebastián Ramírez 2024].

3.2.4. React Native e Expo

O React Native é uma plataforma amplamente adotada por empresas ao redor do mundo para o desenvolvimento de aplicativos móveis multiplataforma. Ele permite criar aplicativos nativos para iOS e Android a partir de uma única base de código, simplificando o processo de desenvolvimento e evitando a necessidade de escrever código separado para cada plataforma. Com o React Native, você utiliza as mesmas ferramentas que já conhece, como o React e o JavaScript, para construir aplicações nativas de alto desempenho. Isso significa que, ao contrário de aplicativos web que dependem de adaptadores ou tradutores, os aplicativos React Native são compilados diretamente para o código nativo, garantindo maior desempenho, velocidade e confiabilidade, além de possibilitar a publicação direta na App Store e na Google Play Store [React Native 2020]. A versão atual é 0.76.3.

Além disso, o Expo, um framework complementar ao React Native, oferece uma série de ferramentas e funcionalidades que facilitam e agilizam o processo de desenvolvimento. O Expo fornece uma plataforma pronta para uso com bibliotecas integradas, componentes de UI e recursos adicionais que permitem a criação rápida de aplicativos, simplificando o fluxo de trabalho e aumentando a produtividade dos desenvolvedores [Expo 2024].

3.3. Banco de dados e SGBD

Um banco de dados é uma estrutura que armazena e organiza dados e metadados. Os dados incluem informações dos usuários, enquanto os metadados descrevem essas informações, como tipo e formato dos dados. Metadados ajudam a entender e utilizar melhor os dados no banco.

Um Sistema de Gerenciamento de Banco de Dados (SGBD) é um software que controla a estrutura do banco de dados e gerencia o acesso a ele. Ele serve como intermediário entre os usuários e o banco de dados, facilitando o acesso e escondendo a complexidade do sistema interno. Aplicações interagem com o banco de dados através do SGBD, que traduz as solicitações dos aplicativos em operações complexas [Rob and Coronel 2011].

3.3.1. Structured Query Language

A Structured Query Language (SQL) é a linguagem de consulta padrão utilizada para gerenciar dados em bancos de dados relacionais. Fundamental para a manipulação e gestão de dados, SQL permite que os usuários executem uma série de operações em bases de dados, incluindo a criação, modificação, consulta e exclusão de dados. Embora existam diversas implementações de SQL, como MySQL e PostgreSQL, elas compartilham muitas semelhanças, embora com algumas variações específicas em suas funcionalidades. A maioria dessas implementações segue o padrão SQL-92, que é o mais recente aprovado pelo ANSI (American National Standards Institute) [Anley 2002].

Através do SQL, é possível realizar diversas operações essenciais em bancos de dados, incluindo a criação de tabelas, inserção, atualização e exclusão de dados. A linguagem SQL utiliza um conjunto específico de palavras-chave para executar essas operações, que são comumente usadas em scripts de banco de dados. As principais palavras-chave e suas respectivas funções são:

- **CREATE**: Usada para criar novas tabelas e outros objetos dentro do banco de dados.
- **ALTER e REPLACE**: Utilizadas para modificar a estrutura de tabelas ou outros objetos existentes no banco de dados.
- **SELECT**: Utilizada para consultar e visualizar dados armazenados em uma ou mais tabelas.
- **INSERT**: Usada para inserir novos registros em uma tabela.
- **UPDATE**: Utilizada para modificar dados existentes em uma tabela.
- **DELETE**: Usada para remover registros de uma tabela.
- **DROP**: Utilizada para excluir tabelas ou outros objetos do banco de dados.

Essas operações formam a base para a manipulação eficaz de dados em sistemas de bancos de dados relacionais, e sua correta aplicação é fundamental para o bom funcionamento de qualquer aplicação que dependa de armazenamento de dados estruturados.

4. Computação Ubíqua

A computação ubíqua representa uma evolução natural da computação, onde a interação entre os usuários e os dispositivos tecnológicos ocorre de forma integrada e contínua. Essa abordagem busca criar ambientes repletos de dispositivos inteligentes que se concentram em melhorar a experiência humana, muitas vezes sem a necessidade de intervenção explícita dos usuários.

4.1. Definições

O termo ”ubíquo”refere-se a algo presente em todos os lugares ao mesmo tempo (onipresente). A computação ubíqua, considerada a terceira onda da computação, caracteriza-se pela transição de computadores como objetos isolados para elementos integrados em nosso cotidiano. Essa evolução retira os computadores de suas estações de trabalho tradicionais e os incorpora a dispositivos do dia a dia [Barros 2008].

O principal objetivo da computação ubíqua é a criação de ambientes equipados com dispositivos inteligentes que funcionam em segundo plano, focados nas necessidades das pessoas. Por exemplo, durante uma reunião de trabalho, um dispositivo pode captar áudio em busca de eventos específicos, como sinais de agressão verbal, sem interferir diretamente no foco do usuário.

Esses ambientes podem ser os mais variados: salas de aula, escritórios, residências ou até mesmo dispositivos pessoais como smartphones. Os dispositivos ubíquos incluem sensores, computadores, smartphones e outros equipamentos capazes de captar e processar informações do ambiente [Barros 2008].

Uma das principais características da computação ubíqua é a ”tecnologia calma”(do inglês, Calm Technology). Essa abordagem enfatiza a interação tranquila e quase imperceptível entre usuários e dispositivos, permitindo que as pessoas usufruam dos benefícios tecnológicos sem precisar se preocupar ou mesmo estar cientes da interação com as máquinas [Barros 2008]. Por exemplo, um sensor que grava conversas e processa essas informações em segundo plano é considerado um dispositivo ubíquo.

Diversas tecnologias e metodologias têm sido desenvolvidas e aprimoradas para atender às demandas de sistemas ubíquos. Entre as mais relevantes, destacam-se as Redes de Sensores Sem Fio (RSSF), a Internet das Coisas (IoT), a Computação Móvel, a Computação Pervasiva, a Web Semântica, e os conceitos de Sensibilidade ao Contexto e às Situações [Maran 2016].

A sensibilidade ao contexto, em especial, é uma característica essencial para tornar a computação verdadeiramente ubíqua, permitindo que sistemas respondam dinamicamente às condições do ambiente e às necessidades dos usuários. No entanto, alcançar esse nível de integração exige a superação de desafios tanto tecnológicos quanto culturais, que apresentam alta complexidade e demandam soluções inovadoras [Maran 2016].

A computação ubíqua combina conceitos de Computação Móvel e Computação Pervasiva. A Computação Móvel refere-se à capacidade de sincronizar e manter dados consistentes independentemente da localização do dispositivo, enquanto a Computação Pervasiva descreve a integração invisível dos dispositivos ao ambiente, com mínima ou nenhuma interação direta do usuário [Barros 2008]. Essa integração está diretamente relacionada ao objetivo deste trabalho, onde um sistema de detecção de assédio moral pode ser alimentado por dados ubíquos gerados por qualquer dispositivo próximo ao usuário. O aplicativo desenvolvido é apenas um dos meios para interagir com esse sistema, que poderia ser integrado a outros sensores ou computadores pervasivos.

4.2. Computação Móvel

A Computação Móvel baseia-se na capacidade de transportar serviços computacionais junto ao usuário, transformando o computador em um dispositivo sempre presente. Essa

abordagem amplia as possibilidades de acesso e utilização de serviços computacionais, permitindo que os usuários continuem conectados e produtivos, independentemente de sua localização física [Barros 2008].

4.3. Computação Pervasiva

A Computação Pervasiva, por sua vez, descreve a integração invisível de computadores no ambiente. Nessa abordagem, os dispositivos captam informações do ambiente, constroem modelos computacionais dinâmicos e ajustam suas funções para atender às necessidades específicas dos usuários e dispositivos. Essa interação possibilita que os computadores atuem de forma inteligente, detectando novos dispositivos e adaptando-se ao contexto em que estão inseridos. O resultado é um ambiente repleto de sensores e serviços computacionais que proporcionam experiências mais eficientes e personalizadas [Barros 2008].

4.4. Interação Humano-Computador (IHC)

Para que os usuários possam tirar o melhor proveito de um sistema, as interfaces precisam ser planejadas de forma a promover uma interação eficiente e intuitiva. Isso exige que as interfaces atendam a critérios de usabilidade, experiência do usuário, acessibilidade e comunicabilidade [Santos 2014].

A usabilidade refere-se à facilidade com que os usuários aprendem a utilizar o sistema e realizam suas tarefas. Trata-se de um conjunto de fatores que mede a qualidade da interação do usuário com o sistema. Esses fatores incluem: facilidade de aprendizado, memorização, eficiência, segurança durante o uso e satisfação do usuário ao completar suas atividades [Santos 2014].

Já a experiência do usuário foca nas emoções e sensações geradas pela interação com o sistema. Embora frequentemente associada à satisfação do usuário, vai além disso, considerando a forma como o sistema impacta a percepção e o conforto de quem o utiliza [Santos 2014].

A acessibilidade, por sua vez, diz respeito à capacidade do sistema de remover barreiras que limitam o uso por pessoas com diferentes habilidades. Ao garantir a acessibilidade, torna-se possível ampliar o alcance do software a públicos mais diversos. Um exemplo claro é o uso de leitores de tela, que auxiliam pessoas com deficiência visual a interagir de forma eficiente [Santos 2014].

Por fim, a comunicabilidade aborda como o design do sistema comunica sua lógica de funcionamento aos usuários. Quando o usuário compreende a lógica por trás de uma interface, é mais provável que ele a utilize com criatividade e eficácia. Um exemplo disso é a aplicação de analogias visuais que remetem a objetos do mundo físico, facilitando a familiaridade com o sistema [Santos 2014].

4.4.1. IHC em Sistemas Ubíquos

A interação em sistemas ubíquos vai além das interfaces tradicionais e inclui métodos de interação implícita. Isso significa que os sistemas podem capturar dados de entrada do usuário de maneira automática e natural, sem exigir atenção direta. Um exemplo clássico é o de portas automáticas que detectam a presença de uma pessoa e se abrem. Apesar de

implícita, essa interação é resultado de um sistema que interpreta dados do ambiente e responde ao comportamento humano.

Para garantir uma experiência eficaz, sistemas ubíquos devem adotar interfaces mais naturais, que simplifiquem a interação. Tecnologias como comandos de voz, gestos e telas sensíveis ao toque são exemplos de soluções que substituem ou complementam elementos tradicionais de interação gráfica. Essas tecnologias também ajudam a integrar de maneira sutil os elementos computacionais com o ambiente físico, criando uma experiência homogênea e natural para o usuário [Santos 2014].

Contudo, o desenvolvimento de sistemas ubíquos traz desafios significativos. Entre eles estão:

- A complexidade de implementar e avaliar sistemas em cenários reais.
- A necessidade de coletar e armazenar dados de interação contínua sem comprometer a experiência do usuário.
- A dificuldade de lidar com tarefas que podem ser pausadas, retomadas ou compartilhadas entre usuários.
- O desafio de identificar com precisão o contexto do usuário, dada sua imprevisibilidade e as variáveis do ambiente.

Esses desafios são ainda mais evidentes em sistemas como o proposto neste trabalho, que busca detectar assédio moral. A ideia de manter sensores de áudio discretos, lidar com múltiplos participantes e interpretar conversas em tempo real exige soluções inovadoras. Além disso, determinar a intenção ou contexto de uma conversa representa um obstáculo técnico e ético significativo.

A computação ubíqua transforma radicalmente a relação entre humanos e máquinas. Ao espalhar os serviços computacionais por todos os aspectos do cotidiano, ela amplia as possibilidades de interação. No entanto, o sucesso dessa transformação depende de projetos que priorizem princípios de IHC, garantindo que as aplicações sejam intuitivas, acessíveis e eficazes para seus usuários [Santos 2014].

5. Assédio Moral

O assédio moral caracteriza-se como qualquer comportamento abusivo, repetitivo e sistemático que afete a integridade psíquica, social ou física de uma pessoa, comprometendo sua dignidade, ameaçando seu vínculo empregatício e desestruturando o ambiente de trabalho. Esse tipo de conduta, em geral, manifesta-se de superiores hierárquicos em relação aos seus subordinados, utilizando-se de sua posição de poder para praticar tais abusos. O assédio pode ocorrer de forma explícita ou implícita.

Na forma implícita, o assediador recorre a condutas não verbais, como indiferença ou ironia, que são mais difíceis de comprovar. Essa abordagem permite que o agressor negue suas intenções caso confrontado, mascarando o comportamento abusivo. Já no formato explícito, as ações tornam-se evidentes não apenas para a vítima, mas também para terceiros. Exemplos incluem excluir a vítima de atividades ou grupos sem justificativa, expô-la a situações vexatórias diante de colegas ou menosprezar publicamente seu trabalho. Esse tipo de prática, mais grave, evidencia a perversidade do assediador [DE SOUZA 2018].

O objetivo central do assediador é subjugar a vítima, abalando sua autoestima e exercendo controle sobre ela, muitas vezes de forma insensível e destrutiva. Homens, por exemplo, podem ser atacados em relação à sua virilidade, enquanto mulheres frequentemente enfrentam intimidações que reforçam estereótipos de submissão. Essa prática perversa reflete a necessidade ilimitada do agressor de afirmar seu poder, independentemente das consequências para a vítima [DE SOUZA 2018].

A vítima, por sua vez, muitas vezes opta pelo silêncio, temendo represálias ou a perda do emprego. Esse comportamento de tolerância, embora compreensível, contribui para a perpetuação do ciclo de abuso. A humilhação contínua pode levar a graves impactos na identidade, dignidade e saúde emocional da pessoa assediada, comprometendo tanto sua vida pessoal quanto profissional. Além disso, afeta diretamente sua capacidade laboral e bem-estar geral [DE SOUZA 2018].

O assédio moral no ambiente acadêmico, especialmente em instituições de ensino superior, apresenta nuances específicas. Embora seja um problema relevante, ainda carece de discussão aprofundada na literatura. Nessas instituições, o assédio pode ocorrer tanto entre docentes e discentes quanto entre colegas, gerando consequências negativas para o bem-estar dos envolvidos, a qualidade do ensino e a dinâmica organizacional. Em alguns casos, alunos podem também agir como agentes do assédio, adotando posturas desrespeitosas em relação a professores, que frequentemente optam por não reportar tais incidentes para evitar conflitos ou prejuízos à carreira [Da Silva Franqueira et al. 2024].

A computação ubíqua surge como uma potencial aliada na detecção de assédio moral, independentemente do ambiente em que ocorre. Como grande parte dessas práticas envolve interações verbais ou comportamentos, dispositivos inteligentes podem ser utilizados para monitorar e registrar esses eventos. Por exemplo, sensores de áudio acoplados a ambientes corporativos ou acadêmicos poderiam captar conversas suspeitas e enviá-las para sistemas que avaliem a possibilidade de ocorrência de assédio moral. Da mesma forma, dispositivos que capturam vídeos ou outras formas de interação poderiam fornecer evidências concretas para a análise de comportamentos inadequados.

Empresas equipadas com tecnologia ubíqua e salas de aula monitoradas podem não apenas prevenir incidentes de assédio, mas também garantir maior segurança e justiça, ao proporcionar dados confiáveis que corroboram ou refutam denúncias. Essas inovações têm o potencial de transformar a forma como casos de assédio são identificados, reportados e tratados, contribuindo para ambientes mais saudáveis e equitativos.

6. Detecção de Assédio Moral

A detecção de assédio moral é um desafio que exige a integração de diversas tecnologias, como inteligência artificial, processamento de linguagem natural e computação ubíqua, para criar um sistema capaz de identificar padrões de comportamento abusivo. Este projeto propõe uma solução inovadora que utiliza gravações de áudio como entrada para um sistema composto por APIs interconectadas, modelos de inteligência artificial e técnicas de análise textual. O objetivo é oferecer uma ferramenta prática e eficiente para identificar e classificar possíveis ocorrências de assédio moral em diferentes contextos.

6.1. Arquitetura

A arquitetura do sistema é baseada em APIs que se comunicam para realizar a detecção de assédio moral a partir de áudios recebidos. A aplicação móvel, desenvolvida com foco em testes do sistema, utiliza o microfone do celular para captar conversas. O fluxo geral do sistema é descrito a seguir.

O banco de dados da aplicação foi preenchido com diversas frases consideradas exemplos de assédio moral. Em um cenário real, essa tabela poderia conter um volume muito maior de registros, o que justificaria uma análise posterior para otimizar essa abordagem.

Inicialmente, o aplicativo grava uma conversa entre os participantes usando o microfone do celular. O áudio é então enviado para o sistema de detecção, com a comunicação entre o aplicativo e o backend sendo realizada por meio de uma API desenvolvida em Nest.js. O backend armazena o áudio no sistema de arquivos e utiliza um programa executável em Python, com a biblioteca Whisper, para converter o áudio em texto. A partir do texto gerado, o sistema executa quatro etapas de análise:

1. Consulta ao banco de dados: A API em Nest.js realiza uma consulta SQL para comparar o texto gerado com frases previamente inseridas no banco de dados.
2. Análise por similaridade: O texto gerado é enviado para uma API Python, que realiza uma busca por similaridade no banco de dados, retornando o resultado para a API em Nest.js.
3. Detecção por Mistral: O texto é analisado por uma API que utiliza o modelo de inteligência artificial Mistral, retornando uma análise detalhada para a API em Nest.js.
4. Detecção por Cohere: Similar à etapa anterior, mas utilizando o modelo Cohere, a análise é devolvida para a API em Nest.js.

Após essas análises, os resultados são armazenados no banco de dados e enviados ao aplicativo, permitindo ao usuário visualizar os dados detalhados. A arquitetura completa do fluxo é apresentada na Figura 1.

No caso da detecção baseada em banco de dados sem similaridade, a identificação de assédio moral é atribuída como decisão do ”Administrador”, sem implicar um caráter definitivo ou punitivo. Caso o banco de dados não detecte assédio, os usuários podem votar se consideram ou não o texto como assédio moral. Os votos geram um resultado que pode ser positivo, negativo ou indeterminado, dependendo do consenso ou empate entre os votos. Esse processo permite que o sistema evolua ao incorporar a percepção coletiva dos usuários.

Já as detecções realizadas pelos modelos de IA Mistral e Cohere são imutáveis pelos usuários, uma vez que os modelos são treinados e mantidos por empresas especializadas. A detecção por similaridade, por sua vez, utiliza o mesmo banco de dados da detecção sem similaridade, sendo atualizada conjuntamente, o que dispensa a necessidade de um sistema de votação separado para essa etapa.

Dessa forma, a solução proposta combina técnicas tradicionais e avançadas de análise textual, aproveitando tanto a inteligência coletiva quanto a precisão de modelos modernos de inteligência artificial.

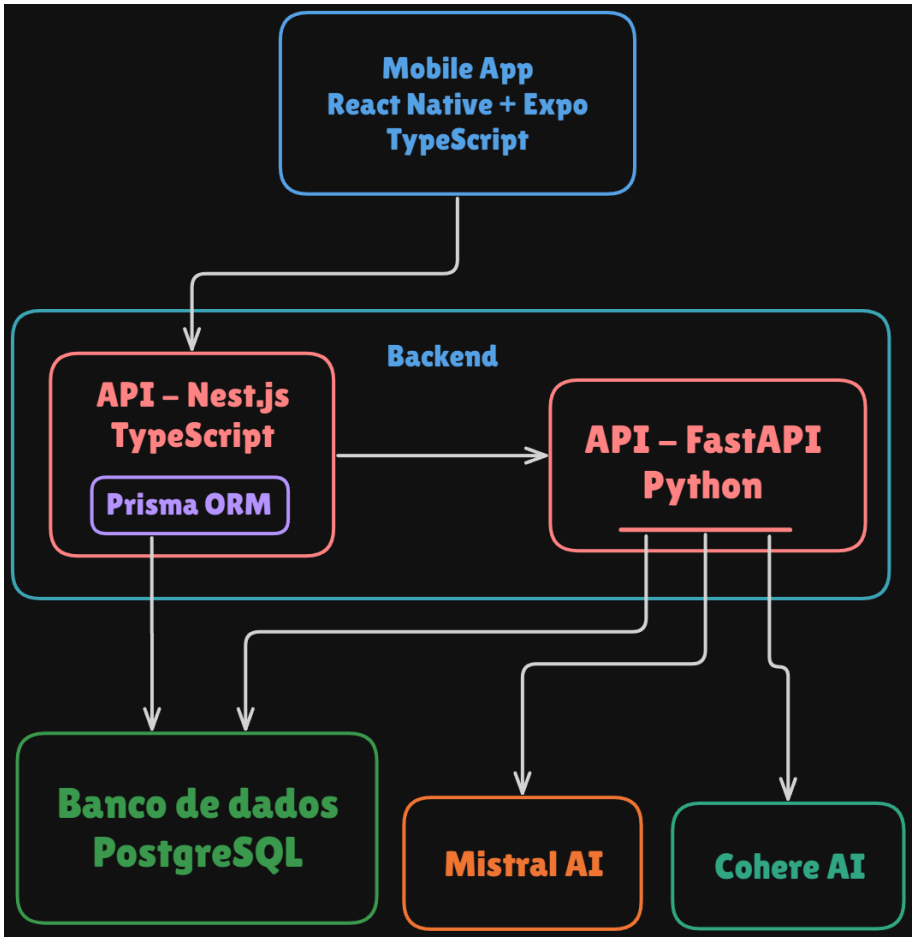

Figura 1. Mapa mental da arquitetura do projeto

6.2. Aplicativo

O aplicativo móvel desenvolvido para este projeto desempenha um papel essencial ao permitir a interação do usuário com o sistema de detecção de assédio moral. Ele foi projetado para ser intuitivo e funcional, oferecendo uma experiência prática para gravação de áudios e consulta aos resultados das análises. O aplicativo conta com uma interface organizada em diferentes telas, cada uma com funcionalidades específicas para atender ao fluxo do sistema.

6.2.1. Tela inicial

A tela inicial do aplicativo (Figura 2) é o ponto de partida para o usuário. Ela apresenta o objetivo do aplicativo por meio de uma interface limpa e direta, com um link que leva até a funcionalidade principal, que está presente na tela de gravação.

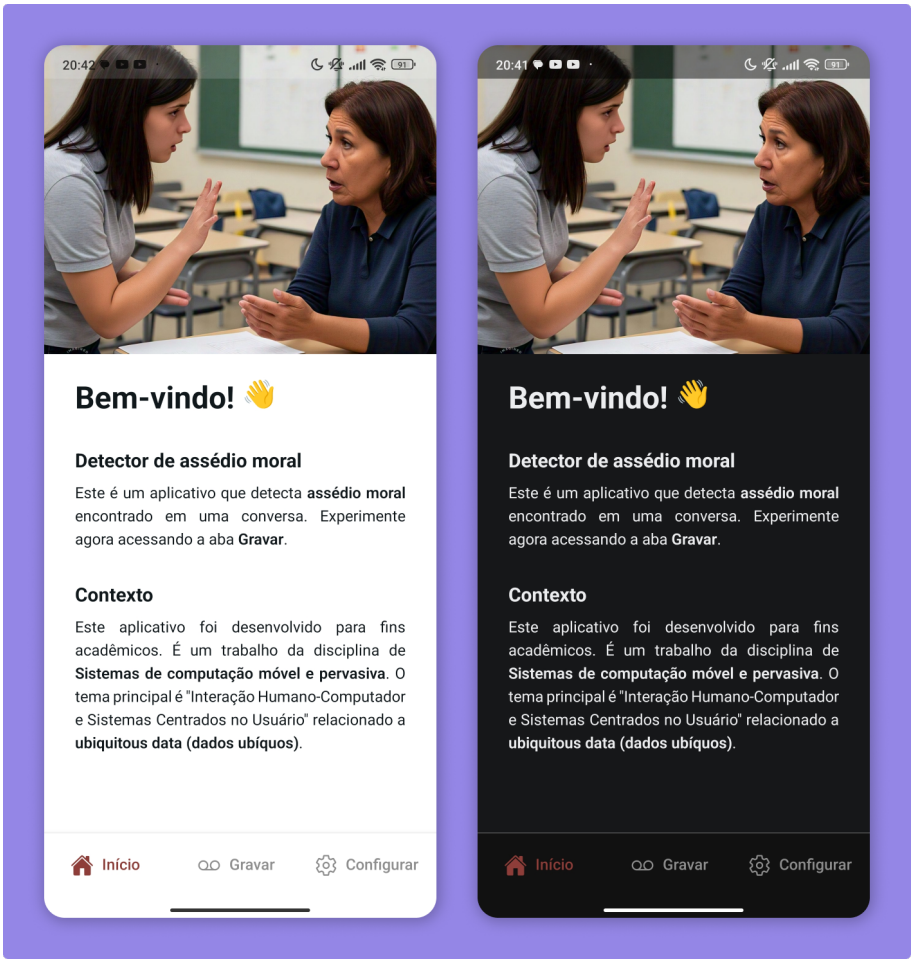

Figura 2. Tela inicial do aplicativo

6.2.2. Tela de gravação e detecção

A tela de gravação e detecção (Figura 3) é onde ocorre a interação principal do usuário com o sistema. Nela, o usuário pode iniciar ou parar a gravação de áudios utilizando um botão centralizado. Após a gravação, o áudio é automaticamente enviado ao sistema de detecção para análise. Durante o processo, a tela exibe um indicador de processamento para informar o usuário que a análise está em andamento. Além disso, os resultados da análise podem ser apresentados diretamente nesta tela, permitindo uma visualização rápida e eficiente.

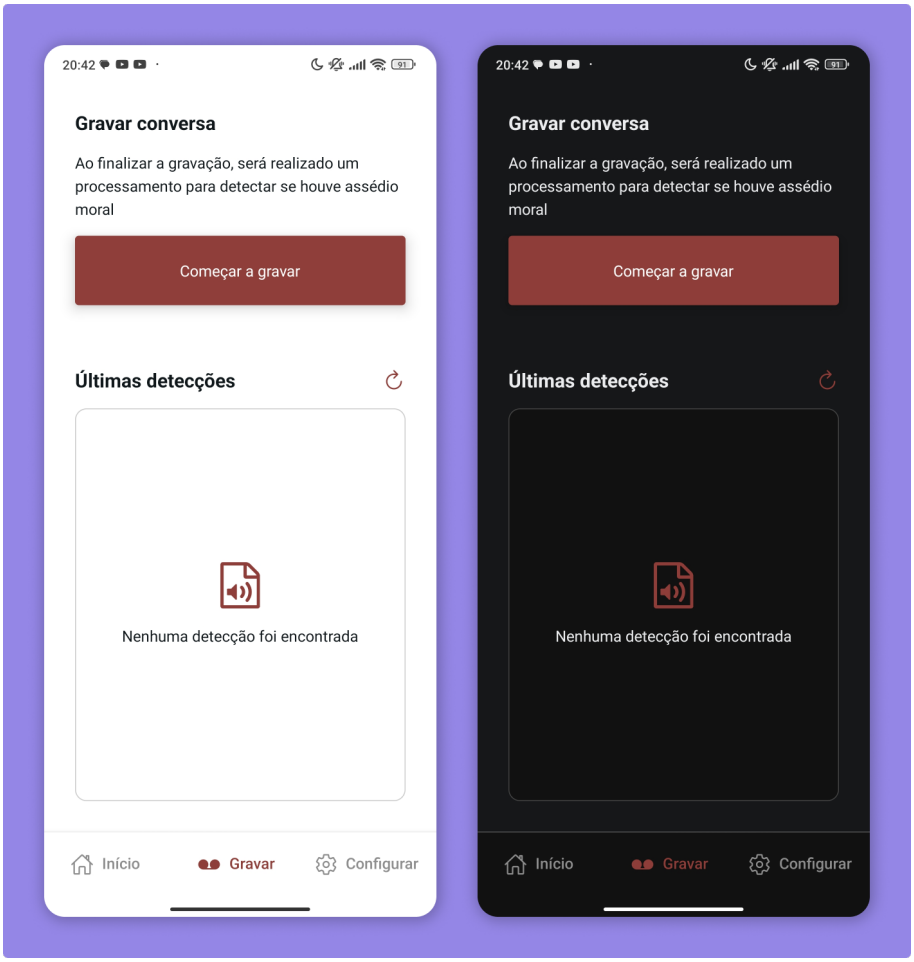

Figura 3. Tela de gravação e detecção

6.2.3. Tela de configurações

A tela de configurações (Figura 4) oferece opções para personalizar o funcionamento do aplicativo. Nessa tela, o usuário pode ajustar somente o tema do aplicativo, alternando entre modo claro, escuro ou baseado no sistema.

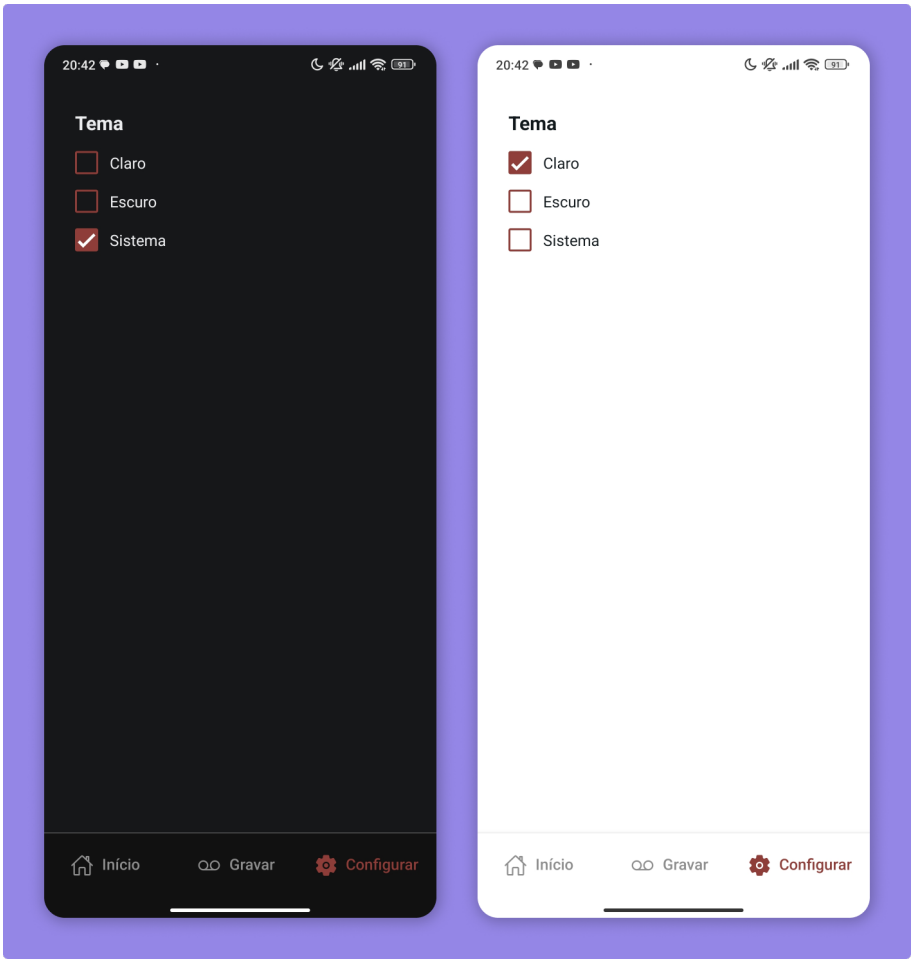

Figura 4. Tela de configurações

6.3. Utilizando o sistema de detecção

O sistema de detecção de assédio moral foi submetido a uma série de testes que exploraram diferentes cenários, desde frases simples até interações mais complexas. O objetivo desses testes foi avaliar a eficácia do sistema na identificação de assédio moral e sua capacidade de aprender e adaptar-se com base em interações dos usuários.

Inicialmente, uma frase simples e ofensiva, ”Você é um inútil”, foi gravada e analisada. O resultado revelou que todos os quatro avaliadores envolvidos consideraram a frase como assédio moral, conforme apresentado na Figura 5. Esse resultado demonstra que, mesmo utilizando frases curtas e previamente registradas no banco de dados, o sistema é capaz de identificar situações de assédio com precisão. A Figura 6 exibe os resultados gerados pelo sistema para este teste.

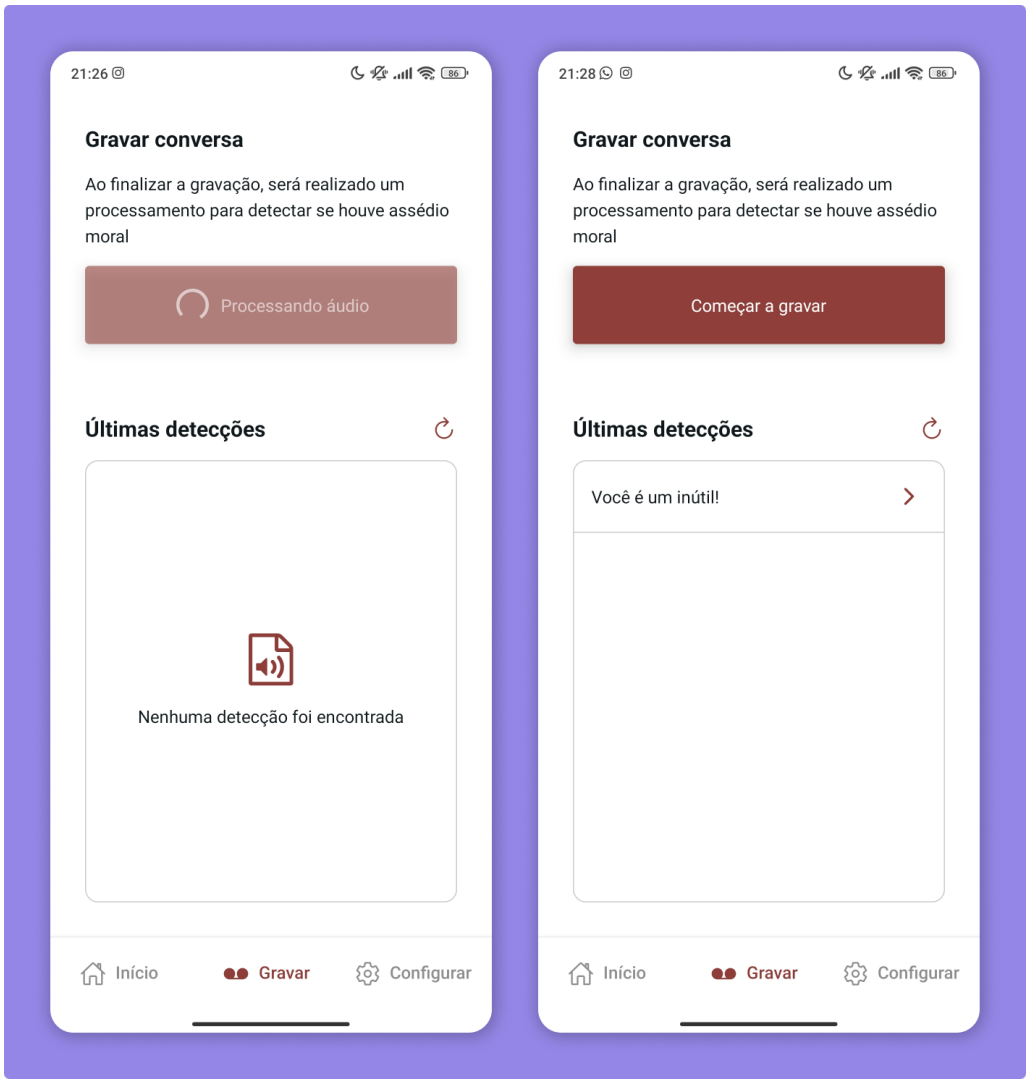

Figura 5. Primeira gravação

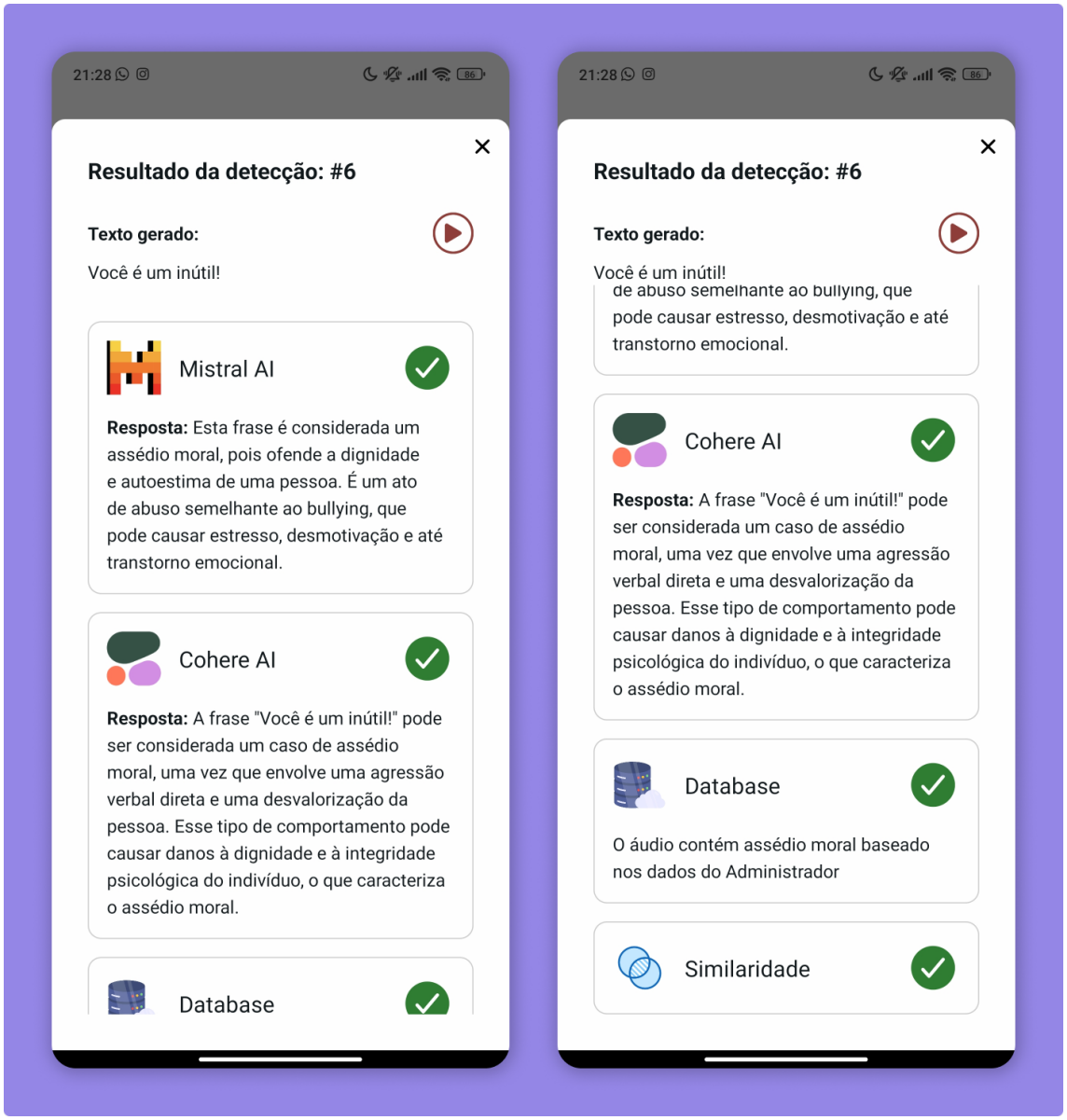

Figura 6. Resultados da primeira gravação

No segundo teste, foi utilizada uma frase comum contendo elogios, sem qualquer conotação agressiva. Os avaliadores não consideraram a frase como assédio moral, o que confirma a capacidade do sistema de diferenciar interações inofensivas de situações potencialmente abusivas. A Figura 7 ilustra o processo de gravação e os resultados deste teste.

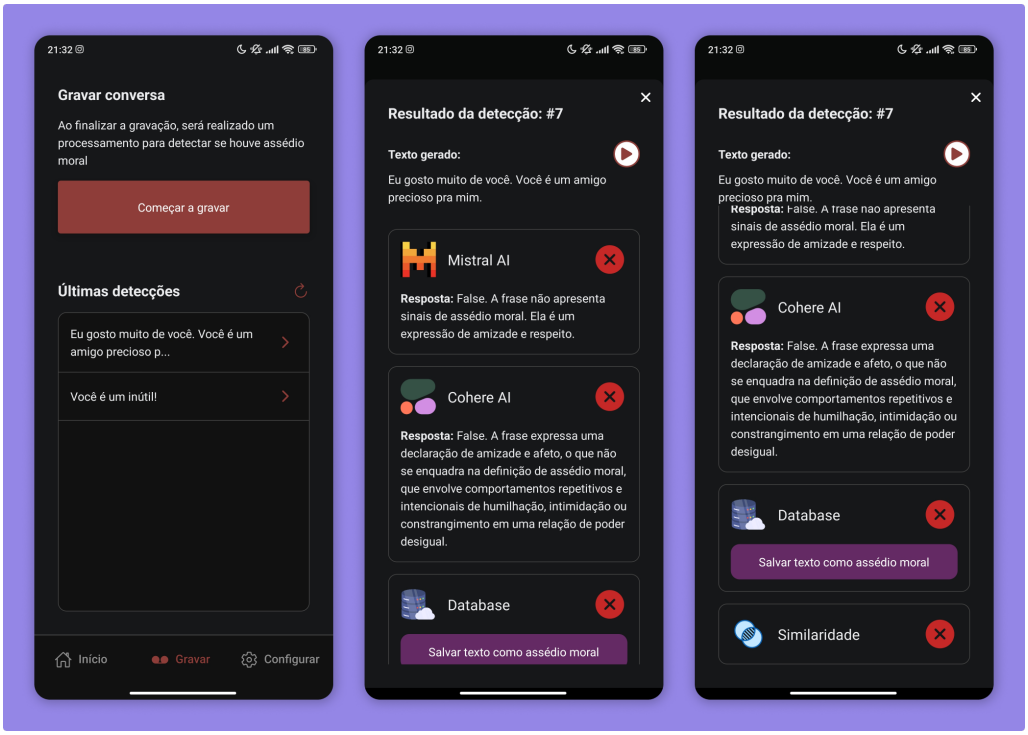

Figura 7. Segunda gravação

No terceiro teste, dois participantes simularam uma conversa profissional, na qual um funcionário hierarquicamente superior delegava tarefas a um subordinado. A análise revelou que o modelo Mistral AI considerou a interação como assédio moral (Figura 8). No entanto, o modelo Cohere não classificou a conversa como assédio moral, embora tenha identificado comportamentos antiéticos e possíveis violações das leis trabalhistas (Figura 9). Além disso, as consultas ao banco de dados e os métodos de similaridade também não detectaram assédio moral neste cenário, destacando as nuances envolvidas na interpretação de interações complexas.

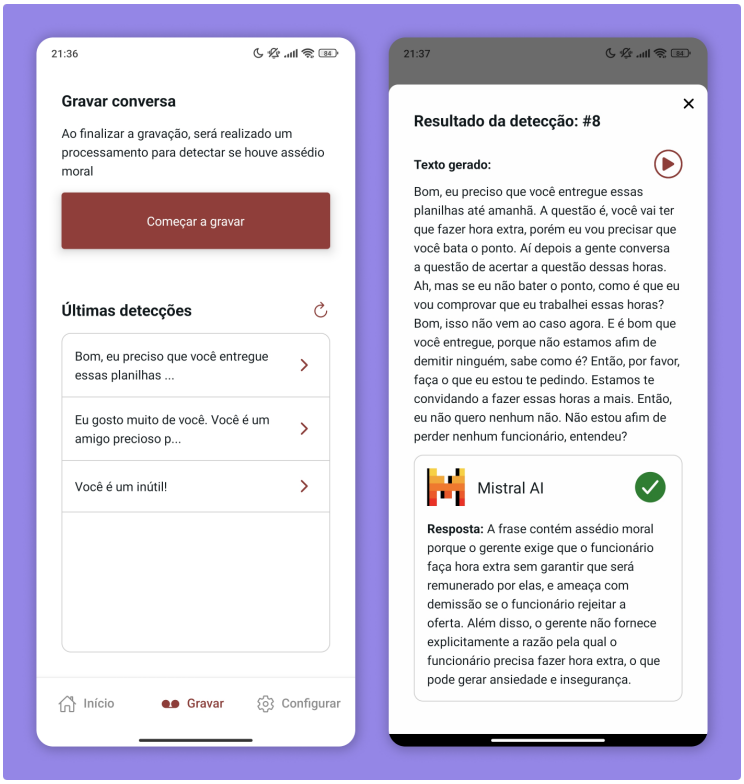

Figura 8. Terceira gravação e resultados

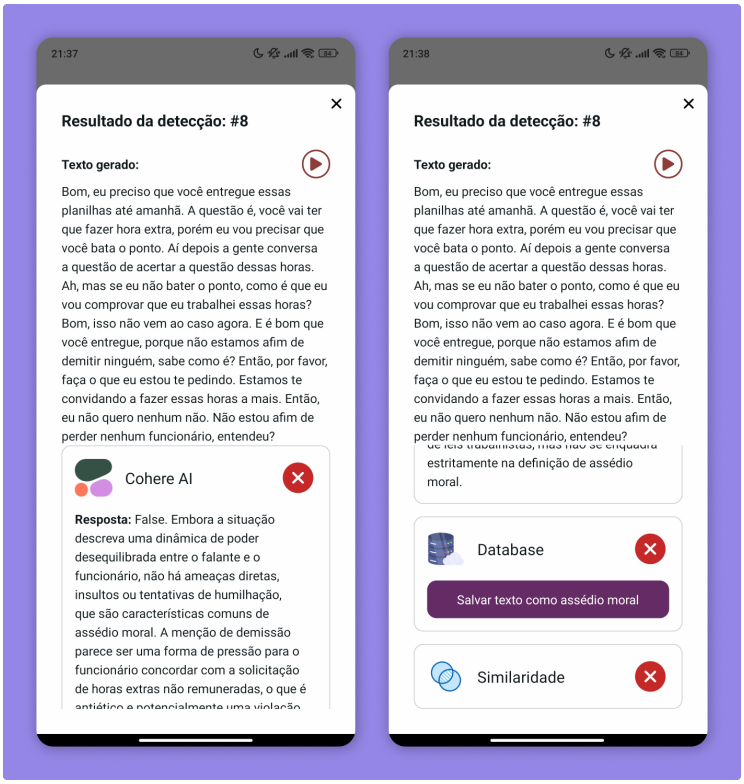

Figura 9. Terceira gravação e mais resultados

O quarto e último teste explorou a funcionalidade de votação pelos usuários. Uma frase com características regionais e gírias foi submetida ao sistema. Embora as inteligências artificiais tenham detectado assédio moral, as consultas ao banco de dados e métodos de similaridade não foram eficazes. A Figura 10 ilustra o processo de gravação e os resultados iniciais do teste.

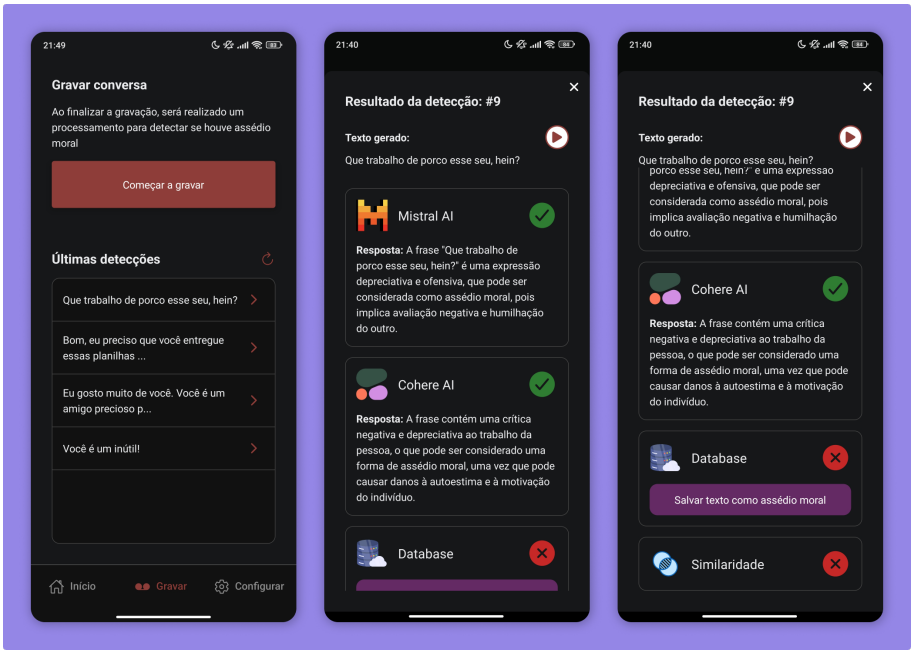

Figura 10. Quarta gravação e resultados

É possível observar na Figura 10 que a avaliação por banco de dados não detectou assédio moral, mas é possível salvar o texto caso o usuário considere assédio moral, ao fazer isso uma tela para salvar é apresentada, o que é representado pela Figura 11.

Um áudio em que assédio moral não foi detectado pelo administrador (banco de dados) permite que o usuário salve essa detecção como assédio moral caso considere, isso abre uma votação onde os usuários do aplicativo podem deixar a sua votação se aquela conversa contém assédio moral ou não. Isso pode ser observado na Figura 12, onde a primeira tela apresenta a detecção de assédio pelos usuárioe e a segunda tela mostra como está a votação para aquele texto.

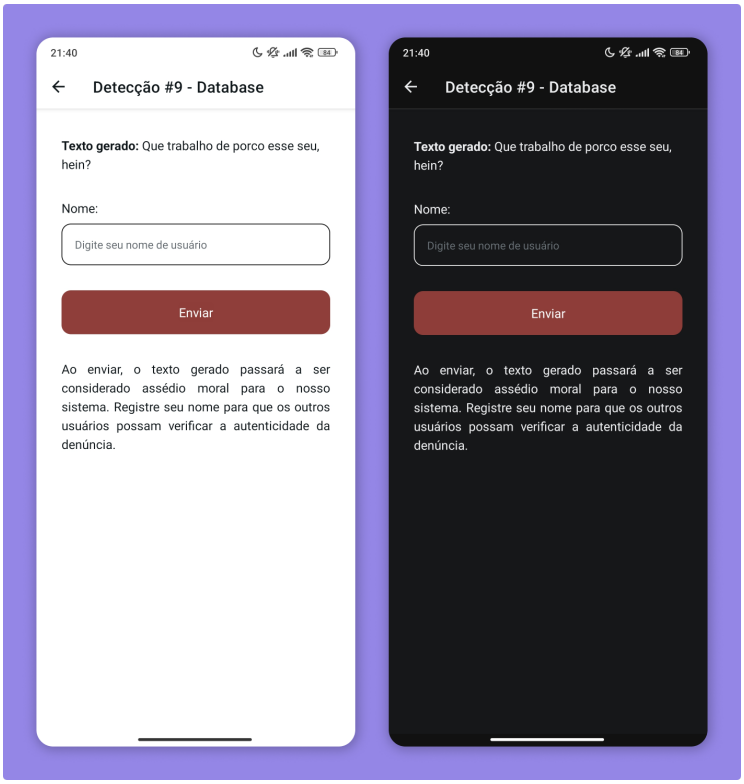

Figura 11. Quarta gravação e resultados

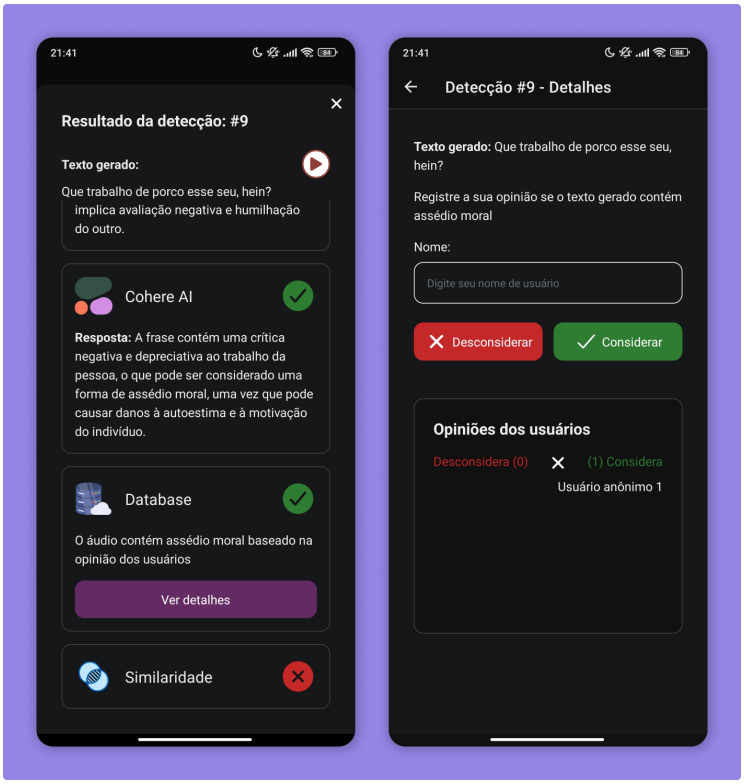

Figura 12. Votação para quarta gravação

Como visto na Figura 12, os usuários podem realizar sua votação para a existência ou não de assédio moral na frase. Em casos de empate, o resultado da detecção considerada indeterminada (Figura 13). Quando a maioria dos votos não considera a interação como assédio moral, o sistema alerta que não foi identificado assédio moral, como mostrado na Figura 14.

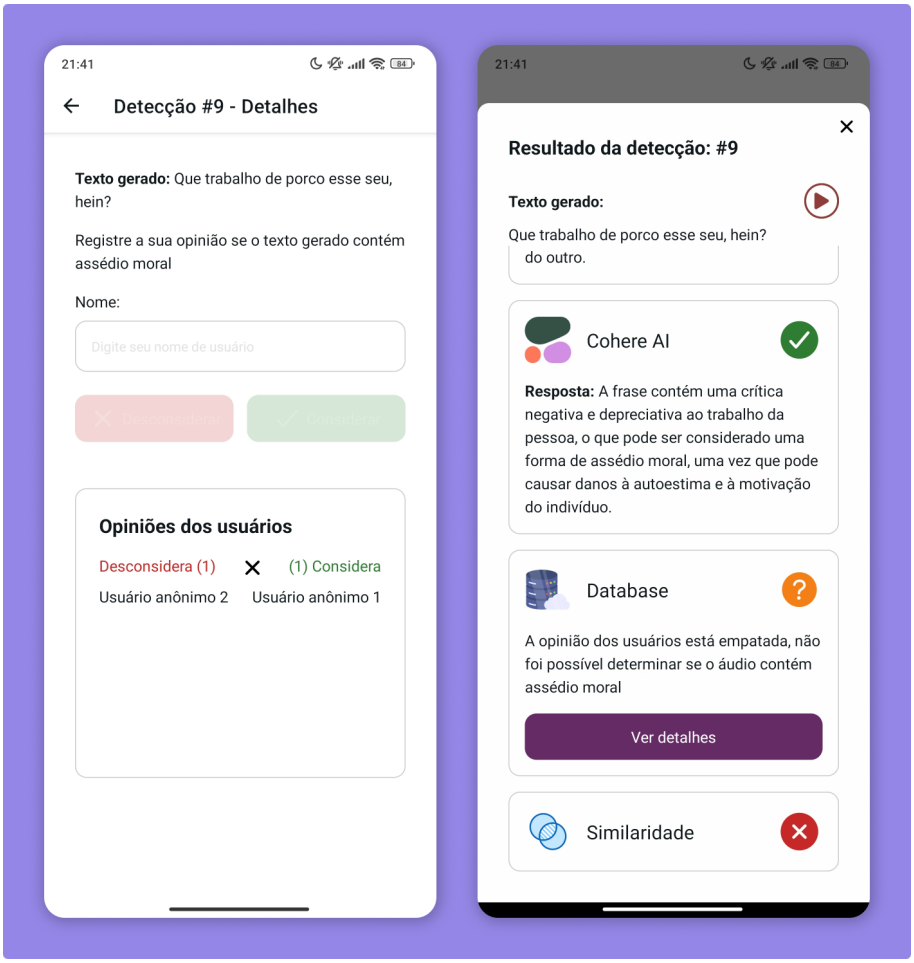

Figura 13. Empate de uma votação

Um último caso possível é a votação onde a maioria dos usuários não consideram o texto como assédio moral, isso faz com que os resultados da detecção alerte que não foi considerado assédio moral baseado na opinião dos usuários. A Figura 14 apresenta a votação onde a maioria não considera assédio moral e como isso é exibido em tela.

O quarto teste apresenta como o sistema pode aprender com os usuários. Além de conseguir detectar utilizando serviços altamente inteligentes como duas inteligências artificiais, também é possível incrementar a base do próprio sistema de detecção de assédio moral.

Este sistema pode ser integrado a outros dispositivos que sejam considerados mais pervasivos e assim detectando irregularidades nas conversas do dia-a-dia, o aplicativo desenvolvido foi idealizado principalmente para testar esse sistema e trazer a possibilidade de usuários do aplicativo incrementar o sistema baseado em suas opiniões.

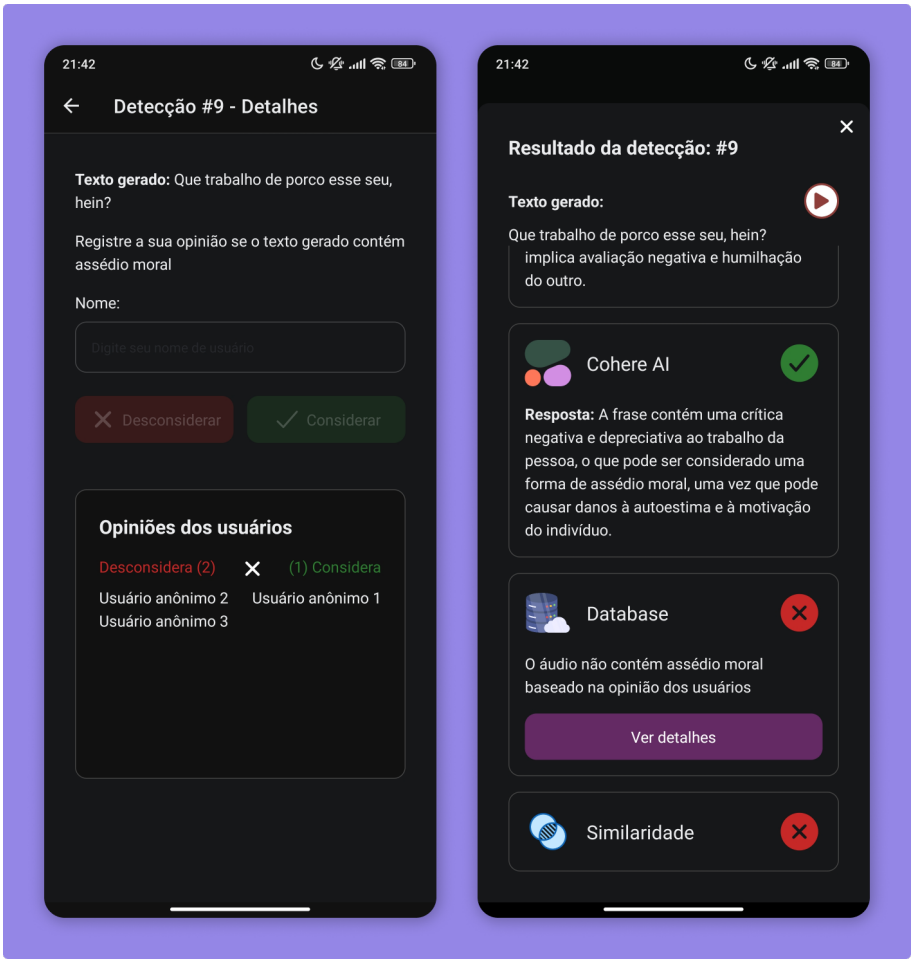

Figura 14. Desconsideram a presença de assédio moral

7. Conclusão

O sistema de detecção de assédio moral apresentado demonstrou significativo potencial na identificação de comportamentos inadequados em contextos laborais e cotidianos. Utilizando uma abordagem híbrida que combina inteligência artificial, análise de similaridade textual, banco de dados e votação colaborativa de usuários, o sistema demonstrou ser uma ferramenta adaptável e com capacidade de aprendizado contínuo. Os testes realizados evidenciaram a eficácia do sistema ao lidar com diferentes cenários, desde frases simples até diálogos mais complexos, e destacaram sua versatilidade ao integrar métodos de detecção variados, como os modelos Mistral AI e Cohere.

Apesar das contribuições, foram identificados desafios que podem ser explorados em trabalhos futuros. Primeiramente, a análise de conversas longas e complexas revelou limitações no processamento, especialmente pela demora associada à conversão de áudio em texto e ao grande número de palavras analisadas pelo banco de dados. Além disso, o sistema depende de interação explícita do usuário para iniciar e finalizar gravações, o que limita sua pervasividade. Dispositivos ubíquos, equipados com sensores que captam conversas automaticamente, poderiam tornar o sistema mais integrado ao ambiente, mas levantam questões éticas e desafios técnicos relacionados à detecção e ao armazenamento de dados em tempo real.

Outra limitação relevante está na base de dados utilizada para comparação textual. Em um cenário real, seria necessária uma base significativamente maior e mais diversificada, o que geraria novos desafios, como validar a autenticidade das frases inseridas e gerenciar o impacto do aumento de dados sobre o desempenho do sistema.

Recomendações para Trabalhos Futuros:

- **Ética em Dispositivos Ubíquos:** Investigar as implicações éticas e sociais de dispositivos que captam conversas de maneira contínua, considerando privacidade e consentimento dos indivíduos.
- **Desenvolvimento de Sistemas Ubíquos:** Projetar um dispositivo pervasivo que utilize sistemas similares ao desenvolvido, capaz de operar de forma autônoma e eficaz em ambientes diversos, minimizando a necessidade de interação direta do usuário.
- **Otimização e Escalabilidade:** Implementar técnicas de otimização para o processamento de áudios longos e a gestão de grandes volumes de dados na base de frases, garantindo maior rapidez e eficiência na detecção.

Em síntese, este trabalho valida a viabilidade técnica de um sistema para detecção de assédio moral e destaca o potencial de tecnologias de computação móvel e pervasiva na promoção de ambientes sociais e laborais mais éticos e respeitosos. Ao mesmo tempo, aponta caminhos para o desenvolvimento de soluções mais robustas e integradas, reforçando a importância de abordar o tema sob uma perspectiva interdisciplinar, que una tecnologia, ética e bem-estar social.

Referências

Anley, C. (2002). Advanced sql injection in sql server applications. Technical report, NGSSoftware Insight Security Research.

Barros, P. H. L. (2008). Uma proposta para sincronização de dados em dispositivos ubíquos. Trabalho de Conclusão de Curso (Graduação).

Bierman, G., Abadi, M., and Torgersen, M. (2014). Understanding typescript. In *ECOOP 2014–Object-Oriented Programming: 28th European Conference, Uppsala, Sweden, July 28–August 1, 2014. Proceedings*, pages 257–281. Springer Berlin Heidelberg.

Da Silva Franqueira, A. et al. (2024). Assédio moral nas instituições não escolares.

DE SOUZA, E. S. (2018). O assédio moral nas empresas.

Expo (2024). Expo. Último acesso: 04 de dezembro de 2024.

Gotardo, R. (2015). *Linguagem de Programação*. SESES, Rio de Janeiro, 1ª edition.

Júnior, A. A., de Souza Meireles, L., Figueira, L. A. R., Carmo, V. M. M., Marques-Neto, H. T., and Xavier, L. (2022). Entendendo o engajamento das comunidades front-end e back-end nos repositórios do github. In *Anais do X Workshop de Visualização, Evolução e Manutenção de Software*. SBC.

Maran, V. (2016). Modelo de consulta de dados relacionais baseada em contexto para sistemas ubíquos.

Masse, M. (2011). *REST API Design Rulebook*. O'Reilly Media, Inc.

NestJS Team (2024). Nestjs official website. Acessado: 12/04/2024.

Python Software Foundation (2024). Python official website. Acessado: 12/04/2024.

React Native (2020). React native.

Rob, P. and Coronel, C. (2011). *Sistemas de banco de dados: Projeto, implementação e gerenciamento*. Cengage Learning, 1ª edition.

Saepuloh, A. M. and Ginting, S. (2022). Perancangan sistem informasi manajemen proyek dengan menggunakan software nest. js berbasis web di pt. mitra pajakku. *INFOKOM (Informatika & Komputer)*, 10(1).

Santos, R. M. (2014). Características e medidas de software para avaliação da qualidade da interação humano-computador em sistemas ubíquos.

Sebastián Ramírez (2024). Fastapi official website. Acessado: 12/04/2024.

TypeScript Team (2024). Typescript official website. Acessado: 12/04/2024.

Weiser, M. (1994). The world is not a desktop. *Interactions*, 1(1).

Weiser, M. (1995). The computer for the 21st century: specialized elements of hardware and software, connected by wires, radio waves and infrared, will be so ubiquitous that no one will notice their presence. In *Readings in Human–Computer Interaction*. Morgan Kaufmann.

Why Python (2021). Python releases for windows.