v7.n11.2026

REVISTA CIENTÍFICA - RECIMA21 ISSN 2675-6218

MINDSYNC: ARQUITETURA E DESEMPENHO DE UMA PLATAFORMA WEB DE MIXAGEM SONORA

MINDSYNC: ARCHITECTURE AND PERFORMANCE OF A WEB SOUND-MIXING PLATFORM

MINDSYNC: ARQUITECTURA Y RENDIMIENTO DE UNA PLATAFORMA WEB DE MEZCLA SONORA

Krysthian Galbiatti Hernández[1], João Henrique Gião Borges[2]

e7119531
https://doi.org/10.47820/recima21.v7i11.9531

PUBLICADO: 11/2026

RESUMO

Este trabalho apresenta o MindSync, plataforma *web* de mixagem sonora que combina uma faixa estéreo com tons distintos por canal a uma textura ambiente escolhida conforme o nível de estresse informado pelo usuário, e avalia sua arquitetura, segurança e desempenho. O *back-end* usa .NET, Clean Architecture, CQRS com MediatR e o micro-ORM Dapper. A autenticação usa JSON Web Tokens em *cookies* HttpOnly, com verificação de posse dos recursos. O *front-end* usa React, TypeScript e Web Audio API, e a infraestrutura foi empacotada em *containers* Docker. Em um *notebook* de uso geral, 77 testes de unidade no *back-end*, 56 no *front-end*, cenários de carga com k6 e 22 verificações da superfície HTTP mostraram que, com 25 usuários virtuais simultâneos, 25.098 requisições foram processadas sem falhas e as operações de sessão tiveram mediana inferior a 25 ms. Uma comparação com o Entity Framework Core mostrou latências de escrita menores no Dapper. Todas as verificações de controle de acesso foram aprovadas, e o tempo até o primeiro som foi dominado pela decodificação do áudio (mediana de 2,6 s sem limitação de rede). O trabalho não avalia efeitos terapêuticos.

PALAVRAS-CHAVE: Clean Architecture. Controle de acesso. CQRS. Desempenho. Mixagem sonora.

ABSTRACT

This work presents MindSync, a web sound-mixing platform that combines a stereo track carrying distinct tones per channel with an ambient texture chosen according to the stress level reported by the user, and evaluates its architecture, security and performance. The back-end uses .NET, Clean Architecture, CQRS with MediatR and the Dapper micro-ORM. Authentication uses JSON Web Tokens in HttpOnly cookies with resource-ownership checks. The front-end uses React, TypeScript and the Web Audio API, and the infrastructure was packaged in Docker containers. On a general-purpose laptop, 77 back-end and 56 front-end unit tests, k6 load scenarios and 22 checks of the HTTP surface showed that, with 25 simultaneous virtual users, 25,098 requests were processed without failures and session operations had medians below 25 ms. A comparison with Entity Framework Core showed lower write latencies for Dapper. All access-control checks passed, and the time to first sound was dominated by audio decoding (median of 2.6 s without network throttling). The study does not evaluate therapeutic effects.

KEYWORDS: *Access control. Clean Architecture. CQRS. Performance. Sound mixing.*

[1] Universidade de Araraquara, Graduando em Sistemas de Informação.
[2] Universidade de Araraquara, Doutor em Química Orgânica.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

REVISTA CIENTÍFICA - RECIMA21 ISSN 2675-6218

RESUMEN
Este trabajo presenta MindSync, una plataforma web de mezcla sonora que combina una pista estéreo de tonos distintos por canal con una textura ambiental elegida según el nivel de estrés informado por el usuario, y evalúa su arquitectura, seguridad y rendimiento. El back-end usa .NET, Clean Architecture, CQRS con MediatR y el micro-ORM Dapper. La autenticación usa JSON Web Tokens en cookies HttpOnly, con verificación de propiedad de los recursos. El front-end usa React, TypeScript y Web Audio API, y la infraestructura se empaquetó en contenedores Docker. En un notebook de uso general, 77 pruebas unitarias en el back-end, 56 en el front-end, escenarios de carga con k6 y 22 verificaciones de la superficie HTTP mostraron que, con 25 usuarios virtuales simultáneos, se procesaron 25.098 solicitudes sin fallos y las operaciones de sesión tuvieron una mediana inferior a 25 ms. Una comparación con Entity Framework Core mostró latencias de escritura menores en Dapper. Todas las verificaciones de control de acceso fueron aprobadas, y el tiempo hasta el primer sonido estuvo dominado por la decodificación del audio (mediana de 2,6 s sin limitación de red). El trabajo no evalúa efectos terapéuticos.

PALABRAS CLAVE*: Clean Architecture. Control de acceso. CQRS. Rendimiento. Mezcla sonora.*

INTRODUÇÃO

A Organização Mundial da Saúde (2022) aponta que a demanda por cuidados em saúde mental é alta e que as respostas existentes ainda são insuficientes, o que reforça o interesse por soluções tecnológicas que ampliem o acesso a recursos de apoio ao bem-estar. É nesse cenário que surge o MindSync: uma plataforma *web* em .NET que permite compor e reproduzir misturas sonoras, combinando frequências binaurais e texturas ambientes, e registrar as sessões de uso.

O objetivo deste trabalho é projetar, implementar e avaliar a arquitetura, a segurança e o desempenho de uma plataforma *web* de mixagem sonora que coleta o nível de estresse relatado pelo usuário, compõe a mixagem e registra as sessões de forma segura. O trabalho não avalia efeitos terapêuticos, e o MindSync não é um dispositivo médico. Para isso, quatro aspectos técnicos foram priorizados:

- um *back-end* orientado a comandos e consultas (CQRS - *Command Query Responsibility Segregation*) com baixa latência de persistência;
- um motor de reprodução e mixagem de áudio em tempo real executado no navegador;
- uma infraestrutura baseada em *containers*; e
- mecanismos de autenticação e controle de acesso para proteger os dados de uso, que podem revelar o estado emocional do usuário.

Do ponto de vista técnico, o projeto adota uma arquitetura em camadas na qual o *Repository Pattern* intermedia a lógica de negócio e o acesso a dados, o que, segundo Martin (2019), favorece sistemas independentes e fáceis de testar. Quanto ao estímulo sonoro, como os batimentos binaurais (OSTER, 1973), as evidências de eficácia ainda são mistas (GARCIA-

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

ARGIBAY; SANTED; REALES, 2019; CHAIEB *et al*., 2015), justificando o foco na engenharia da plataforma.

O desafio desta pesquisa é verificar se é possível combinar a proteção de dados de uso, que podem revelar o estado emocional e ser enquadrados como dado referente à saúde, portanto dado pessoal sensível pela Lei Geral de Proteção de Dados (BRASIL, 2018, art. 5º, II), com uma resposta rápida, em uma plataforma de áudio executada em *hardware* convencional.

A expectativa é que uma aplicação em camadas, com CQRS, micro-ORM e banco isolado em *containers*, atenda em um *notebook* de uso geral às seguintes metas: mediana inferior a 100 ms e percentil 95 inferior a 1 s nas operações de sessão (exceto o *login*, que usa *hash* de senha deliberadamente custoso), taxa de erro HTTP inferior a 1% com 25 usuários virtuais simultâneos e aprovação em 100% das verificações de controle de acesso e de superfície HTTP.

Os limites de tempo das metas derivam dos de Nielsen (1993), de 0,1 s para resposta imediata e de 1 s para manter o fluxo de raciocínio, e as metas de erro e de aprovação adotam os valores usuais para esse tipo de verificação. Todas foram refinadas após execuções preliminares e servem como referência de interpretação, de modo que os resultados são descritivos. O tempo até o primeiro som, que avalia o motor de reprodução, não tem meta própria e é comparado a esse limite de 1 s.

1. REFERENCIAL TEÓRICO

1.1. Estímulos Auditivos e Tecnologias de Bem-Estar

Oster (1973) descreveu que tons de frequências diferentes apresentados separadamente a cada ouvido produzem a percepção de um batimento binaural, fenômeno que se origina no processamento auditivo do cérebro e fundamenta o uso de frequências binaurais como estímulo auditivo.

A literatura sobre os efeitos é heterogênea: a meta-análise de Garcia-Argibay, Santed e Reales (2019) reuniu 35 tamanhos de efeito de 22 estudos e encontrou efeito global médio de g = 0,45 sobre memória, atenção, ansiedade e dor, com efeito dependente da frequência, da duração e do momento da exposição, enquanto Chaieb *et al*. (2015) concluem que a maioria das aplicações tem resultados isolados ou contraditórios, com exceção da ansiedade.

Diante disso, o MindSync se limita a combinar uma faixa estéreo com tons distintos por canal a uma textura ambiente e a permitir compor, reproduzir e registrar sessões, sem pretender demonstrar benefício terapêutico.

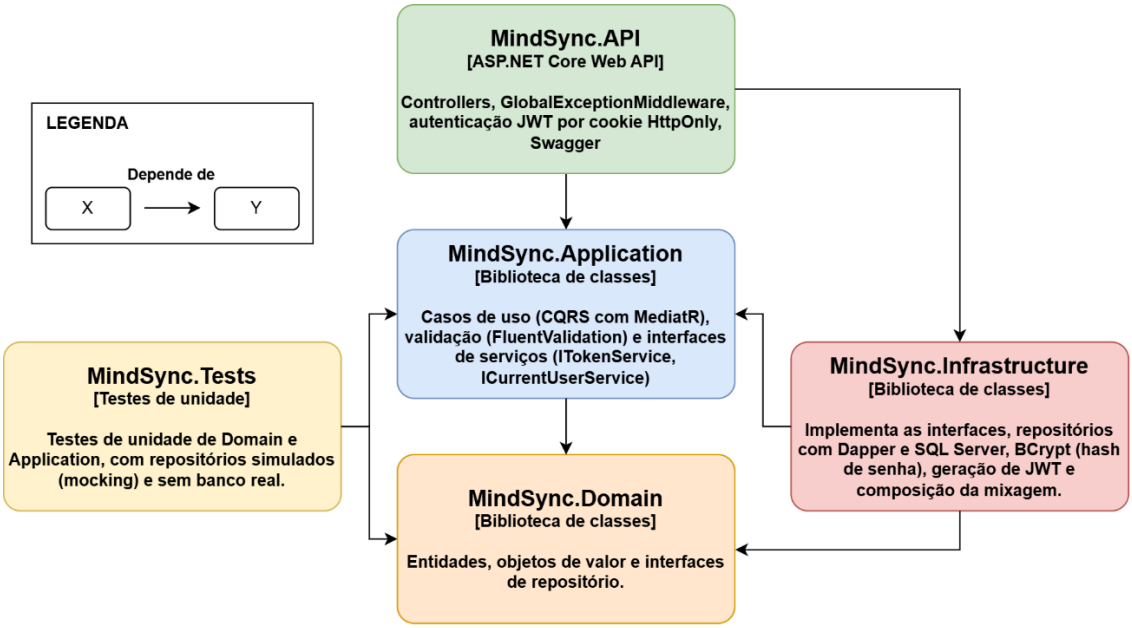
Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

1.2. Arquitetura e Padrões de Projeto

1.2.1. Repository Pattern e Camadas

O MindSync possui uma estrutura em camadas (Figura 1) organizada segundo a *Clean Architecture* (MARTIN, 2019), na qual as dependências apontam sempre para o núcleo: Application depende de Domain, e Api e Infrastructure dependem de Application, de modo que as regras de negócio não conhecem o banco de dados nem o *framework web*. O *Repository Pattern* (FOWLER *et al*., 2002), que faz a mediação entre o domínio e o acesso a dados por uma interface semelhante a uma coleção, é a fronteira dessa regra: as interfaces de repositório pertencem ao Domain e as implementações com Dapper ficam em Infrastructure, o que facilita os testes e a substituição de tecnologias. A API referencia a Infrastructure apenas para o registro das dependências, por injeção de dependência.

Figura 1. Camadas do MindSync e sentido das dependências entre os projetos



Fonte: Autoria própria, 2026.

1.2.2. CQRS e Mediator

O *Command Query Responsibility Segregation* (CQRS) separa operações que alteram o estado (*commands*) de operações que apenas consultam dados (*queries*), tratando-as como fluxos distintos (MICROSOFT, 2025), o que permite escalar e otimizar cada lado de forma independente. No MindSync, o padrão é implementado com o *Mediator Pattern*, em que um componente central encaminha cada comando ou consulta ao *handler* responsável, com

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

validação automática de entrada antes da regra de negócio, o que também facilita testar cada fluxo isoladamente.

1.3. Micro-ORMs

ORMs tradicionais oferecem muitos recursos automáticos ao custo de processamento, enquanto os micro-ORMs abrem mão de algumas abstrações para ganhar velocidade. O Dapper é um mapeador de objetos simples para .NET que adiciona métodos de extensão à conexão com o banco e executa as consultas SQL escritas manualmente, com o desempenho como característica central (DAPPER LIB, 2026). A escolha buscou registrar as sessões de forma ágil, e o custo efetivo é medido nos resultados.

1.4. Conteinerização

O Docker empacota a aplicação com suas dependências em um *container* isolado (DOCKER, 2026) e, diferentemente das máquinas virtuais, os *containers* compartilham o mesmo kernel, o que reduz o *overhead*. A escolha buscou comportamento consistente do SQL Server e do *back-end* em ambientes distintos.

1.5. Segurança

Dados de saúde mental exigem cuidados adicionais: informações sobre o estado emocional podem ser enquadradas como dado referente à saúde e, portanto, como dado pessoal sensível pela LGPD (BRASIL, 2018, art. 5º, II), cujo tratamento só pode ocorrer nas hipóteses do art. 11 e exige medidas de segurança aptas a proteger os dados (BRASIL, 2018, art. 46).

No desenvolvimento de APIs *web*, isso costuma ser endereçado por meio de autenticação baseada em *tokens*, como o JSON Web Token (JWT), formato compacto para representar *claims* entre duas partes (JONES; BRADLEY; SAKIMURA, 2015), combinada a mecanismos de verificação de posse (*ownership*), garantindo que um usuário autenticado só acesse ou modifique recursos aos quais tem direito legítimo e mitigando vulnerabilidades da categoria *Insecure Direct Object Reference* (IDOR). Essa categoria integra o controle de acesso quebrado (Broken Access Control), primeira colocada do OWASP Top 10 de 2021 (OWASP, 2021).

Para as senhas, utiliza-se o BCrypt, esquema de *hash* adaptativo cujo custo pode ser aumentado com a evolução do *hardware* (PROVOS; MAZIÈRES, 1999), o que explica o tratamento separado do *login* nas metas de desempenho.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

1.6. Interfaces e Processamento de Áudio

O React organiza a interface em componentes reutilizáveis e atualiza a tela quando os dados mudam (REACT, 2026), o que o MindSync usa para refletir o progresso de *download* de cada camada e o estado do *player*. O TypeScript é um verificador estático de tipos para JavaScript (TYPESCRIPT, 2026), que acrescenta checagem de tipos ao cliente antes da execução.

A Web Audio API é uma interface da plataforma *web* para controlar áudio no cliente (MDN WEB DOCS, 2026). Por meio de um grafo de nós de áudio, ela combina várias fontes, ajusta volumes de forma independente e analisa o sinal em tempo real, o que permite ao MindSync reproduzir simultaneamente a faixa binaural e a textura ambiente, com volume individual por camada.

1.7. Trabalhos Relacionados

Relacionam-se com este trabalho algumas plataformas comerciais de sons funcionais, como o Endel, o Brain.fm e o myNoise.

O Endel gera paisagens sonoras personalizadas que se adaptam a horário, clima, frequência cardíaca e localização (ENDEL, 2026). O Brain.fm oferece música para foco e lista, em seu site, estudos sobre os métodos que utiliza (BRAIN.FM, 2026), entre eles o de Woods *et al*. (2024), que relata que música com modulação rápida favoreceu a atenção sustentada, com efeito maior em participantes com mais sintomas de TDAH.

O myNoise disponibiliza geradores de ruído e paisagens sonoras personalizáveis, com opções binaurais e isocrônicas (MYNOISE, 2026). As alegações de eficácia partem, em sua maior parte, dos próprios fornecedores e não são avaliadas aqui. A contribuição do MindSync é descrever e avaliar quantitativamente uma arquitetura para esse tipo de plataforma.

2. METODOLOGIA

2.1. Caracterização da Pesquisa

O trabalho é uma pesquisa aplicada, voltada ao desenvolvimento de um artefato tecnológico e à avaliação quantitativa de sua arquitetura, conduzida em ciclo iterativo de cinco etapas: levantamento de requisitos e escopo (variáveis de entrada da sessão e fluxo de composição da mixagem), modelagem arquitetural em camadas, persistência de dados otimizada, isolamento de infraestrutura em ambientes portáveis e validação por testes de unidade, de carga e de segurança.

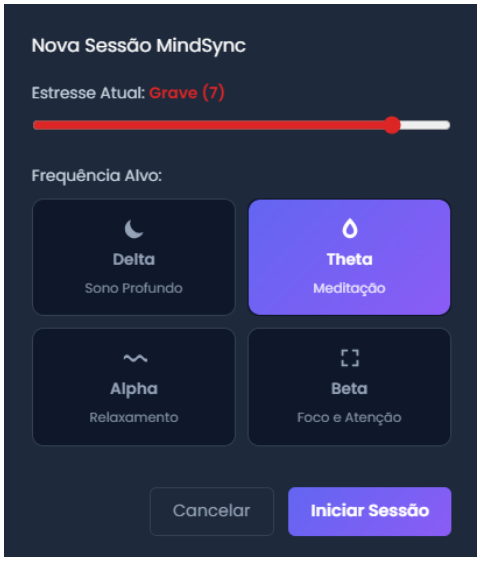
Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

2.2. Mapeamento de Requisitos

Para que o registro das sessões não interfira na experiência de uso, o fluxo no *back-end* foi projetado com processamento assíncrono e sem bloqueios, com duas operações principais:

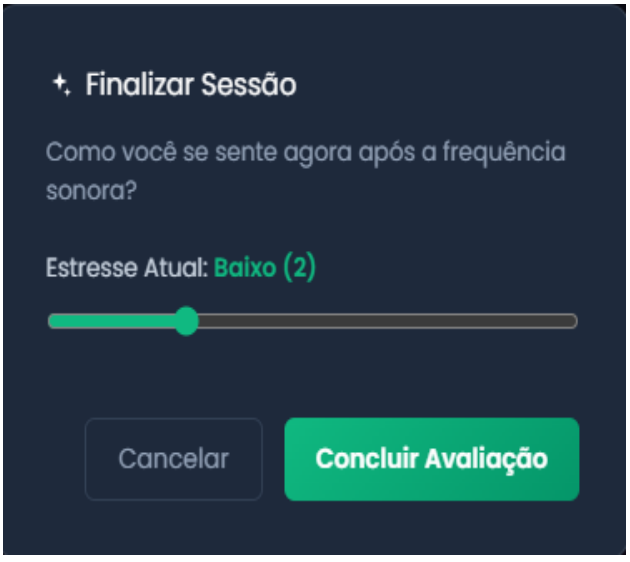
- Dados de Entrada (Figura 2): o cliente envia uma requisição com a medição inicial do nível de estresse e a frequência-alvo desejada. O identificador do usuário (*UserId*) é extraído do *token* JWT presente no *cookie* de autenticação, impedindo que uma sessão seja registrada em nome de outro usuário. O nível de estresse é autodeclarado em escala ordinal de nove pontos (0 a 8), definida para este trabalho.
- Processamento e Saída (Figura 3): o *back-end* valida os dados recebidos e disponibiliza os metadados da faixa de áudio adequada para a mixagem. Após a avaliação da sessão, o sistema registra a medição de estresse final e a data de encerramento, concluindo a operação de persistência.

Figura 2. Criação de uma sessão



Fonte: Autoria própria, 2026.

Figura 3. Encerramento de uma sessão



Fonte: Autoria própria, 2026.

2.3. Camada de Domínio

A entidade principal, *StressSession*, é uma entidade rica cujas mudanças de estado são controladas por métodos que protegem suas regras de negócio. Uma sessão nasce com o nível de estresse inicial (StressLevelBefore) e a frequência-alvo (TargetFrequency) e só é concluída por CompleteSession(), que registra o nível final (StressLevelAfter) e a data de encerramento (EndedAt) e impede que a mesma sessão seja finalizada mais de uma vez.

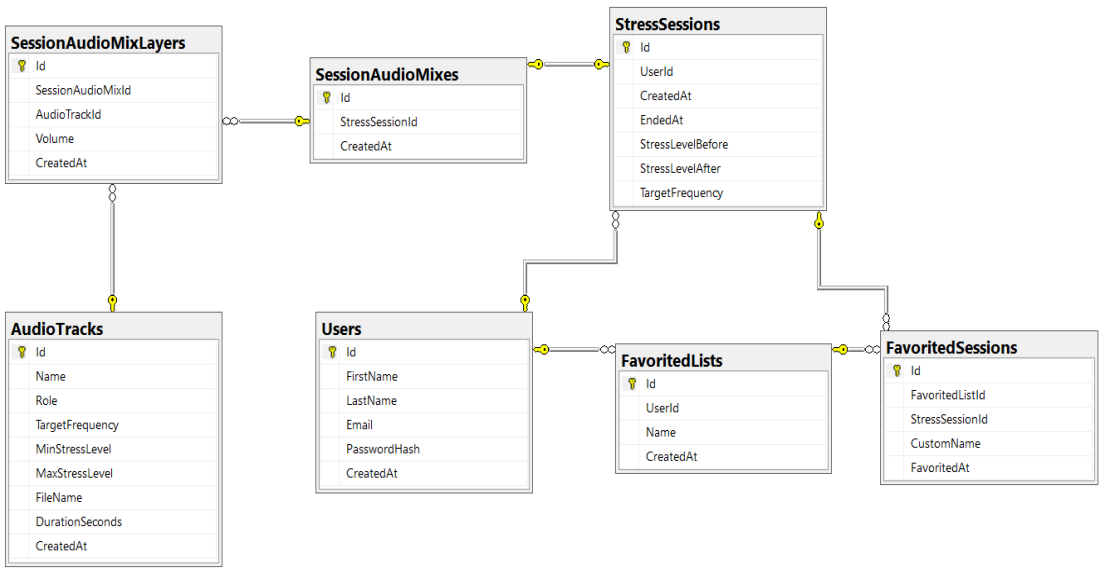
Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

O *Repository Pattern* com inversão de dependência conecta o domínio ao banco. A interface IStressSessionRepository, definida no domínio, especifica operações assíncronas como InsertAsync e GetByUserIdAsync, e os detalhes de execução ficam na infraestrutura. Na camada de aplicação, o MediatR localiza o *handler* de cada *command* e *query*, e o ValidationBehaviour aplica antes dele, via FluentValidation, as regras de validação (campos obrigatórios e limites de tamanho). Se a validação falha, uma exceção é lançada antes da regra de negócio, e o domínio nunca recebe dados estruturalmente inválidos.

2.4. Modelagem de Dados Relacional

O modelo relacional (Figura 4) tem sete tabelas. A tabela Users concentra os dados de autenticação e se relaciona com StressSessions e FavoritedLists. Cada sessão pode originar, no máximo, uma mixagem (SessionAudioMixes) com várias camadas (SessionAudioMixLayers), e cada camada referencia uma faixa do catálogo (AudioTracks). As tabelas FavoritedLists e FavoritedSessions organizam sessões em listas de favoritos.

Figura 4. Modelagem do banco de dados relacional do MindSync



Fonte: Autoria própria, 2026.

Adotou-se UNIQUEIDENTIFIER (UUID) para as chaves. A aplicação gera o identificador (Guid.NewGuid()) antes da inserção, sem esperar confirmação do banco para obtê-lo. Identificadores imprevisíveis dificultam a varredura de *endpoints*, embora não substituam a verificação de posse, pois não impedem o acesso a um recurso cujo identificador seja conhecido.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

Como a chave primária é clusterizada, GUIDs aleatórios espalham as inserções pelo índice e podem fragmentá-lo à medida que a tabela cresce, efeito que não foi medido neste estudo e fica como ponto de atenção para volumes maiores.

2.5. Camada de Persistência

A camada de persistência pode ser um dos principais pontos de lentidão em sistemas *web*, e os ORMs tradicionais podem adicionar custo por rastreamento de mudanças e geração automática de SQL. No MindSync, a camada foi construída com o Dapper, com instruções SQL e mapeamento de propriedades escritos manualmente, e espera-se que o tempo de resposta fique mais próximo do mínimo imposto pela rede e pelo banco.

2.6. Conteinerização

A infraestrutura foi isolada com Docker Compose em dois *containers* que se comunicam por rede virtual interna (mindsync-network). O *container* mindsync-sqlserver executa o SQL Server em Linux, com volume persistente. O *container* mindsync-api-*container* executa o runtime do .NET, aguarda o banco e, na primeira execução, cria o schema e popula o catálogo de faixas.

A porta do SQL Server é publicada apenas na interface de *loopback* (127.0.0.1:1433), e o ambiente da API assume Production por padrão, o que, entre outros efeitos, mantém a interface Swagger oculta. O atributo Secure do *cookie* de autenticação é controlado pela variável AuthCookie__Secure, que deve ser verdadeira quando a aplicação é servida por HTTPS. As credenciais (senha do SQL Server e segredo de assinatura do JWT) são fornecidas por variáveis de ambiente em um arquivo local .env, mantido fora do repositório.

2.7. Autenticação e Autorização

A autenticação usa JSON Web Tokens gerados após validação de credenciais com *hash* de senha via BCrypt. O *token* é entregue exclusivamente por *cookie* HttpOnly com SameSite=Lax, nunca no corpo da resposta, o que impede sua leitura por *scripts* do cliente e reduz o risco de roubo de sessão por *Cross-Site Scripting* (XSS). A defesa contra *Cross-Site Request Forgery* (CSRF) combina SameSite=Lax, CORS restrito à origem do *front-end* e aceitação apenas de corpos JSON (formulário ou text/plain recebem 415).

Além da autenticação, todo recurso de um usuário passa por verificação de posse na camada de aplicação: antes de ler, atualizar ou excluir, o *handler* compara o identificador do *token* com o proprietário do recurso e, se a verificação falha, retorna 404 em vez de 403, para que não se saiba se o recurso existe e pertence a outra pessoa.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

2.8. Composição da Mixagem Sonora

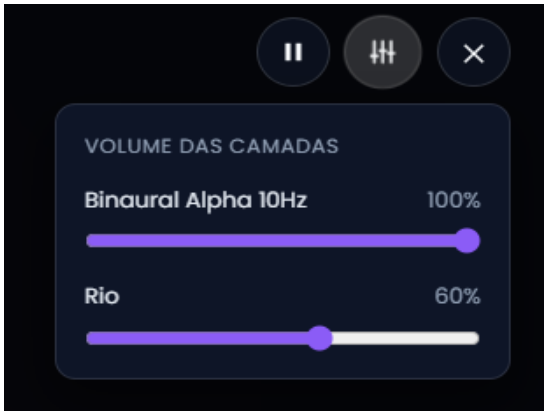
O *AudioMixComposer* recebe o nível de estresse relatado e a frequência-alvo e escolhe, entre as faixas candidatas, as que compõem a mixagem. A primeira camada é sempre a faixa binaural da frequência-alvo, e a segunda é uma textura ambiente cujo intervalo de compatibilidade (MinStressLevel/MaxStressLevel) contempla o nível relatado (Figura 5).

Havendo mais de uma compatível, uma é sorteada de forma uniforme, para variar a experiência, e, não havendo nenhuma, a mixagem fica só com a faixa binaural.

As faixas do catálogo (frequências binaurais e texturas ambientes) foram obtidas do Pixabay e são utilizadas sob a Pixabay Content License (PIXABAY, 2026), que permite o uso gratuito e sem atribuição obrigatória, mas não a venda ou a distribuição dos arquivos de forma isolada.

Os MP3 pré-gravados foram preferidos aos osciladores da Web Audio API para que a camada binaural e as texturas naturais, que exigem gravações, passem pelo mesmo caminho de *download* e decodificação medido neste trabalho. Os nomes das faixas binaurais (Delta, Theta, Alpha e Beta) são rótulos do catálogo, pois a análise espectral dos arquivos indicou batimentos entre os canais de 0,5 a 2 Hz, diferentes dos nomes.

Figura 5. Sobreposição da camada binaural Alpha sobre textura Rio



Fonte: Autoria própria, 2026.

2.9. Reprodução de Áudio

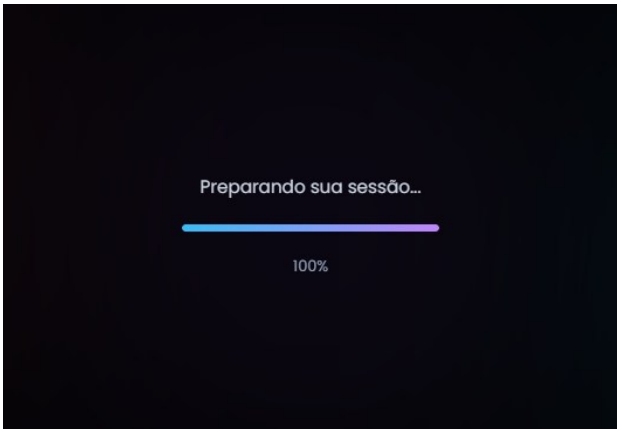
O cliente consome a mixagem retornada pela API e a reproduz com a Web Audio API. As camadas são baixadas e decodificadas em paralelo, com o progresso exposto na interface (Figura 6), e a reprodução ocorre em *loop sample-accurate*, sem a descontinuidade audível do elemento HTML audio ao repetir MP3. Cada camada expõe um *AnalyserNode* lido por um laço

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

de animação independente do React, que sincroniza uma visualização reativa ao som, e o volume de cada camada é ajustável durante a reprodução.

Para medir o início da reprodução, o cliente foi instrumentado com a Performance API: são registradas as marcas de montagem do *player*, fim da resolução da mixagem, início e fim do *download*, fim da decodificação e primeiro som, e o relatório (tempos por etapa e total) é publicado em *window.__mindsyncTiming*, de onde foi coletado.

Figura 6. Tela de espera com *download* das faixas de áudio em *background*



Fonte: Autoria própria, 2026.

2.10. Histórico e Evolução do Nível de Estresse

A tela de progresso reúne o histórico de sessões e mostra a evolução do nível de estresse relatado antes e depois de cada sessão (StressLevelBefore e StressLevelAfter), além da variação média por frequência-alvo. Como os valores são autodeclarados e não há grupo de controle, a tela apenas descreve o histórico do próprio usuário, sem permitir inferir efeito causal.

2.11. Testabilidade da Arquitetura

A separação em camadas permite validar o sistema com testes de unidade sem conexão real com o banco. As regras ficam na entidade *StressSession* e o acesso a dados na interface IStressSessionRepository, o que possibilita simular falhas e *timeouts* com *mocking* sem iniciar o SQL Server. No cliente, 56 testes em Vitest e Testing Library (ambiente jsdom) cobrem, entre outros, o cronômetro de reprodução e o *hook* do motor de mixagem.

Não há testes de integração com banco real. A integração entre API, banco e cliente é exercitada pelos cenários de carga e de segurança.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

2.12. Procedimento de Avaliação

A avaliação foi executada em um único *notebook* com Windows 11, Intel Core i7-8565U (1,80 GHz), 16 GB de memória e Docker via Rancher Desktop (máquina virtual de 7,693 GiB). API, SQL Server e gerador de carga rodaram na mesma máquina, e por isso os valores representam um ambiente de recursos limitados e compartilhados, diferente de um servidor de produção.

O k6 (imagem grafana/k6) executou os seguintes cenários: linha de base (1 usuário virtual, 100 iterações, repetida três vezes), carga com *download* do áudio (10 e 25 usuários), carga sem *download* (25 usuários), verificações de segurança e controle de acesso com dois usuários. A comparação com o Entity Framework Core repetiu o cenário sem *download*. Cada usuário autentica-se uma vez e repete as seguintes etapas: criar a sessão, resolver a mixagem, baixar as faixas (quando habilitado), encerrar a sessão e listar as sessões.

Para cada operação foram calculadas a mediana e o percentil 95, além da taxa de falhas HTTP e da aprovação das verificações (*checks*). A CPU dos *containers* foi amostrada com *docker stats* a cada cerca de 4 s. O tempo até o primeiro som foi medido no navegador, com a instrumentação do cliente, em 30 reproduções por condição, após uma de aquecimento descartada, sem limitação de rede e com o perfil Fast 4G do Chrome DevTools, no Google Chrome 155.0.8059.80 (64 bits), com *cache* desativado, cliente compilado para produção e o *player* carregado, a cada rodada, em um *iframe* oculto da mesma origem.

Não houve participantes humanos nem dados pessoais reais, pois os usuários dos cenários são contas sintéticas criadas pelos *scripts* de teste (perf/mindsync.js e perf/security.js; HERNÁNDEZ, 2026).

3. RESULTADOS E DISCUSSÃO

3.1. Testes de Unidade

Os 77 testes de unidade do *back-end* cobrem entidades (36), listas de sessões favoritas (21), mixagens de áudio (4), sessões de estresse (5), usuários (5), objetos de valor (4) e verificação de posse de mixagens (2), e os 56 testes do cliente passam sem falhas.

3.2. Latência de Persistência e Comportamento sob Carga

A linha de base teve 601 de 601 verificações aprovadas, nenhuma falha HTTP e 3,3 a 3,5 requisições por segundo. Com 25 usuários virtuais e sem *download* do áudio, foram 6.262 iterações (25.098 requisições, 148,1 por segundo), 0% de falhas e 31.335 de 31.335 verificações aprovadas. A Tabela 1 apresenta os tempos por operação.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

Tabela 1. Tempo de resposta por operação (linha de base e 25 usuários virtuais, sem *download* do áudio)

Operação	Linha de base: mediana (ms)	Linha de base: maior p95 (ms)	25 usuários: mediana (ms)	25 usuários: p95 (ms)
Login	134,0 a 185,8	Não se aplica	173,0	211,3
Criar sessão	14,3 a 17,6	66,1	14,2	32,0
Resolver mixagem	17,0 a 19,4	66,4	23,7	44,6
Encerrar sessão	10,0 a 10,3	63,1	11,6	24,3
Listar sessões	3,2 a 3,3	9,1	9,2	23,1

Fonte: Autoria própria, 2026.

Todas as operações de sessão ficaram abaixo das metas (mediana inferior a 100 ms e p95 inferior a 1 s), e o *login*, excluído delas, teve mediana de 173,0 ms.

Com o *download* das faixas (cerca de 17,5 MB por iteração e 3,5 GB por cenário, conforme a Tabela 2), a vazão fica em torno de 7,24 requisições por segundo, o gerador consome cerca de 80% da CPU, a API usa 12,6% a 15,2% e o SQL Server, 3,8% a 4,1%.

Nesse cenário, o ambiente de teste saturou antes da API. A mediana das operações de sessão subiu de cerca de 50 a 80 ms (10 usuários) para cerca de 220 a 240 ms (25) por disputa de recursos da máquina, pois sem o *download* os mesmos 25 usuários obtêm medianas inferiores a 25 ms.

Tabela 2. Comportamento sob carga, com e sem *download* do áudio

Cenário	Iterações	Req/s	Mediana das operações de sessão (ms)	audio_read y mediana / p95 (s)	Verific.	CPU API / SQL / k6 (%)
10 usuários, com áudio	200	7,24	48,6 a 80,0	5,9 / 13,6	1210/1210	12,6 / 3,8 / 80,4
25 usuários, com áudio	199	7,24	218,5 a 241,7	11,9 / 39,7	1219/1219	15,2 / 4,1 / 82,9
25 usuários, sem áudio	6262	148,1	9,2 a 23,7	Não se aplica	31335/31335	39,6 / 41,8 / 19,9

Fonte: Autoria própria, 2026.

3.3. Controle de Acesso (IDOR)

Um teste manual de acesso indevido, a partir de uma sessão autenticada, mostrou que a API retorna 404, e não 403, ao acessar a sessão de outro usuário.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

O mesmo cenário foi automatizado no k6: dois usuários tentam ler, resolver e encerrar recursos um do outro, e toda resposta indevida deve ser 404. As 390 verificações passaram, o que indica que a verificação de posse é aplicada em cada rota que recebe um identificador.

3.4. Superfície HTTP

O cenário de segurança do k6 executa 22 verificações contra a API em Production. Elas cobrem o *cookie* HttpOnly e SameSite=Lax, com Secure conforme a configuração, o corpo do *login* sem o *token*, o *preflight* de origem estranha sem cabeçalho de permissão e da origem do *front-end* aceito com credenciais, e o POST com formulário ou text/plain rejeitado com 415 sem criar recursos.

Cobrem também o acesso anônimo, o *token* com assinatura alterada e o *token* sem assinatura (*alg none*) com 401, a senha incorreta rejeitada, os erros sem *stack trace* e o Swagger não exposto, além dos controles positivos (POST JSON do dono aceito e *token* válido com 200). Todas as 22 foram aprovadas, sem substituir um teste de intrusão.

3.5. Tempo até o Primeiro Som

O tempo até o primeiro som foi decomposto em resolução, *download*, decodificação e demais etapas (Tabela 3 e Figura 7). Sem limitação de rede, a mediana total foi de 2,57 s (p95 de 3,27 s), dos quais cerca de 89% (2,30 s) são decodificação e 0,16 s, *download*. Com Fast 4G, a mediana foi de 25,2 s (p95 de 32,1 s), com o *download* em cerca de 91% (22,8 s).

Tabela 3. Tempo até o primeiro som por etapa, mediana (mínimo a máximo) e percentil 95 do total, em milissegundos

Etapa	Sem limitação de rede (n = 30)	Fast 4G (n = 30)
Resolução da mixagem	30,4 (22,8 a 139,7)	206,6 (182,4 a 461,6)
Download das faixas	161,9 (110,0 a 466,6)	22.836,0 (12.688,9 a 25.515,0)
Decodificação	2.297,8 (1.972,4 a 3.377,5)	2.186,4 (1.889,0 a 7.816,0)
Demais etapas	54,5 (38,5 a 98,1)	60,1 (36,2 a 174,8)
Total até o primeiro som	2.568,5 (2.152,5 a 3.747,2)	25.214,2 (14.920,8 a 33.748,2)
Total até o primeiro som, p95	3.266,0	32.051,2

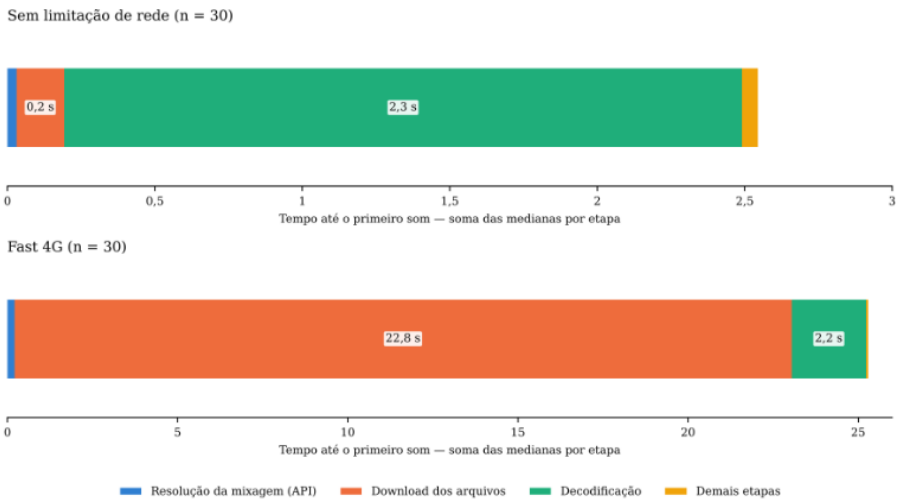
Fonte: Autoria própria, 2026.

Nas duas condições o total excede o limite de 1 s de Nielsen (1993), e em Fast 4G excede também o de 10 s, a partir do qual o usuário precisa de um indicador de progresso, como

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

a tela da Figura 6. Como as faixas são decodificadas por completo antes de tocar, a decodificação impõe um piso de cerca de 2 s mesmo em rede rápida.

Figura 7. Composição do tempo até o primeiro som (medianas por etapa)



Fonte: Autoria própria, 2026.

3.6. Comparação entre Dapper e Entity Framework Core

Para isolar o efeito da camada de dados, o acesso foi reimplementado com o Entity Framework Core 10 (consultas de leitura com AsNoTracking) atrás das mesmas interfaces e comparado ao Dapper no cenário de 25 usuários virtuais, sem *download* do áudio. Foram feitas três repetições alternadas de 3 minutos por implementação, com a API reiniciada a cada execução, e nenhuma das seis execuções teve falha. Há assimetrias entre as implementações: o Dapper usa WITH (NOLOCK) em duas leituras, e o EF Core usa um contexto por requisição sem *pool* explícito.

Os valores a seguir vêm dessas execuções, com a média de três repetições por implementação, e por isso diferem levemente dos da Tabela 1 e da Tabela 2. As medianas médias das três repetições (Dapper e EF Core, respectivamente) foram de 13,8 e 19,3 ms para criar sessão, 23,4 e 27,8 ms para resolver mixagem, 11,2 e 17,0 ms para concluir sessão e 9,1 e 9,4 ms para listar sessões. Nos percentis 95, o EF Core ficou entre 33,5 e 86,9 ms nas operações de sessão, contra 26,5 a 56,8 ms do Dapper, com forte variação entre repetições (de 40,7 a 108,6 ms em criar sessão).

A vazão foi de 36,5 iterações por segundo com o Dapper e de 34,8 com o EF Core. A API consumiu em média 45% e 68% de CPU e 82 e 122 MiB de memória, respectivamente, e o SQL Server ficou equivalente nas duas.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

3.7. Síntese dos Resultados

A Tabela 4 resume as evidências e as confronta com as metas da Introdução.

Tabela 4. Síntese dos resultados e das metas

Métrica	Resultado	Meta ou referência	Situação
Testes de unidade (*back-end*)	77 testes, sem falhas	Não se aplica	Não se aplica
Testes de unidade (*front-end*)	56 testes, sem falhas	Não se aplica	Não se aplica
Latência das operações de sessão (25 usuários)	Mediana de 9,2 a 23,7 ms e p95 de 23,1 a 44,6 ms	Mediana < 100 ms e p95 < 1 s	Atendido
Taxa de erro HTTP (25 usuários)	0% (0 em 25.098 requisições)	< 1%	Atendido
Controle de acesso (IDOR)	390/390 verificações	100%	Atendido
Superfície HTTP	22/22 verificações	100%	Atendido
Tempo até o primeiro som	Mediana de 2,57 s (sem limitação) e 25,2 s (Fast 4G)	Referência: 1 s (NIELSEN, 1993)	Não atendido

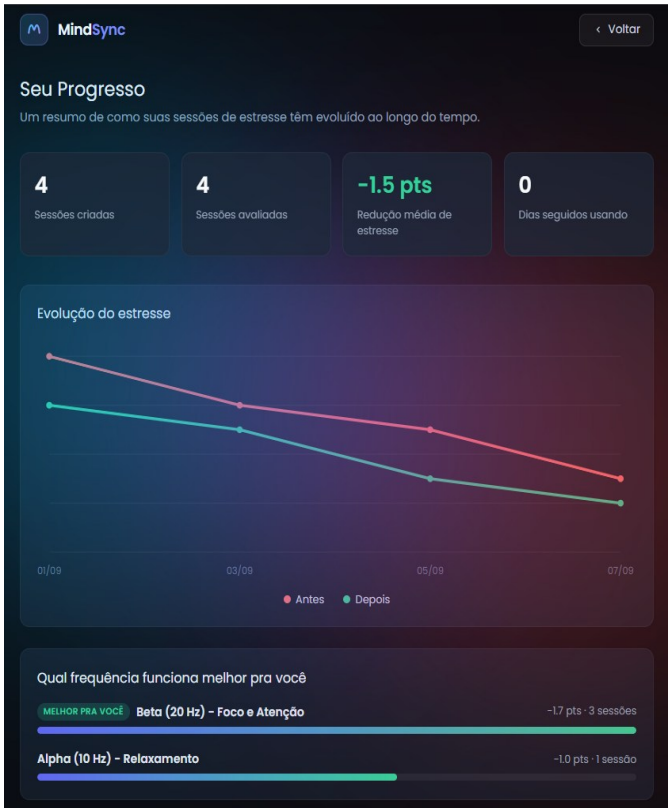
Fonte: Autoria própria, 2026.

3.8. Acompanhamento de Progresso

A funcionalidade de acompanhamento foi verificada com quatro sessões registradas em uso de teste (de 8 para 6, de 6 para 5, de 5 para 3 e de 3 para 2). A tela exibiu corretamente a contagem de sessões, a redução média do nível relatado (-1,5 ponto) e o *ranking* de frequências (Figura 8), o que confirma a consistência dos dados persistidos. Por serem dados de teste autodeclarados e sem grupo de controle, eles não sustentam conclusão sobre eficácia.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

Figura 8. Tela de acompanhamento de progresso do usuário



Fonte: Autoria própria, 2026.

3.9. Interpretação dos Resultados

Os resultados indicam que a arquitetura em camadas, com CQRS, Dapper e *containers*, atendeu às metas de latência e de erro em um *notebook* de uso geral. Com 25 usuários virtuais, as medianas das operações de sessão ficaram entre 9,2 e 23,7 ms e nenhuma das 25.098 requisições falhou.

A comparação com o Entity Framework Core mostrou medianas maiores nas operações de escrita e de resolução da mixagem (19% a 52%), cerca de 50% mais CPU e 49% mais memória na API, e empate nas leituras simples. Ainda assim, as duas implementações atenderam às metas, de modo que, neste cenário, o Dapper acrescenta margem de desempenho e as metas seriam cumpridas com qualquer uma delas. Com apenas três repetições e as assimetrias da comparação, as diferenças nos percentis 95 são apenas indicativas.

A verificação de posse é o ponto central do controle de acesso. Ela é aplicada pelo *handler* em cada rota que recebe um identificador e é coberta por testes de unidade e pelo cenário automatizado com dois usuários, o que a torna verificável de forma repetível.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

O ponto fraco é o tempo até o primeiro som. Mesmo sem limitação de rede, a mediana de 2,57 s excede o limite de 1 s de Nielsen (1993), com a decodificação completa das faixas como etapa dominante, e, em rede limitada, o *download* passa a dominar.

3.10. Limitações

- Ambiente único: API, banco e gerador de carga compartilham um *notebook*. Nas cargas com áudio, o gerador saturou antes da API, e a capacidade máxima da API não foi determinada. Os resultados de desempenho valem para esse ambiente e não permitem generalizar para outras infraestruturas.
- Metas descritivas: refinadas após execuções preliminares e usadas como referência de interpretação, o que não configura teste confirmatório.
- Tempo até o primeiro som: 30 reproduções por condição, em um navegador e dispositivo, com rede simulada.
- Escopo de saúde: sem estudo com participantes, avaliação de usabilidade ou avaliação de efeitos terapêuticos. O efeito binaural depende de fones estéreo, que o sistema não detecta nem exige, e a escala de estresse é autodeclarada e não validada.
- Segurança: verificações automatizadas de vetores selecionados, sem teste de intrusão. O atributo Secure depende de HTTPS na implantação.

3.11. Trabalhos Futuros

Como continuidade, propõe-se iniciar a reprodução de forma progressiva, para reduzir o tempo até o primeiro som, e repetir os testes de carga em servidor dedicado, com o gerador em outra máquina. Propõe-se também repetir a comparação entre Dapper e EF Core com mais repetições e configurações equivalentes de isolamento e de *pool*, realizar teste de intrusão e implantar com HTTPS. Por fim, sugere-se um estudo com participantes para avaliar usabilidade e efeitos percebidos.

4. CONSIDERAÇÕES FINAIS

O objetivo de projetar, implementar e avaliar a arquitetura, a segurança e o desempenho de uma plataforma *web* de mixagem sonora foi alcançado, resultando em um fluxo funcional de ponta a ponta, da autenticação ao acompanhamento do histórico de sessões do usuário.

Os resultados atendem às metas da Introdução: com 25 usuários virtuais, as operações de sessão tiveram mediana entre 9,2 e 23,7 ms e p95 máximo de 44,6 ms, sem falhas HTTP, e todas as verificações de controle de acesso e de superfície HTTP foram aprovadas. As metas

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

foram refinadas após execuções preliminares, e os resultados são descritivos e específicos do ambiente de teste, sem generalização para outras infraestruturas.

O tempo até o primeiro som, com mediana de 2,57 s sem limitação de rede e de 25,2 s em Fast 4G, não atendeu ao limite de 1 s de Nielsen (1993) e constitui a principal oportunidade de melhoria, por exemplo, com o início progressivo da reprodução. O trabalho não avalia efeitos terapêuticos dos estímulos sonoros, cuja evidência na literatura é mista.

REFERÊNCIAS

BRAIN.FM. **Science**. 2026. Disponível em: https://www.brain.fm/science. Acesso em: 16 out. 2026.

BRASIL. **Lei nº 13.709, de 14 de agosto de 2018**. Lei Geral de Proteção de Dados Pessoais (LGPD). Brasília, DF: Presidência da República, [2018]. Disponível em: https://www.planalto.gov.br/ccivil_03/_ato2015-2018/2018/lei/l13709.htm. Acesso em: 8 set. 2026.

CHAIEB, L.; WILPERT, E. C.; REBER, T. P.; FELL, J. **Auditory beat stimulation and its effects on cognition and mood states**. Frontiers in Psychiatry, v. 6, art. 70, 2015. DOI: 10.3389/fpsyt.2015.00070.

DAPPER LIB. **Dapper - a simple object mapper for .NET**. 2026. Disponível em: https://github.com/DapperLib/Dapper. Acesso em: 19 mar. 2026.

DOCKER. **Docker documentation**. 2026. Disponível em: https://docs.docker.com. Acesso em: 1 abr. 2026.

ENDEL. **Endel**. 2026. Disponível em: https://endel.io. Acesso em: 16 out. 2026.

FOWLER, M. *et al*. **Patterns of enterprise application architecture**. Boston: Addison-Wesley, 2002.

GARCIA-ARGIBAY, M.; SANTED, M. A.; REALES, J. M. **Efficacy of binaural auditory beats in cognition, anxiety, and pain perception: a meta-analysis**. Psychological Research, v. 83, n. 2, p. 357-372, 2019. DOI: 10.1007/s00426-018-1066-8.

HERNÁNDEZ, K. G. **MindSync: repositório do código-fonte**. 2026. Disponível em: https://github.com/krysgh/MindSync. Acesso em: 10 set. 2026.

JONES, M.; BRADLEY, J.; SAKIMURA, N. **JSON Web Token (JWT)**. RFC 7519. Internet Engineering Task Force (IETF), maio 2015. DOI: 10.17487/RFC7519. Disponível em: https://www.rfc-editor.org/rfc/rfc7519. Acesso em: 18 out. 2026.

MARTIN, R. C. **Arquitetura limpa**: o guia do artesão para estrutura e design de software. 1. ed. Rio de Janeiro: Alta Books, 2019.

MDN WEB DOCS. **Web Audio API**. 2026. Disponível em: https://developer.mozilla.org/en-US/docs/Web/API/Web_Audio_API. Acesso em: 21 jul. 2026.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.

MICROSOFT. **Command Query Responsibility Segregation (CQRS) pattern**. Azure Architecture Center, atualizado em 21 fev. 2025. Disponível em: https://learn.microsoft.com/en-us/azure/architecture/patterns/cqrs. Acesso em: 16 out. 2026.

MYNOISE. **myNoise**. 2026. Disponível em: https://mynoise.net. Acesso em: 16 out. 2026.

NIELSEN, J. **Usability engineering**. Boston: Academic Press, 1993.

ORGANIZAÇÃO MUNDIAL DA SAÚDE. **World mental health report: transforming mental health for all**. Genebra: OMS, 2022. Disponível em: https://www.who.int/publications/i/item/9789240049338. Acesso em: 10 fev. 2026.

OSTER, G. **Auditory beats in the brain**. Scientific American, v. 229, n. 4, p. 94-102, out. 1973.

OWASP. **A01:2021 – Broken Access Control**. OWASP Top 10:2021. 2021. Disponível em: https://top10.owasp.org/2021/A01_2021-Broken_Access_Control/index.html. Acesso em: 16 out. 2026.

PIXABAY. **Content License Summary**. 2026. Disponível em: https://pixabay.com/service/license-summary/. Acesso em: 16 out. 2026.

PROVOS, N.; MAZIÈRES, D. **A future-adaptable password scheme**. In: USENIX ANNUAL TECHNICAL CONFERENCE, 1999, Monterey. Proceedings [...]. Berkeley: USENIX Association, 1999. p. 81-92. Disponível em: https://www.usenix.org/legacy/publications/library/proceedings/usenix99/provos/provos_html/index.html. Acesso em: 16 out. 2026.

REACT. **React documentation**. 2026. Disponível em: https://react.dev. Acesso em: 13 jun. 2026.

TYPESCRIPT. **TypeScript documentation**. 2026. Disponível em: https://www.typescriptlang.org/docs/. Acesso em: 16 jun. 2026.

WOODS, K. J. P. *et al*. **Rapid modulation in music supports attention in listeners with attentional difficulties**. Communications Biology, v. 7, n. 1, p. 1-13, out. 2024. DOI: 10.1038/s42003-024-07026-3.

Este artigo é publicado em acesso aberto (Open Access) sob a licença Creative Commons Atribuição 4.0 Internacional (CC-BY), que permite uso, distribuição e reprodução irrestritos em qualquer meio, desde que o autor original e a fonte sejam creditados.