INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DO PIAUÍ

CAMPUS PICOS

TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

JOSÉ PEDRO EREMITA FILHO

DESENVOLVIMENTO DE UM SISTEMA WEB COM AUTENTICAÇÃO SEGURA E PROTEÇÃO DE DADOS

PICOS, PIAUÍ

2026

JOSÉ PEDRO EREMITA FILHO

DESENVOLVIMENTO DE UM SISTEMA WEB COM AUTENTICAÇÃO SEGURA E PROTEÇÃO DE DADOS

Trabalho de Conclusão de Curso (artigo científico) apresentado como exigência parcial para obtenção do diploma do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas do Instituto Federal de Educação, Ciência e Tecnologia do Piauí, Campus Picos.

Orientador: Prof. Me. Ciro Ferreira de Carvalho Junior
Coorientadora: Profa. Josenês Rocha Eremita

PICOS, PIAUÍ
2026

DESENVOLVIMENTO DE UM SISTEMA WEB COM AUTENTICAÇÃO SEGURA E PROTEÇÃO DE DADOS

José Pedro Eremita Filho[1]
Ciro Ferreira de Carvalho Junior (Orientador)[2]
Josenês Rocha Eremita (Coorientadora)[3]

RESUMO

O uso crescente de sistemas web para o tratamento de dados pessoais tornou as falhas de autenticação e de controle de acesso entre as principais causas de incidentes de segurança, exigindo mecanismos técnicos capazes de atender também às exigências da legislação brasileira de proteção de dados pessoais. Este trabalho teve como objetivo desenvolver uma aplicação web estruturada em interface de programação orientada a serviços, implementando mecanismos de autenticação, controle de acesso e proteção de credenciais de usuários, avaliando sua implementação por meio de testes técnicos sistemáticos. A pesquisa, de natureza aplicada e abordagem qualitativa, foi conduzida em três etapas: levantamento teórico sobre segurança em aplicações web, desenvolvimento da interface para gerenciamento de usuários e autenticação, e avaliação técnica dos mecanismos implementados por meio de testes manuais realizados com uma ferramenta de requisições e um script de automação. Os resultados indicaram, nos aspectos verificados, o alinhamento da aplicação com práticas internacionalmente recomendadas para segurança de aplicações web e com os princípios de segurança técnica estabelecidos na legislação brasileira de proteção de dados, ao mesmo tempo em que identificaram lacunas pontuais, como a ausência de limitação de tentativas repetidas de login e a validação parcial de entradas em alguns módulos da aplicação. Conclui-se que os mecanismos implementados atendem a requisitos técnicos de segurança da informação relevantes para a conformidade legal, embora aspectos de governança e a padronização da validação de entradas ainda demandem aprimoramentos em versões futuras da aplicação.

Palavras-chave: segurança da informação; autenticação; controle de acesso; proteção de dados; API REST.

[1]Graduando em Tecnologia em Análise e Desenvolvimento de Sistemas. Instituto Federal de Educação, Ciência e Tecnologia do Piauí, Campus Picos. E-mail: jeremitafilho@gmail.com.
[2]Mestre em Ciência da Computação. Docente do Instituto Federal de Educação, Ciência e Tecnologia do Piauí, Campus Picos. E-mail: ciro.junior@ifpi.edu.br.
[3]Graduada em Direito, em Matemática e em Pedagogia. Especialista em Matemática e Física. Mestranda em Educação. Professora Seduc - PI. E-mail: josenesrochaeremita@gmail.com

ABSTRACT

The growing use of web systems for processing personal data has made authentication and access-control failures among the leading causes of security incidents, requiring technical mechanisms able to meet the requirements of Brazil's personal data protection legislation as well. This study aimed to develop a web application structured as a service-oriented programming interface, implementing user authentication, access control and credential protection mechanisms, and to evaluate its implementation through systematic technical testing. The research, applied in nature and qualitative in approach, was carried out in three stages: a theoretical review of web application security, development of the interface for user management and authentication, and a technical evaluation of the implemented mechanisms through manual tests performed with a request-testing tool and an automation script. The results indicated, in the aspects verified, that the application aligns with internationally recommended practices for web application security and with the technical security principles established in Brazilian data protection legislation, while also identifying specific gaps, such as the absence of rate limiting on repeated login attempts and partial input validation in some application modules. It is concluded that the implemented mechanisms meet information-security technical requirements relevant to legal compliance, although governance aspects and the standardization of input validation still require improvement in future versions of the application.

Keywords: information security; authentication; access control; data protection; REST API.

Data de aprovação: 24/07/2026

1 INTRODUÇÃO

O uso de sistemas web tornou-se parte do cotidiano de diferentes setores da sociedade, como educação, comércio eletrônico, serviços públicos e instituições financeiras. Esse avanço ampliou consideravelmente o volume de dados pessoais e sensíveis tratados por aplicações digitais, tornando necessária a adoção de mecanismos técnicos voltados ao controle de acesso às informações e à proteção contra acessos não autorizados (Castells, 2015).

Paralelamente a essa expansão, observa-se também o crescimento do número de incidentes de segurança relacionados a aplicações web. O relatório ENISA Threat Landscape, publicado pela European Union Agency for Cybersecurity (ENISA), aponta que vulnerabilidades em autenticação e gestão de identidade continuam entre as principais causas de invasões e vazamentos de dados em sistemas digitais (ENISA, 2023). Na mesma direção, o OWASP Top 10 (2021) destaca falhas como Broken Access Control e Cryptographic Failures entre os riscos mais críticos enfrentados por aplicações web na atualidade.

Diante desse cenário, o desenvolvimento de aplicações web modernas exige a adoção de práticas que reduzam a exposição a essas vulnerabilidades, entre as quais se destacam o uso de criptografia para o armazenamento de credenciais, a validação das entradas fornecidas pelos usuários, o controle de acesso baseado em tokens de autenticação e o gerenciamento adequado de sessões (Stallings, 2021; OWASP, 2021).

No contexto brasileiro, essas exigências técnicas se relacionam diretamente a obrigações legais previstas na Lei Geral de Proteção de Dados Pessoais (LGPD, Lei nº 13.709/2018), cujos fundamentos específicos são apresentados na seção 2.2 deste artigo. Dessa forma, o desenvolvimento de sistemas web precisa considerar não apenas os aspectos funcionais da aplicação, mas também os mecanismos técnicos que permitam atender a essas exigências legais.

A partir desse contexto, este trabalho buscou responder à seguinte questão: de que maneira mecanismos técnicos de autenticação, controle de acesso e proteção de credenciais podem ser aplicados no desenvolvimento de uma API web, de modo a

alinhar boas práticas de segurança da informação aos requisitos de proteção de dados pessoais previstos na LGPD?

Para responder a essa questão, foi desenvolvida uma aplicação web estruturada em arquitetura de API REST, na qual foram implementados mecanismos de autenticação de usuários, armazenamento de credenciais por meio de hash criptográfico e controle de acesso por meio de tokens de autenticação. O objetivo geral do trabalho consistiu em desenvolver essa aplicação aplicando práticas de segurança recomendadas pela literatura e pela legislação vigente, avaliando sua implementação por meio de testes técnicos baseados em critérios estabelecidos pelo projeto OWASP. De forma específica, buscou-se: identificar critérios sobre segurança em aplicações web relacionados à autenticação, ao armazenamento de senhas, ao gerenciamento de sessões, à validação de entradas e ao controle de acesso; desenvolver uma aplicação baseada em API REST que implementasse esses mecanismos; e avaliar, por meio de testes sistemáticos, em que medida os mecanismos implementados se relacionam com os princípios de proteção de dados previstos na legislação brasileira.

Este artigo está organizado da seguinte forma: a seção 2 apresenta o referencial teórico que fundamenta o trabalho, articulando os conceitos de autenticação, armazenamento de credenciais, gerenciamento de sessões e controle de acesso com a literatura de segurança da informação e com os princípios de proteção de dados pessoais previstos na LGPD; a seção 3 descreve os procedimentos metodológicos adotados, incluindo o tipo de pesquisa, as etapas de desenvolvimento e os critérios utilizados na avaliação técnica dos mecanismos implementados; a seção 4 apresenta o desenvolvimento da aplicação, detalhando a arquitetura adotada e os resultados dos testes de segurança realizados; e a seção 5 apresenta as considerações finais, discutindo as implicações dos resultados para a conformidade com a LGPD, as limitações do sistema e as perspectivas de trabalhos futuros.

2 REFERENCIAL TEÓRICO

2.1 O projeto OWASP e a segurança em aplicações web

O Open Web Application Security Project (OWASP) constitui uma iniciativa internacional dedicada à promoção de boas práticas para o desenvolvimento seguro de aplicações web. O projeto reúne recomendações técnicas amplamente adotadas

por desenvolvedores e profissionais de segurança da informação, com o propósito de auxiliar na identificação e na mitigação de vulnerabilidades comuns em sistemas web (OWASP, 2021).

As diretrizes propostas pelo OWASP abrangem diferentes aspectos da segurança de aplicações, como mecanismos de autenticação de usuários, armazenamento seguro de credenciais, gerenciamento de sessões, controle de acesso e validação adequada dos dados fornecidos pelos usuários. A adoção dessas práticas contribui para reduzir os riscos associados à exploração de falhas de segurança e ao acesso não autorizado a informações sensíveis. Entre os mecanismos mais recomendados, destacam-se o uso de funções de hash criptográfico para o armazenamento de senhas, a implementação de processos seguros de autenticação, o gerenciamento de sessões por meio de tokens, a definição de regras de controle de acesso e a validação dos dados recebidos nas requisições (OWASP, 2021).

2.2 Proteção de dados pessoais e a Lei Geral de Proteção de Dados Pessoais

Além dos aspectos técnicos relacionados à segurança dos sistemas, aplicações que tratam dados de usuários precisam considerar princípios voltados à proteção de dados pessoais. No Brasil, esse tema é regulamentado pela Lei nº 13.709/2018, conhecida como Lei Geral de Proteção de Dados Pessoais (LGPD), que estabelece normas para o tratamento de dados pessoais por organizações públicas e privadas (Brasil, 2018).

Entre os princípios estabelecidos pela legislação, destaca-se o previsto no Artigo 6º, inciso VII, relacionado à segurança, que orienta a adoção de medidas capazes de proteger os dados pessoais contra acessos não autorizados e situações ilícitas. Entende-se, neste trabalho, que esse princípio se materializa tecnicamente por meio de mecanismos como o uso de hash criptográfico no armazenamento de credenciais, a autenticação baseada em tokens e a definição de perfis de controle de acesso. De forma complementar, o Artigo 46 da LGPD determina que os agentes de tratamento devem implementar medidas técnicas e administrativas aptas a proteger os dados pessoais contra acessos não autorizados e contra situações acidentais ou ilícitas de perda, destruição ou alteração das informações (Brasil, 2018). Nesse sentido, a adoção de mecanismos de segurança em aplicações web contribui tanto

para a redução de riscos relacionados ao tratamento indevido de informações quanto para o atendimento das exigências legais relacionadas à proteção de dados pessoais.

Cabe ainda observar que a aplicação desenvolvida trata dados como peso corporal, medidas antropométricas e informações de rotina alimentar dos usuários. Dependendo do contexto e da finalidade de uso, esse conjunto de dados pode se aproximar da categoria de dados pessoais sensíveis relacionados à saúde, conforme o Artigo 5º, inciso II, da LGPD, sujeitos a hipóteses de tratamento mais restritas, como o consentimento específico e destacado previsto no Artigo 11 da mesma lei (Brasil, 2018). Embora o escopo deste trabalho não tenha se aprofundado nesse enquadramento, trata-se de um ponto relevante de conformidade que não pode ser ignorado em versões futuras da aplicação, sendo retomado como limitação nas considerações finais.

2.3 Mecanismos de segurança em aplicações web

2.3.1 Autenticação e gerenciamento de identidade

A autenticação corresponde ao processo responsável por verificar a identidade de um usuário antes de conceder acesso a um sistema, normalmente por meio da validação de credenciais como nome de usuário e senha. Sistemas de autenticação bem implementados reduzem de forma significativa o risco de acesso não autorizado e são considerados um dos pilares da segurança em sistemas computacionais (Anderson, 2020). Diretrizes técnicas amplamente utilizadas recomendam a adoção de mecanismos seguros de verificação de identidade, incluindo políticas adequadas de senha, proteção contra tentativas repetidas de login e o uso de métodos criptográficos para o armazenamento de credenciais (National Institute of Standards and Technology, 2017).

2.3.2 Armazenamento seguro de senhas

O armazenamento de senhas em sistemas computacionais deve evitar o registro das credenciais em formato legível. Recomenda-se, em vez disso, a utilização de funções de hash criptográfico, que transformam a senha em uma representação irreversível, reduzindo o risco de comprometimento das credenciais caso o banco de dados seja acessado indevidamente (Provos; Mazières, 1999). Esse tipo de

abordagem dificulta a recuperação das senhas originais mesmo quando os hashes armazenados são expostos.

2.3.3 Tokens de autenticação e gerenciamento de sessões

Em aplicações web modernas, o gerenciamento de sessões frequentemente é realizado por meio de tokens de autenticação, que permitem ao servidor identificar requisições provenientes de usuários já autenticados. Um dos formatos amplamente utilizados para esse fim é o JSON Web Token (JWT), que possibilita transportar informações de autenticação de forma estruturada e assinada digitalmente, permitindo a verificação da integridade dos dados transmitidos entre cliente e servidor (Internet Engineering Task Force, 2015).

2.3.4 Controle de acesso

Além da autenticação, sistemas seguros precisam contar com mecanismos de controle de acesso capazes de determinar quais recursos cada usuário pode acessar dentro da aplicação. O princípio do menor privilégio estabelece que os usuários devem possuir apenas as permissões necessárias para realizar suas tarefas, reduzindo a possibilidade de exploração indevida de funcionalidades do sistema ou de acesso não autorizado a dados sensíveis (Saltzer; Schroeder, 1975).

2.3.5 Validação de entradas em aplicações web

A validação de entradas constitui uma prática essencial para reduzir a exploração de vulnerabilidades em aplicações web, uma vez que dados fornecidos por usuários podem conter comandos maliciosos capazes de manipular consultas, executar scripts ou comprometer o funcionamento da aplicação. Por esse motivo, recomenda-se que todas as entradas recebidas sejam verificadas e tratadas antes de serem processadas pelo sistema ou utilizadas em consultas ao banco de dados (Viega; McGraw, 2001).

2.4 Comparação de abordagens técnicas e posicionamento em relação à literatura

Embora as subseções anteriores apresentem, de forma individual, os principais mecanismos de segurança considerados neste trabalho, é importante situar as escolhas técnicas adotadas em relação a abordagens alternativas discutidas na literatura, de modo a evidenciar os critérios que fundamentam as decisões tomadas no desenvolvimento da aplicação.

2.4.1 JWT em comparação com sessões tradicionais

O gerenciamento de sessões em aplicações web pode ser implementado por meio de duas abordagens principais: sessões tradicionais, armazenadas no servidor e referenciadas por um identificador enviado ao cliente, ou tokens autocontidos, como o JSON Web Token (JWT), que carregam as informações de autenticação de forma assinada digitalmente (Internet Engineering Task Force, 2015). Sessões tradicionais dependem do armazenamento de estado no servidor, o que facilita a revogação imediata de acessos, mas compromete a escalabilidade horizontal da aplicação, uma vez que exige compartilhamento do estado da sessão entre instâncias do servidor. O JWT, por sua vez, elimina a necessidade de armazenamento de estado no servidor a cada requisição, favorecendo a escalabilidade, mas dificulta a revogação imediata de tokens já emitidos, já que sua validade é verificada localmente até a expiração (Stallings, 2021). Nesta aplicação, essa limitação foi mitigada pela manutenção de um identificador de sessão vinculado ao token e armazenado na base de dados, permitindo a invalidação de tokens anteriores mesmo em uma arquitetura baseada em JWT, conforme descrito na seção 4.4.

2.4.2 Armazenamento de senhas: bcrypt em comparação com Argon2

Entre as funções de hash recomendadas para o armazenamento de senhas, o bcrypt (Provos; Mazières, 1999), utilizado por este trabalho por meio do serviço de autenticação do Supabase, baseia-se em um fator de custo computacional que dificulta ataques de força bruta. Uma alternativa mais recente é o Argon2, vencedor da Password Hashing Competition de 2015, que introduz o conceito de dificuldade de memória (memory-hardness), tornando os ataques realizados com hardware especializado, como GPUs e ASICs, significativamente mais custosos (Biryukov; Dinu; Khovratovich, 2016). Embora o Argon2 seja atualmente considerado a recomendação

mais robusta para novos sistemas, o uso do bcrypt neste trabalho decorre da delegação do armazenamento de credenciais ao serviço de autenticação do Supabase, que adota esse algoritmo como padrão; a escolha entre as duas funções, portanto, não foi uma decisão isolada do desenvolvimento da aplicação, mas uma consequência da arquitetura baseada em provedor de identidade externo.

2.4.3 Posicionamento em relação a trabalhos aplicados de segurança

Diferentemente de estudos teóricos sobre segurança da informação, que se concentram na análise conceitual de vulnerabilidades e controles (Anderson, 2020; Saltzer; Schroeder, 1975), este trabalho adota uma abordagem aplicada, na qual os mecanismos discutidos na literatura são implementados em uma API REST real e avaliados por meio de testes técnicos sistemáticos. Essa característica aproxima o trabalho de estudos de caso de desenvolvimento seguro de software, nos quais a contribuição não está apenas na descrição dos mecanismos, mas na verificação empírica de sua efetividade em um sistema em funcionamento, incluindo a identificação de lacunas de implementação, como as apresentadas na seção 4.

3 METODOLOGIA

Esta pesquisa caracteriza-se como aplicada, pois teve como objetivo desenvolver e analisar uma solução prática voltada à implementação de mecanismos de segurança em uma aplicação web. Pesquisas dessa natureza buscam produzir conhecimento direcionado à solução de problemas específicos, utilizando fundamentos teóricos para orientar a construção de soluções tecnológicas (Gil, 2008). Quanto à abordagem, o estudo possui caráter qualitativo, uma vez que analisa a implementação dos mecanismos de segurança considerando aspectos técnicos relacionados à autenticação, ao controle de acesso e à proteção de credenciais, permitindo compreender os processos e estruturas presentes no sistema desenvolvido (Creswell, 2014).

O desenvolvimento do trabalho foi organizado em três etapas. Na primeira, foi realizado o levantamento teórico sobre autenticação de usuários, armazenamento seguro de senhas, gerenciamento de sessões, validação de entradas e controle de acesso, fornecendo a base conceitual apresentada na seção anterior. Na segunda

etapa, foi desenvolvida a aplicação web, estruturada em arquitetura de API REST, responsável pelo gerenciamento de usuários e pela autenticação. Na terceira etapa, foi realizada a avaliação técnica dos mecanismos implementados por meio de testes sistemáticos baseados nas categorias de risco do OWASP Top 10 e nos princípios de proteção de dados estabelecidos pela LGPD.

A aplicação foi desenvolvida em JavaScript, utilizando o ambiente de execução Node.js e o framework Hono para a construção da API REST. Para a camada de dados e para os serviços de autenticação de usuários, foi utilizado o Supabase, plataforma que fornece banco de dados PostgreSQL e um serviço de autenticação integrado. A comunicação entre cliente e servidor ocorre por meio de requisições HTTP, seguindo o modelo de arquitetura REST, no qual cada módulo da aplicação, autenticação, perfis e academias, rotinas de treino, exercícios, progresso físico, calendário, nutrição, administração e feedback, é organizado em controladores e rotas próprias.

Para a avaliação técnica dos mecanismos de segurança implementados, foram realizados testes manuais sobre os endpoints da API por meio da ferramenta Postman, complementados por um script de automação em PowerShell para o teste de tentativas repetidas de login. Os testes cobriram os seguintes aspectos: controle de acesso às rotas protegidas, gerenciamento de sessões, tentativas repetidas de login, armazenamento de credenciais, validação de entradas e proteção contra enumeração de usuários. Os resultados de cada teste foram registrados com capturas de tela e analisados em relação às categorias do OWASP Top 10 (2021) e ao Artigo 46 da LGPD (Brasil, 2018).

Ao todo, foram executados dez testes técnicos, distribuídos entre as categorias de risco do OWASP Top 10 (2021), os princípios de proteção de dados do Artigo 46 da LGPD (Brasil, 2018) e as diretrizes de gerenciamento de sessão do NIST SP 800-63B (National Institute of Standards and Technology, 2017), conforme sintetizado no Quadro 1 (seção 4.7). Cada mecanismo foi considerado implementado quando o comportamento observado na resposta da API correspondia ao comportamento esperado para o mecanismo avaliado, terminologia adotada de forma consistente com a coluna "Resultado" do Quadro 1 (seção 4.7). Como exemplo, foram considerados implementados os casos em que houve bloqueio de acesso com código HTTP 401 para requisições sem credenciais válidas ou retorno de respostas idênticas para e-

mails cadastrados e não cadastrados no fluxo de recuperação de senha. Da mesma forma, um mecanismo foi considerado não implementado quando o comportamento observado divergia do esperado, como ocorreu nos casos de ausência de limitação de tentativas de login e de exposição de erros internos do banco de dados. Os testes foram executados em ambiente de desenvolvimento local, com Node.js v20.17.0, framework Hono 4.7 e Postman 12.15.7 para as requisições manuais, utilizando o banco de dados e o serviço de autenticação fornecidos pelo Supabase em sua configuração padrão de projeto gratuito.

A seleção dos mecanismos submetidos à avaliação técnica obedeceu a dois critérios de inclusão: (I) o mecanismo estar efetivamente implementado na aplicação, conforme descrito nas subseções 4.2 a 4.6; e (II) o mecanismo corresponder a um requisito de segurança discutido no referencial teórico (seção 2) e ser passível de associação a uma categoria de risco do OWASP Top 10 (2021), a um princípio do Artigo 46 da LGPD (Brasil, 2018) ou às diretrizes do NIST SP 800-63B (National Institute of Standards and Technology, 2017). Como critérios de exclusão, não foram objeto de teste: (I) categorias do OWASP Top 10 sem superfície de ataque correspondente na aplicação desenvolvida, a exemplo de falhas de integridade de software e dados (Software and Data Integrity Failures) e de falsificação de requisição do lado do servidor (Server-Side Request Forgery); e (II) princípios da LGPD de natureza não técnica, relacionados à governança de dados pessoais, como gestão de consentimento, portabilidade e demais direitos dos titulares, os quais são apontados como limitações do escopo deste trabalho na seção 5.

4 DESENVOLVIMENTO DA APLICAÇÃO E RESULTADOS

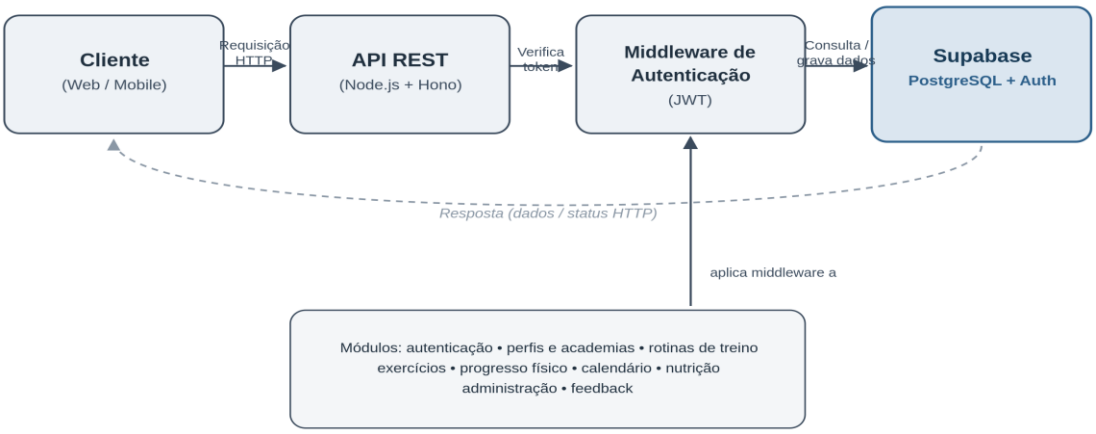
4.1 Visão geral da arquitetura

Antes de detalhar a arquitetura técnica, é importante situar o domínio da aplicação desenvolvida. Denominada Fit Track, trata-se de um sistema de acompanhamento de treino físico, voltado a usuários finais e a academias, cujo propósito é permitir o registro e o monitoramento de rotinas de treino, exercícios, progresso corporal (peso e medidas antropométricas) e hábitos nutricionais ao longo do tempo. É justamente esse conjunto de dados pessoais, potencialmente sensíveis quando relacionados à saúde do usuário, que fundamenta a relevância de se avaliar

os mecanismos de autenticação, controle de acesso e proteção de credenciais discutidos neste artigo, conforme já apontado na seção 2.2.

A aplicação desenvolvida consiste em uma API REST construída em JavaScript, executada sobre o ambiente Node.js e estruturada com o framework Hono, responsável pelo roteamento das requisições HTTP e pela definição dos middlewares utilizados pela aplicação. A persistência dos dados e a autenticação de usuários são providas pelo Supabase, plataforma que oferece um banco de dados PostgreSQL e um serviço de autenticação integrado. A API está organizada em módulos correspondentes às diferentes funcionalidades do sistema, entre eles autenticação, perfis e academias, rotinas de treino, exercícios, progresso físico, calendário, nutrição, administração e feedback dos usuários. Essa organização modular permite que os mecanismos de segurança relacionados à autenticação e ao controle de acesso sejam aplicados de forma centralizada às rotas de usuário autenticado, com exceção das rotas administrativas, que adotam mecanismo próprio discutido na seção 4.5. A Figura 1 ilustra, de forma simplificada, o fluxo de comunicação entre o cliente, a API REST, o middleware de autenticação e os serviços do Supabase.

Figura 1 – Visão geral da arquitetura da aplicação.



Fonte: autor (2026).

4.2 Autenticação de usuários

O módulo de autenticação implementa o cadastro de usuários por meio de e-mail e senha, com confirmação obrigatória do endereço de e-mail antes da liberação do acesso: ao se cadastrar, o usuário recebe um token de verificação válido por 24 horas, e o login somente é permitido após a confirmação desse token. Também foi implementado o login por meio de contas Google, no qual o token de identidade fornecido pelo provedor é validado diretamente junto ao serviço do Google antes da criação ou da localização do usuário correspondente na base de dados. Adicionalmente, a aplicação oferece um fluxo de recuperação de senha baseado em token temporário enviado por e-mail, válido por uma hora.

Nesse fluxo de recuperação, a resposta retornada pela API é idêntica independentemente de o endereço de e-mail informado existir ou não na base de dados, o que evita que a aplicação revele, a partir das respostas da API, quais endereços de e-mail estão cadastrados no sistema, prática conhecida como proteção contra enumeração de usuários, alinhada ao princípio de proteção de dados pessoais previsto no Artigo 46 da LGPD (Brasil, 2018).

4.3 Armazenamento de credenciais

Em conformidade com a recomendação de que senhas não sejam armazenadas em formato legível (Provos; Mazières, 1999), a aplicação não realiza o armazenamento direto das senhas dos usuários em sua própria base de dados. O cadastro e a verificação das credenciais são delegados ao serviço de autenticação do Supabase, responsável pelo armazenamento das senhas por meio de hash criptográfico gerado pelo algoritmo bcrypt (a coluna encrypted_password do Supabase mantém, por razões históricas, um nome que sugere cifra reversível, embora o valor armazenado seja, na prática, um hash irreversível). Ainda que a senha em texto claro trafegue pela aplicação durante as operações de cadastro e login, antes de ser repassada ao Supabase, ela não é persistida em nenhuma tabela própria da aplicação. Dessa forma, mesmo em caso de acesso indevido às tabelas da aplicação, as senhas dos usuários não ficam expostas em texto claro. Ressalva-se, no entanto, que essa passagem da senha pela aplicação exige cuidados adicionais, como evitar seu registro acidental em logs de requisição ou em mensagens de erro, e garantir o uso de TLS em ambiente de produção, pontos que não foram objeto de verificação neste trabalho.

4.4 Tokens de autenticação e gerenciamento de sessões

Após a autenticação bem-sucedida, a aplicação gera um token no formato JSON Web Token (JWT), protegido por HMAC (algoritmo HS256) com uma chave secreta mantida em variável de ambiente e não incluída no controle de versão do projeto; por se tratar de chave simétrica, trata-se de um código de autenticação de mensagem, e não de uma assinatura digital no sentido criptográfico estrito (que pressupõe par de chaves assimétricas, como em RS256 ou ES256). Cabe ressaltar também que o payload do JWT é apenas codificado em Base64URL, e não cifrado: qualquer pessoa que obtenha o token é capaz de ler seu conteúdo, razão pela qual nenhum dado pessoal é incluído nas claims do token, em conformidade com o Artigo 46 da LGPD. O token contém o identificador do usuário e um identificador de sessão, com validade de sete dias. Para o gerenciamento de sessões, foi implementado um controle de sessão única por conta/usuário: a cada novo login, um novo identificador de sessão é gerado e registrado na base de dados, substituindo o identificador anterior. Quando uma requisição é realizada com um token cujo identificador de sessão não corresponde mais ao identificador armazenado, o acesso é negado com o código de retorno SESSION_REPLACED, e o usuário é informado de que sua conta foi acessada em outro dispositivo.

A validade de sete dias definida para o token está em conformidade com as diretrizes do NIST SP 800-63B (National Institute of Standards and Technology, 2017), que recomenda, para aplicações no nível de garantia AAL1, um período máximo de reautenticação de até trinta dias; o intervalo adotado é, portanto, mais restritivo que o limite recomendado, buscando reduzir a janela de exposição em caso de comprometimento do token sem impor reautenticações excessivamente frequentes ao usuário.

Quanto ao controle de sessão única, a decisão de invalidar tokens anteriores a cada novo login prioriza a segurança em detrimento da conveniência. Um usuário autenticado simultaneamente em mais de um dispositivo, como um smartphone e um computador, tem a sessão mais antiga automaticamente encerrada ao efetuar login em um novo dispositivo. Essa limitação foi adotada para reduzir o risco de uso indevido de tokens obtidos por meio de vazamento ou do comprometimento de dispositivos, embora isso impeça a manutenção de múltiplas sessões ativas ao

mesmo tempo. Como direção para trabalhos futuros, esse mecanismo poderá ser substituído por um controle de múltiplas sessões identificadas por dispositivo, permitindo a revogação individual de cada sessão sem impedir o uso simultâneo em diferentes dispositivos.

4.5 Controle de acesso

O controle de acesso às funcionalidades da aplicação é realizado por meio de um middleware de autenticação aplicado às rotas que exigem usuário autenticado, como as relacionadas a rotinas de treino, exercícios, progresso físico, calendário, nutrição, perfil do usuário e envio de avaliações. Esse middleware verifica a presença e a validade do token enviado no cabeçalho de autorização da requisição e impede o acesso ao recurso solicitado em caso de token ausente, inválido, expirado ou correspondente a uma sessão substituída.

As rotas de administração, por sua vez, adotam um mecanismo de verificação próprio, baseado na comparação de uma senha enviada em um cabeçalho específico da requisição com uma senha definida em variável de ambiente do servidor. Embora esse mecanismo cumpra a função de restringir o acesso às funcionalidades administrativas, ele difere do padrão baseado em tokens utilizado para os demais usuários da aplicação, representando um ponto identificado para possível aprimoramento sob a perspectiva do princípio do menor privilégio (Saltzer; Schroeder, 1975).

4.6 Validação de entradas

A validação dos dados recebidos nas requisições da API foi identificada em parte das funcionalidades da aplicação. No módulo de feedback dos usuários (endpoint /feedback), a nota informada é validada quanto ao tipo e ao intervalo permitido, e os campos de texto possuem limite máximo de caracteres. No módulo de administração de academias, o identificador informado para a criação de uma nova academia é tratado e normalizado antes do armazenamento. Em outros módulos, como o de registro de dados de progresso físico e de rotinas de treino, os dados recebidos na requisição são processados sem uma camada explícita de validação de

tipos e formatos, o que foi confirmado pelos testes realizados e é discutido na seção seguinte.

4.7 Avaliação dos mecanismos de segurança implementados

Esta seção apresenta os resultados da avaliação técnica dos mecanismos de segurança implementados na aplicação, conduzida por meio de testes manuais sobre os endpoints da API. Os testes foram realizados com a ferramenta Postman e um script de automação em PowerShell para o teste de tentativas repetidas de login. Cada resultado foi documentado por meio de capturas de tela e analisado em relação às categorias do OWASP Top 10 (2021) e ao Artigo 46 da LGPD (Brasil, 2018). O Quadro 1 apresenta uma síntese dos resultados obtidos, detalhados nas subseções a seguir.

Quadro 1 – Resultados da avaliação dos mecanismos de segurança.

Mecanismo avaliado	Critério	Resultado	Evidência
Bloqueio de rota sem token	OWASP A01	Implementado	Figuras 2 e 3
Acesso admin sem credencial	OWASP A01	Implementado	Figura 4
Acesso com token válido	OWASP A01	Implementado	Figura 5
Sessão única por conta/usuário	OWASP A07	Implementado	Figura 6
Limitação de tentativas de login	OWASP A07 / NIST 800-63	Não implementado	Figura 7
Hash de senha (bcrypt)	OWASP A02	Implementado	Figuras 8 e 9
Validação em /feedback	OWASP A03	Implementado	Figuras 10 e 11
Validação em /progress	OWASP A03	Não implementado	Figura 12
Anti-enumeração de usuários	LGPD Art. 46	Implementado	Figuras 13 e 14
Token de verificação inválido	LGPD Art. 46	Implementado	Figura 15

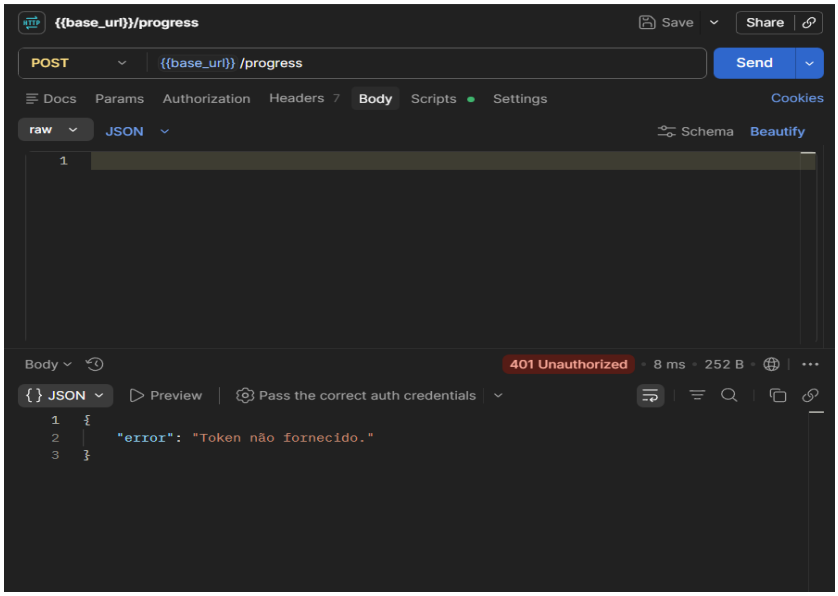
Fonte: autor (2026).

4.7.1 Broken Access Control (OWASP A01)

O primeiro grupo de testes verificou se as rotas protegidas da aplicação bloqueiam adequadamente requisições realizadas sem credenciais válidas. Foram realizadas requisições ao endpoint POST /progress sem o cabeçalho de autorização, resultando na resposta HTTP 401 com a mensagem "Token não fornecido", conforme ilustrado na Figura 2. Em seguida, foi testado o acesso ao endpoint GET /admin/stats sem o cabeçalho de senha administrativa, obtendo-se igualmente um retorno HTTP

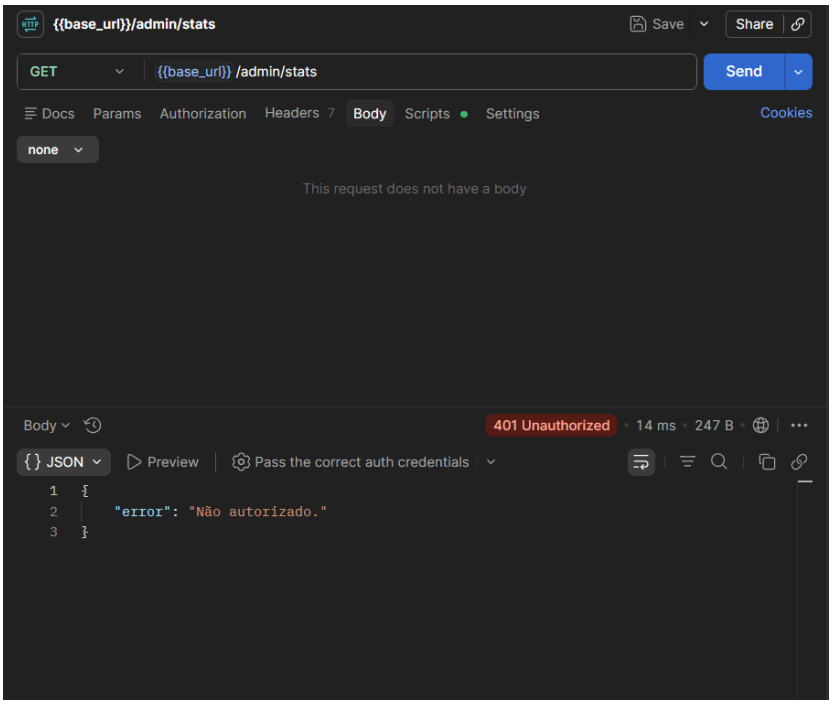
401 com a mensagem "Não autorizado" (Figura 3). Por fim, a mesma rota foi acessada com a senha correta, retornando HTTP 200 com os dados estatísticos do sistema (Figura 4). Foram também realizados testes com token de autenticação válido nas rotas de usuário, que retornaram os dados correspondentes com HTTP 200 (Figura 5). Os resultados indicam que o middleware de autenticação implementado cumpre sua função de bloquear requisições não autenticadas às rotas protegidas. Cabe ressaltar, contudo, que os testes realizados verificaram exclusivamente mecanismos de autenticação (se a requisição possui um token válido), e não mecanismos de autorização entre usuários já autenticados. A categoria Broken Access Control do OWASP Top 10 (2021) trata predominantemente de falhas desse segundo tipo, como o acesso indevido de um usuário autenticado a recursos pertencentes a outro usuário (referência direta insegura a objetos, ou IDOR) e a ausência de políticas de Row Level Security no banco de dados do Supabase, nenhuma das quais foi testada neste trabalho. A verificação desses cenários é indicada como extensão necessária da avaliação de controle de acesso em trabalhos futuros.

Figura 2 – Acesso sem token retorna HTTP 401.



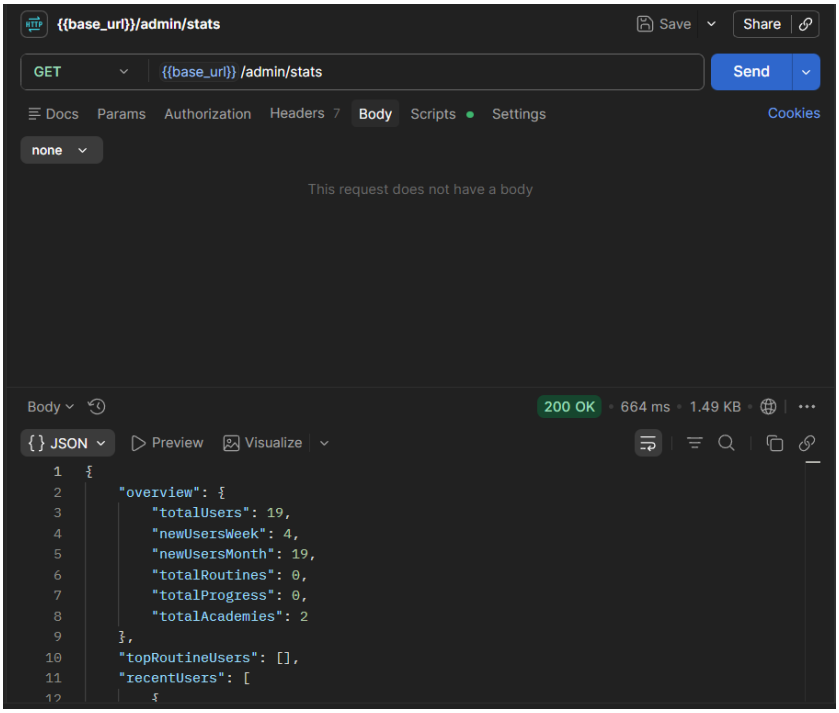
Fonte: autor (2026).

Figura 3 – Acesso à rota admin sem senha retorna HTTP 401.



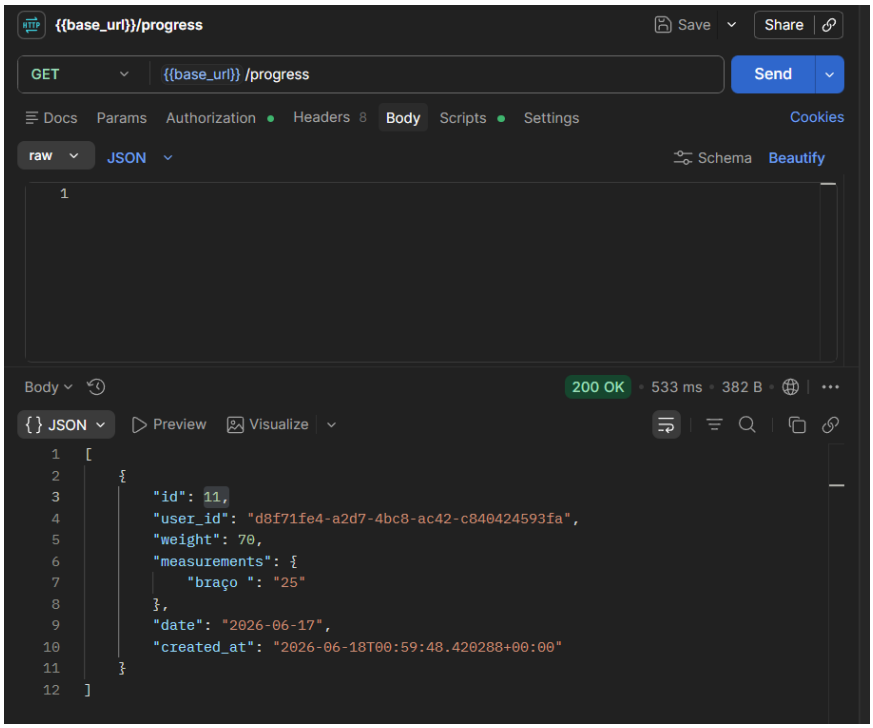
Fonte: autor (2026).

Figura 4 – Acesso à rota admin com senha correta retorna HTTP 200.



Fonte: autor (2026).

Figura 5 – Acesso com token válido retorna HTTP 200 com dados do usuário.



Fonte: autor (2026).

4.7.2 Identification and Authentication Failures (OWASP A07)

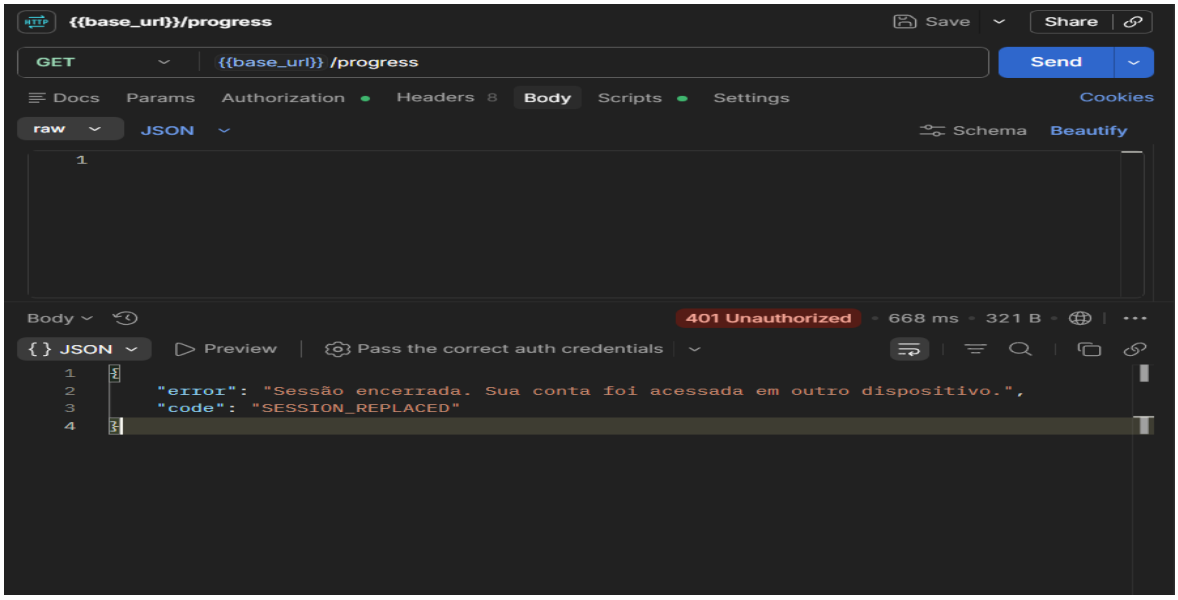
O segundo grupo de testes avaliou dois aspectos relacionados à autenticação: o mecanismo de sessão única por conta/usuário e a presença de proteção contra tentativas repetidas de login.

Para o teste de sessão única, foram realizados dois logins sucessivos com o mesmo usuário. Em seguida, o token gerado no primeiro login foi utilizado em uma requisição ao endpoint GET /progress. A aplicação retornou HTTP 401 com o código SESSION_REPLACED e a mensagem "Sessão encerrada. Sua conta foi acessada em outro dispositivo", conforme ilustrado na Figura 6. Esse comportamento confirma que o mecanismo de sessão única por conta/usuário está funcionando corretamente, invalidando tokens anteriores quando um novo login é realizado.

Para o teste de limitação de tentativas de login, foi elaborado um script em PowerShell que realizou dez requisições consecutivas ao endpoint POST /auth/login com senha incorreta. O resultado, apresentado na Figura 7, demonstrou que todas as dez tentativas retornaram HTTP 400 sem nenhum mecanismo de bloqueio ou atraso entre as requisições. A ausência desse controle representa uma lacuna identificada

em relação às diretrizes do NIST SP 800-63B (National Institute of Standards and Technology, 2017), que recomenda a implementação de proteção contra ataques de força bruta em sistemas de autenticação, e constitui um ponto de aprimoramento para versões futuras da aplicação.

Figura 6 – Token de sessão substituída retorna HTTP 401 com código SESSION_REPLACED.



Fonte: autor (2026).

Figura 7 – Dez tentativas de login com senha incorreta sem nenhum bloqueio.

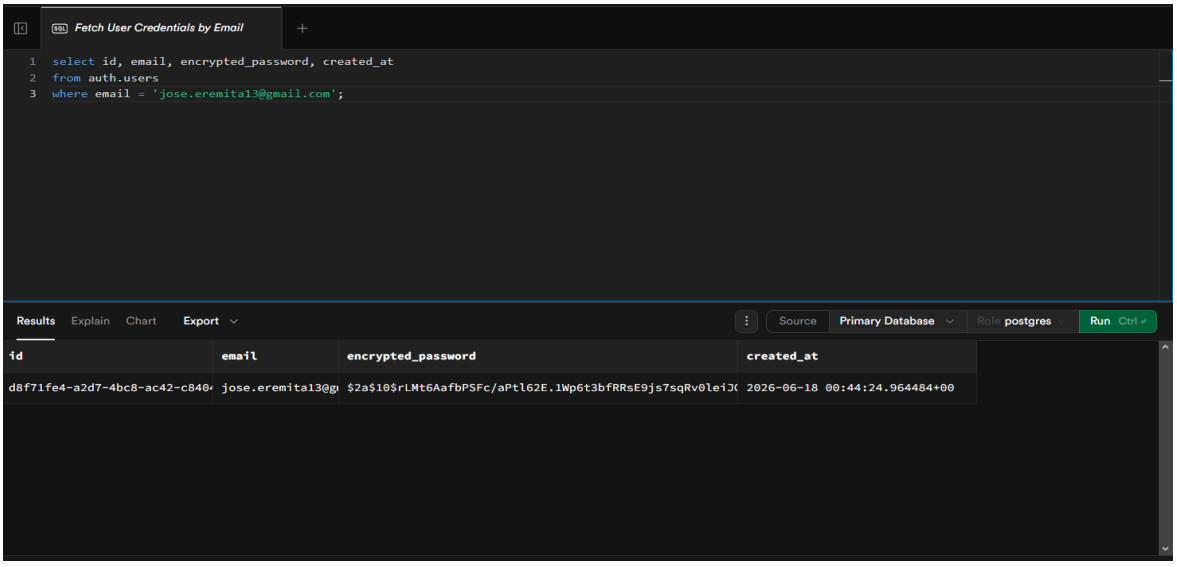
```
for ($i=1; $i -le 10; $i++) {
>>   $r = Invoke-WebRequest -Uri "http://localhost:3000/auth/login" `
>>     -Method POST `
>>     -ContentType "application/json" `
>>     -Body '{"email":"jose.eremita13@gmail.com","password":"senhaerrada999"}' `
>>     -SkipHttpErrorCheck
>>   Write-Host "Tentativa $i : $($r.StatusCode)"
>> }
Tentativa 1 : 400
Tentativa 2 : 400
Tentativa 3 : 400
Tentativa 4 : 400
Tentativa 5 : 400
Tentativa 6 : 400
Tentativa 7 : 400
Tentativa 8 : 400
Tentativa 9 : 400
Tentativa 10 : 400
```

Fonte: autor (2026).

4.7.3 Cryptographic Failures (OWASP A02)

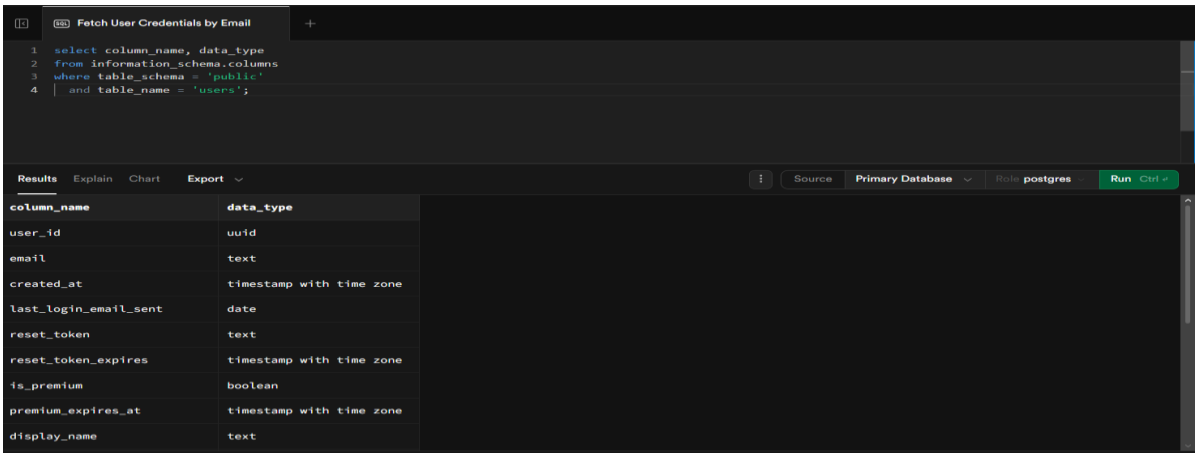
Para avaliar o armazenamento de credenciais, foram realizadas consultas diretas ao banco de dados por meio do editor SQL do Supabase. A primeira consulta, direcionada à tabela auth.users, confirmou que a coluna encrypted_password armazena as senhas em formato de hash bcrypt (identificado pelo prefixo $2a$10$), nunca em texto claro, conforme apresentado na Figura 8. A segunda consulta verificou a estrutura da tabela public.users, demonstrando a ausência de qualquer coluna destinada ao armazenamento de senhas (Figura 9). Esses resultados confirmam que a decisão arquitetural de delegar o gerenciamento de credenciais ao serviço de autenticação do Supabase resulta em conformidade com a recomendação de armazenamento seguro de senhas por meio de funções de hash criptográfico (Provos; Mazières, 1999), atendendo também às exigências de proteção técnica de dados pessoais previstas no Artigo 46 da LGPD (Brasil, 2018).

Figura 8 – Consulta SQL em auth.users confirma armazenamento de senha como hash bcrypt.



Fonte: autor (2026).

Figura 9 – Estrutura de public.users sem nenhuma coluna de senha.



Fonte: autor (2026).

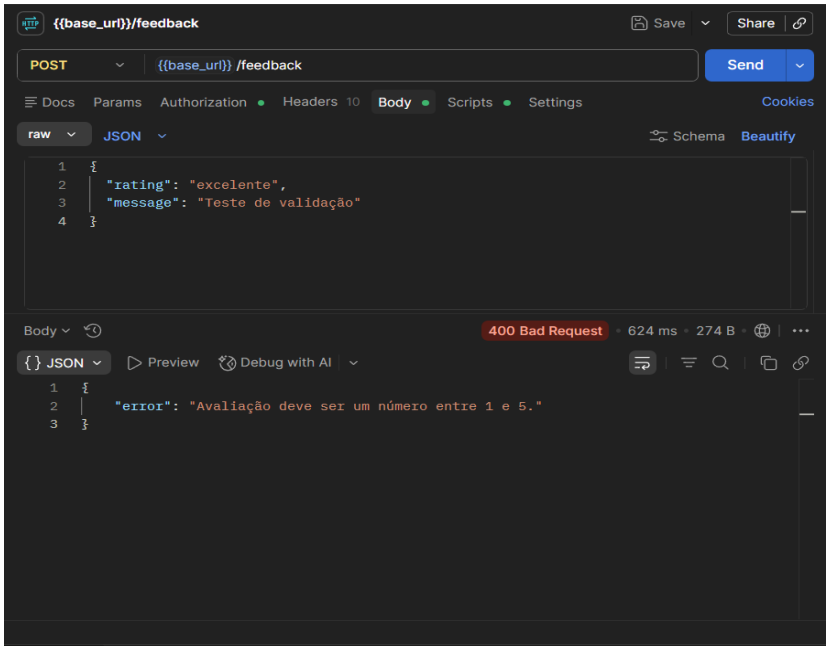
4.7.4 Injection e validação de entradas (OWASP A03)

Os testes de validação de entradas buscaram comparar o comportamento de rotas com e sem mecanismos explícitos de verificação de dados recebidos. No módulo de feedback, o envio de um campo rating com valor textual resultou em retorno HTTP 400 com a mensagem "Avaliação deve ser um número entre 1 e 5" (Figura 10), enquanto o envio de uma mensagem com mais de mil caracteres produziu o retorno HTTP 400 com a mensagem "Comentário deve ter no máximo 1000 caracteres" (Figura 11). Esses resultados demonstram que o módulo de feedback implementa validação de tipos e de limites de tamanho de forma adequada.

Em contraste, o envio de dados com tipos incorretos ao endpoint POST /progress, incluindo um campo de data com valor numérico inválido e um campo adicional não esperado pelo sistema, resultou em retorno HTTP 500 Internal Server Error, com a mensagem de erro do banco de dados exposta diretamente na resposta: "invalid input syntax for type date: 99999" (Figura 12). Esse comportamento evidencia

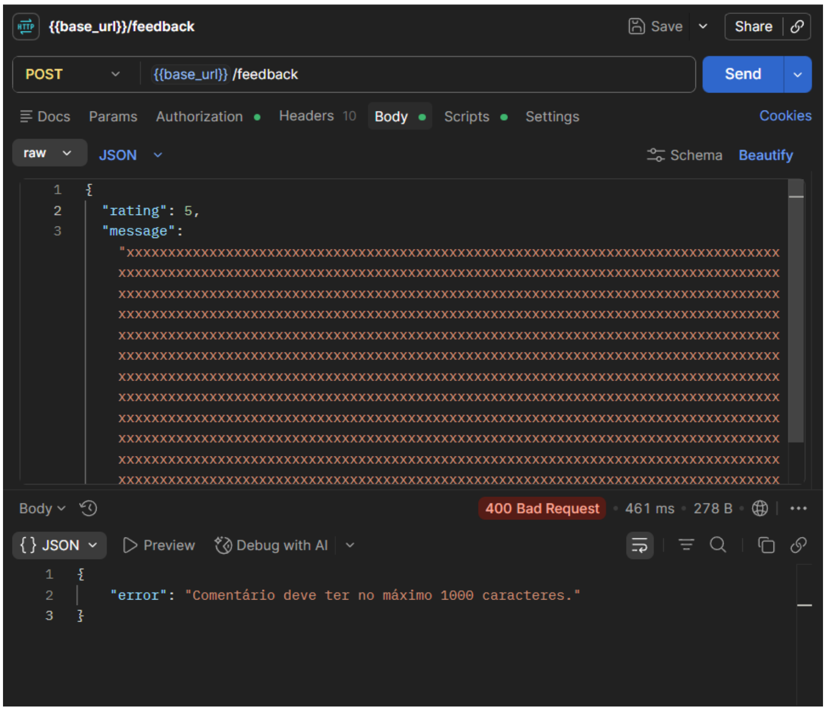
dois problemas: a ausência de validação de tipos na camada da aplicação antes da consulta ao banco de dados, e a exposição de detalhes internos da stack de tecnologia na resposta de erro, o que pode fornecer informações úteis para tentativas de exploração. Ambos os pontos se relacionam com a categoria Injection do OWASP Top 10 (2021) e com as recomendações de Viega e McGraw (2001) sobre o tratamento adequado de entradas em sistemas web.

Figura 10 – Feedback com rating inválido retorna HTTP 400 com mensagem descritiva.



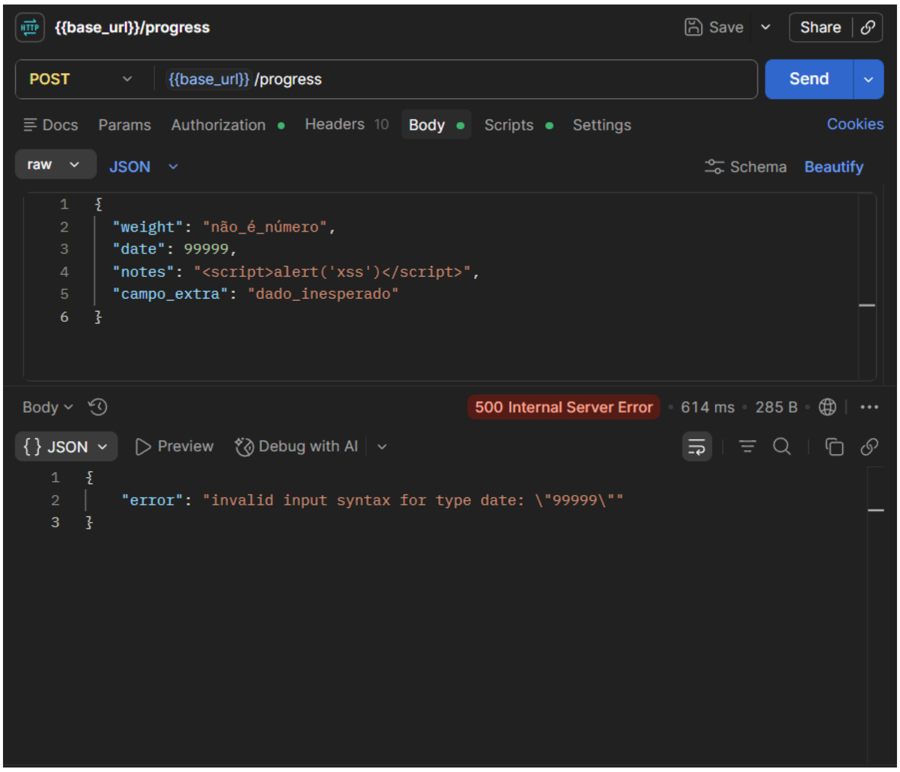
Fonte: autor (2026).

Figura 11 – Feedback com mensagem acima do limite retorna HTTP 400.



Fonte: autor (2026).

Figura 12 – Dados inválidos em /progress retornam HTTP 500 com erro interno do banco exposto.



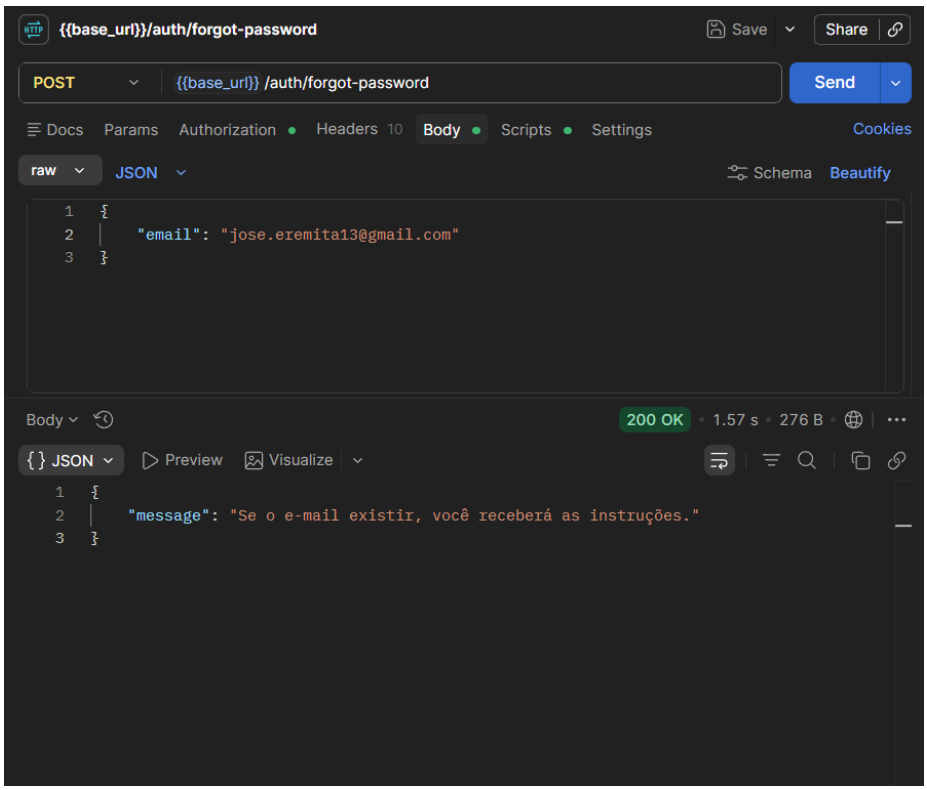
Fonte: autor (2026).

4.7.5 Proteção de dados pessoais e LGPD

Os testes relacionados à proteção de dados pessoais verificaram o comportamento do fluxo de recuperação de senha e a validade dos tokens de verificação de e-mail. Para o teste de anti-enumeração, foram realizadas duas requisições ao endpoint POST /auth/forgot-password: uma com um endereço de e-mail cadastrado na aplicação e outra com um endereço inexistente. Em ambos os casos, a resposta retornada foi idêntica, HTTP 200 com a mensagem "Se o e-mail existir, você receberá as instruções", conforme ilustrado nas Figuras 13 e 14. Esse comportamento impede que um atacante utilize as respostas da API para identificar quais endereços de e-mail estão cadastrados no sistema, protegendo os dados dos titulares em conformidade com o Artigo 46 da LGPD (Brasil, 2018).

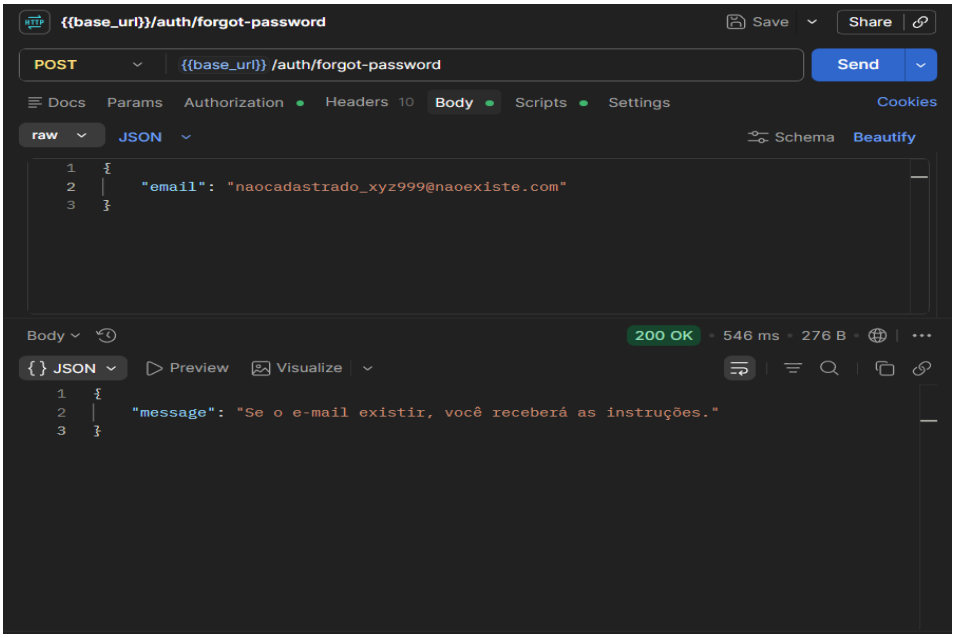
Adicionalmente, foi testada a tentativa de verificação de e-mail com um token inválido, enviado ao endpoint GET /auth/verify-email. A aplicação retornou HTTP 400 com a mensagem "Link de verificação inválido ou já utilizado" (Figura 15), confirmando que tokens não podem ser reutilizados e que a aplicação trata adequadamente tentativas de acesso com tokens forjados ou expirados.

Figura 13 – Forgot-password com e-mail cadastrado retorna mensagem genérica HTTP 200.



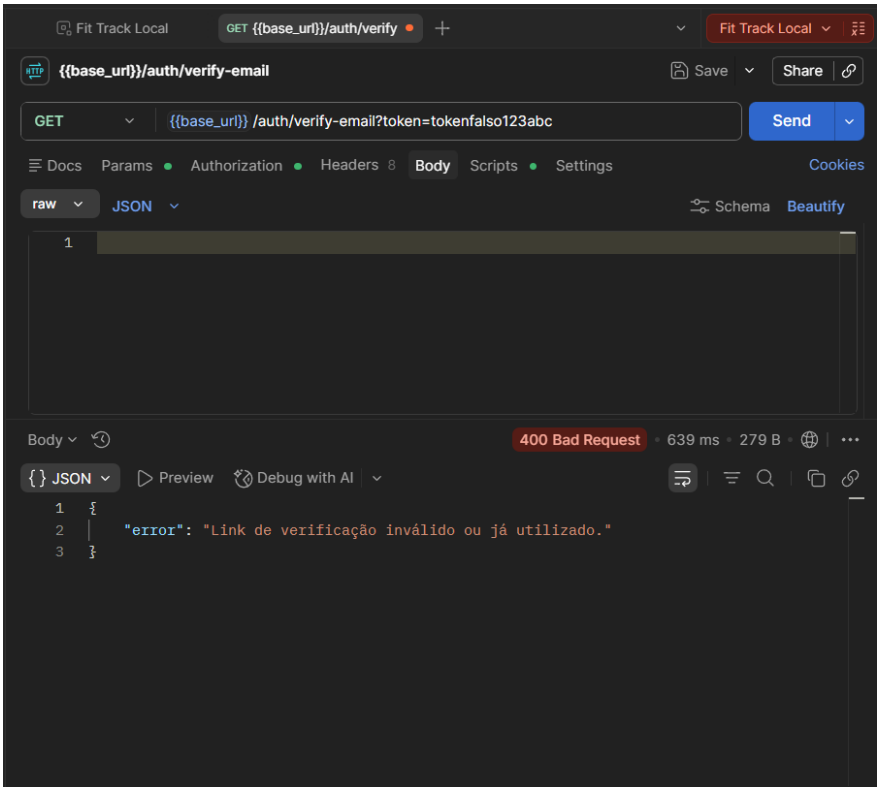
Fonte: autor (2026).

Figura 14 – Forgot-password com e-mail inexistente retorna resposta idêntica à Figura 13.



Fonte: autor (2026).

Figura 15 – Token de verificação inválido retorna HTTP 400.



Fonte: autor (2026).

5 CONSIDERAÇÕES FINAIS

Este trabalho teve como objetivo desenvolver uma aplicação web com mecanismos de autenticação, controle de acesso e proteção de credenciais, aplicando práticas de segurança recomendadas pela literatura e pela legislação vigente, e avaliando sua implementação por meio de testes técnicos sistemáticos. O sistema desenvolvido, estruturado em arquitetura de API REST com Node.js, Hono e Supabase, implementou os principais mecanismos propostos no projeto de pesquisa: verificação de e-mail no cadastro de usuários, autenticação por credenciais e por conta Google, geração de tokens JWT com controle de sessão única por conta/usuário, middleware de autenticação para as rotas protegidas e delegação do armazenamento de senhas, por meio de hash criptográfico, a um provedor de identidade especializado.

Os testes técnicos realizados indicaram que a maior parte dos mecanismos implementados funciona de forma consistente com o que foi proposto na literatura e na legislação: o controle de autenticação nas rotas protegidas, o mecanismo de sessão única, o armazenamento de credenciais por meio de hash criptográfico e a proteção contra enumeração de usuários no fluxo de recuperação de senha operaram conforme esperado. Ao mesmo tempo, a avaliação revelou lacunas relevantes, como a ausência de limitação de tentativas repetidas de login, a validação parcial de entradas em módulos como o de progresso físico, com exposição de erros internos do banco de dados, e o uso de um mecanismo de autenticação administrativa baseado em senha estática, divergente do padrão adotado para os demais usuários e contrário ao princípio do menor privilégio. Os resultados detalhados de cada teste, bem como sua relação com as categorias do OWASP Top 10 e com o Artigo 46 da LGPD, são apresentados na seção 4.7.

É importante ressaltar que a LGPD abrange aspectos mais amplos do que aqueles tratados neste trabalho, incluindo temas como gestão de consentimento, transparência no tratamento de dados, direitos dos titulares, retenção e descarte de informações e governança em privacidade. Embora a aplicação desenvolvida atenda a requisitos técnicos relacionados à segurança da informação, especialmente aos previstos no Artigo 46 da LGPD, ela não contempla todas as dimensões necessárias para uma conformidade plena com a legislação. Da mesma forma, é preciso

reconhecer que os mecanismos implementados não eliminam integralmente os riscos de segurança do sistema. A ausência de limitação de tentativas de login mantém a aplicação exposta a ataques de força bruta, enquanto a validação incompleta de entradas em determinados módulos representa uma superfície de ataque que ainda deverá ser tratada em versões futuras da aplicação. Da mesma forma, os testes relativos à categoria Broken Access Control cobriram apenas mecanismos de autenticação, não tendo sido verificados cenários de autorização entre usuários autenticados (como o acesso indevido a recursos de outro usuário) nem a configuração de políticas de Row Level Security no banco de dados do Supabase, o que limita o alcance das conclusões relatadas na seção 4.7.1. Por fim, o tratamento de dados como peso corporal e informações nutricionais, discutido na seção 2.2 como possível aproximação a dados sensíveis de saúde nos termos da LGPD, não foi aprofundado neste trabalho e permanece como uma lacuna de conformidade a ser tratada.

Constituem ainda ameaças à validade dos resultados apresentados: a condução de testes manuais e não repetidos, em um único ambiente de desenvolvimento local, sem uso de HTTPS; a ausência de varredura automatizada de vulnerabilidades e de teste de intrusão; a cobertura parcial dos endpoints disponíveis na aplicação; e o fato de os testes terem sido conduzidos pelo próprio desenvolvedor do sistema, o que introduz risco de viés do avaliador, mitigável em trabalhos futuros por meio de revisão por pares ou da adoção de um checklist independente, como o OWASP ASVS (Application Security Verification Standard).

De forma geral, o trabalho permitiu verificar na prática como mecanismos de autenticação, armazenamento seguro de credenciais e controle de acesso podem ser implementados em uma API REST moderna, contribuindo tanto para a discussão acadêmica sobre a aplicação prática de recomendações de segurança da informação quanto para o desenvolvimento de sistemas que necessitem atender a exigências da LGPD. Como direções para trabalhos futuros, destacam-se a implementação de limitação de tentativas de login, a padronização da validação de entradas em todos os módulos da aplicação, a revisão do mecanismo de autenticação das rotas administrativas e a ampliação do escopo de conformidade com a LGPD para além dos aspectos técnicos de segurança da informação, contemplando também os demais princípios e obrigações previstos na legislação.

REFERÊNCIAS

ANDERSON, Ross. **Security engineering: a guide to building dependable distributed systems**. 3. ed. Indianapolis: Wiley, 2020.

BIRYUKOV, Alex; DINU, Daniel; KHOVRATOVICH, Dmitry. **Argon2: new generation of memory-hard functions for password hashing and other applications.** In: IEEE EUROPEAN SYMPOSIUM ON SECURITY AND PRIVACY (EUROS&P), 2016, Saarbrücken. Proceedings [...]. Piscataway: IEEE, 2016. p. 292-302.

BRASIL. Lei nº 13.709, de 14 de agosto de 2018. **Lei Geral de Proteção de Dados Pessoais (LGPD)**. Diário Oficial da União: seção 1, Brasília, DF, 15 ago. 2018. Disponível em: http://www.planalto.gov.br. Acesso em: 17 jan. 2026.

CASTELLS, Manuel. **A sociedade em rede**. 17. ed. São Paulo: Paz e Terra, 2015.

CRESWELL, John W. **Research design: qualitative, quantitative and mixed methods approaches**. 4. ed. Thousand Oaks: Sage Publications, 2014.

ENISA. **ENISA Threat Landscape 2023**. Heraklion: European Union Agency for Cybersecurity, 2023. Disponível em: https://www.enisa.europa.eu/publications/enisa-threat-landscape-2023. Acesso em: 1 fev. 2026.

GIL, Antonio Carlos. **Métodos e técnicas de pesquisa social**. 6. ed. São Paulo: Atlas, 2008.

INTERNET ENGINEERING TASK FORCE. **RFC 7519: JSON Web Token (JWT)**. Fremont: IETF, 2015. Disponível em: https://datatracker.ietf.org/doc/html/rfc7519. Acesso em: 20 jan. 2026.

NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. **Digital identity guidelines: authentication and lifecycle management**. Gaithersburg: NIST, 2017. (NIST Special Publication 800-63B). Disponível em: https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-63b.pdf. Acesso em: 17 mar. 2026.

OWASP. **OWASP Top 10: the ten most critical web application security risks**. 2021. Disponível em: https://owasp.org/www-project-top-ten/. Acesso em: 15 jan. 2026.

PROVOS, Niels; MAZIÈRES, David. **A future-adaptable password scheme.** In: USENIX Annual Technical Conference. Proceedings, 1999. Disponível em: https://www.usenix.org/legacy/publications/library/proceedings/usenix99/full_papers/provos/provos.pdf. Acesso em: 17 mar. 2026.

SALTZER, Jerome H.; SCHROEDER, Michael D. **The protection of information in computer systems.** Proceedings of the IEEE, v. 63, n. 9, p. 1278–1308, 1975. Disponível em: https://ieeexplore.ieee.org/document/4768413. Acesso em: 17 mar. 2026.

STALLINGS, William. **Cryptography and network security: principles and practice**. 8. ed. Boston: Pearson, 2021.

VIEGA, John; MCGRAW, Gary. **Building secure software: how to avoid security problems the right way**. Boston: Addison-Wesley, 2001.