



Relatório de Identificação de Pontos Críticos no sistema MECRED

Autores

Richard Fernando Heise Ferreira
Eduardo Todt

UNIVERSIDADE FEDERAL DO PARANÁ
SETOR DE CIÊNCIAS EXATAS
DEPARTAMENTO DE INFORMÁTICA
CENTRO DE COMPUTAÇÃO CIENTÍFICA E
SOFTWARE LIVRE

Sumário

1	Introdução	2
2	Arquitetura	3
2.1	Atual	3
2.1.1	Frontend	3
2.1.2	Backend	4
2.1.3	Motor de busca: Elasticsearch	6
2.1.4	Armazenamento	6
3	Propostas de Melhoria	9
3.1	Nova Arquitetura	9
3.2	Backend	10
3.3	Armazenamento e Banco de Dados	10

1. Introdução

Este relatório tem como objetivo mostrar os pontos críticos da arquitetura que sustenta a plataforma MECRED, com o intuito de prover suporte a mudanças estruturais que garantam a continuidade da sua implementação, manutenção, evolução e aprimoramento ao longo do tempo. Além disso, este documento compõe parte do entregável correspondente ao Objetivo 2, Meta 3, do plano de trabalho estabelecido em parceria com o Ministério da Educação.

As análises aqui apresentadas baseiam-se no conhecimento acumulado ao longo dos 10 anos de existência da plataforma MECRED, aliado à experiência técnica da equipe do C3SL, composta por graduandos, pós-graduandos e docentes especialistas em suas respectivas áreas. Como resultado, identificou-se a necessidade de uma reestruturação completa da arquitetura do projeto, pois houveram mudanças significativas na tecnologia adotada anteriormente que obsoletaram a arquitetura atual; exigindo, portanto, um reprojeto das tecnologias, com exceção do frontend, cuja reimplementação foi realizada do zero entre fevereiro e junho de 2024.

2. Arquitetura

Esta seção descreve a arquitetura do sistema da MECRED, detalhando os componentes em operação, as versões dos softwares utilizados e a forma como esses sistemas se interconectam. Também são apresentadas breves explicações sobre as decisões tomadas ao longo do ciclo de vida do sistema.

2.1 Atual

A arquitetura atual pode ser resumida pela imagem 2.1.

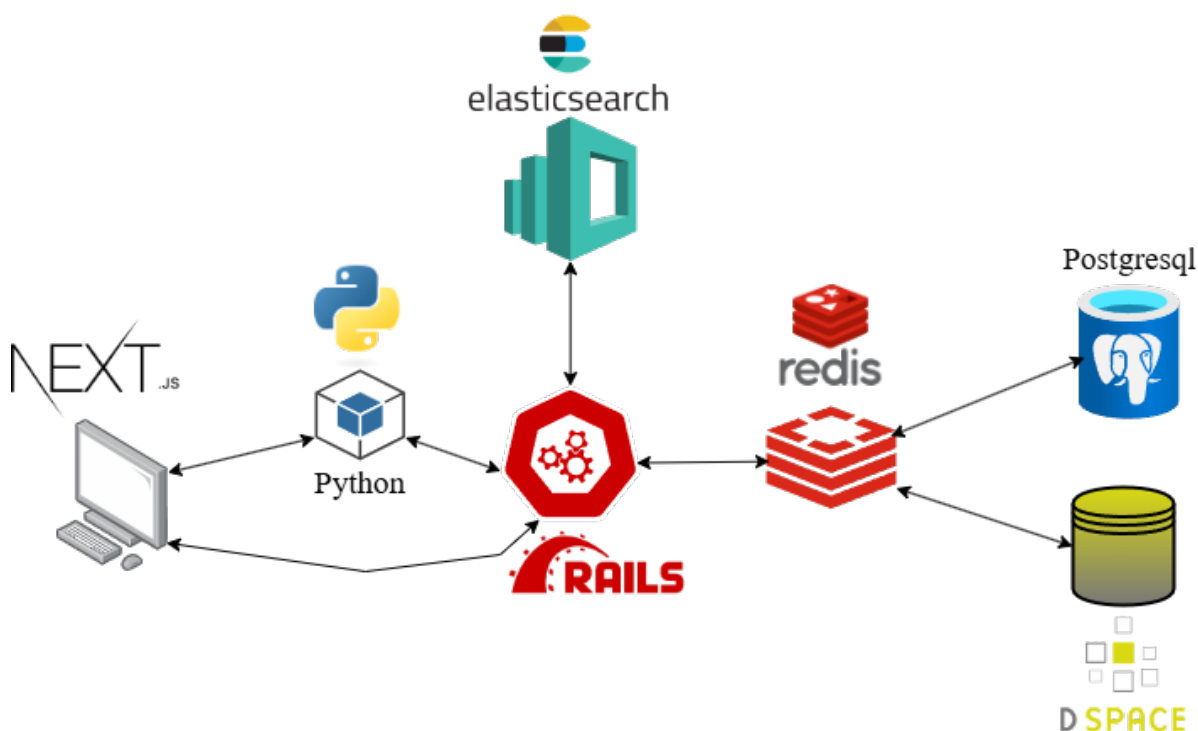


Figura 2.1: Arquitetura atual.

Cada componente será explicado nas subseções a seguir.

2.1.1 Frontend

Entre fevereiro e junho de 2024, a interface da MECRED passou por uma reestruturação significativa. A versão original era baseada na biblioteca JavaScript React (versão 16.14.0,

atualizado para a versão 18.2.0 para fins de manutenção em meados de 2022), mas os avanços recentes em tecnologias de renderização de imagens — um componente essencial para garantir escalabilidade e desempenho —, aliados às melhorias em sistemas de roteamento e renderização do lado do servidor (Server-Side Rendering, ou SSR), evidenciaram a necessidade de adotar um framework mais moderno e eficiente.

A renderização do lado do servidor se mostrou especialmente importante para proporcionar uma experiência mais responsiva e ágil, sobretudo em dispositivos móveis ou em ambientes com recursos computacionais limitados. Nesse contexto, o Next.js foi escolhido para substituir a abordagem anterior por oferecer suporte nativo a SSR, otimização automática de imagens, melhor desempenho em dispositivos móveis e uma estrutura de desenvolvimento mais robusta.

A familiaridade prévia da equipe com a biblioteca React – presente dentro do framework Next.js – amplamente utilizada em outros projetos do C3SL, também contribuiu para que a transição ocorresse de forma natural e eficiente.

A Tabela 2.1.1 apresenta um resumo das tecnologias principais utilizadas anteriormente. O código-fonte com as demais dependências pode ser consultado em https://gitlab.c3sl.ufpr.br/rfhferreira/cleaning-portal-mec-react/-/blob/Develop/package.json?ref_type=heads.

Já a tabela 2.1.2 apresenta um resumo das tecnologias principais utilizadas atualmente, utilizando Next.js. O código-fonte com as demais dependências pode ser consultado em https://gitlab.c3sl.ufpr.br/mecred/frontend-mecred/-/blob/develop/package.json?ref_type=heads.

Tecnologia	Versão
React	16.14.0
Node.js	17.4.0
npm	9.5.1

Tabela 2.1.1: Tecnologias utilizadas na versão anterior da arquitetura.

Tecnologia	Versão
Next.js	14.2.3
React	18.3.1
Node.js	21.6.1
pnpm	10.7.0

Tabela 2.1.2: Tecnologias utilizadas na versão atual da arquitetura.

Para fins técnicos, o deploy do build de ambos os frontends é realizado em contêineres, garantindo a reusabilidade do código e evitando erros críticos de versionamento que poderiam comprometer a manutenção e evolução do sistema. No entanto, o build da versão antiga depende do download de dependências legadas, gerando diversos avisos relacionados a código depreciado. Esse cenário não se repete nas tecnologias atualmente utilizadas, que oferecem um ambiente de desenvolvimento mais estável e compatível com as práticas modernas.

2.1.2 Backend

A API do sistema foi desenvolvida utilizando Ruby on Rails, um framework da linguagem Ruby com abordagem batteries included, bastante popular na década passada. Esse backend

vem sendo utilizado desde 2015, quando a MECRED foi criada, e desde então não passou por atualizações significativas — o que, à luz dos padrões atuais, torna a tecnologia consideravelmente defasada.

As principais dificuldades na manutenção desse sistema são agravadas pela ausência de domínio interno sobre a tecnologia: o C3SL não possui outros projetos em Ruby on Rails há pelo menos oito anos. Devido à alta rotatividade de bolsistas, majoritariamente estudantes de graduação, torna-se inviável e pouco eficiente manter uma tecnologia desatualizada como base lógica da rede social.

Além disso, entre 2018 e 2022, houve hiato no desenvolvimento da aplicação. Esse intervalo resultou em quatro anos de manutenção mínima, agravando a defasagem tecnológica e gerando uma demanda urgente por atualização — tarefa que se mostra especialmente complexa diante da idade e estrutura do código legado.

Além das dificuldades mencionadas, outras limitações foram identificadas: o código apresentou falhas ao tentar realizar o download de coleções no sistema, uma ação comum entre os usuários. Devido à complexidade da manutenção, foi necessário implementar um microserviço de download em Python, integrado diretamente ao frontend para ser acionado sempre que um download de coleção fosse solicitado. Além disso, outro microserviço foi desenvolvido para permitir o streaming de conteúdo no site, funcionalidade anteriormente inexistente, que restringia a reprodução online de vídeos e visualização de PDFs, prejudicando a experiência e usabilidade do sistema.

Outra dificuldade técnica envolveu o processo de deploy do sistema, que anteriormente era excessivamente complexo e pouco escalável. Em 2024, visando garantir maior manutenibilidade e estabilidade, foi implementado um contêiner para auxiliar nesse processo, o qual se mostrou útil ao manter o serviço operacional e facilitar reparos e possíveis atualizações pequenas. Ainda assim, há problemas específicos da tecnologia que, apesar dos esforços de depuração da equipe, exigem manutenção constante e esforço alocado em áreas que estruturalmente precisam de mudança.

No que tange à experiência do usuário, vale ressaltar que o Ruby on Rails foi criado antes da existência da API do gov.br, o que torna sua integração com o login via gov.br desafiadora. Embora existam bibliotecas como o Devise, amplamente utilizado para autenticação em aplicações Rails e utilizada no MECRED, a integração específica com o protocolo OAuth2 do gov.br requer adaptações adicionais. A equipe envidou esforços significativos para contornar essa questão, mas mesmo a implementação de um microserviço de login paralelo ao Rails mostrou-se insuficiente, devido à forma como o framework gerencia automaticamente o código e utiliza seu próprio ORM, o ActiveRecord, para operações de banco de dados. Portanto, a integração com serviços do governo, uma das ações mais interessantes e principais para o futuro do sistema, provou-se inviável na tecnologia atual.

Em suma, o backend desenvolvido em Ruby on Rails já não atende às necessidades futuras da arquitetura, devido à sua complexidade, instabilidade e obsolescência. Torna-se evidente a necessidade de reestruturar essa camada, adotando uma nova solução backend que seja mais robusta, moderna e bem documentada, capaz de oferecer maior estabilidade, facilidade de manutenção e suporte às demandas em constante evolução da aplicação. Para fins de documentação a tabela 2.1.3 contém um resumo das tecnologias e versões do backend. O código-fonte pode ser conferido em <https://gitlab.c3sl.ufpr.br/rfhferreira/cleanning-portalmec>.

Tecnologia	Versão
Ruby	3.1.0p0
Rails	7.0.4.1
Node.js	14.15.1

Tabela 2.1.3: Tecnologias utilizadas no backend atual da arquitetura.

2.1.3 Motor de busca: ElasticSearch

Elasticsearch é um mecanismo de busca e análise de dados, distribuído de código aberto, desenvolvido em Java e baseado na biblioteca Apache Lucene. Lançado em 2010, a ferramenta permite armazenar, pesquisar e analisar grandes volumes de dados em tempo real, oferecendo respostas rápidas às consultas. Ele utiliza uma estrutura baseada em documentos no formato JSON e fornece uma interface RESTful para interagir com os dados. Sua arquitetura distribuída permite escalabilidade horizontal, onde os índices são divididos em fragmentos (shards) que podem ter réplicas, garantindo alta disponibilidade e tolerância a falhas.

Na MECRED, por sua vez, foi implantada uma instância do ElasticSearch, distribuída, que realiza as buscas por conteúdos. Apesar de ser bem documentado, sua instalação e utilização é complexa e altamente especializada – seu concorrente principal de código aberto é o Solr, também cotado para ser utilizado no MECRED, mas no fim foi descartado pela maior dificuldade de integração com os demais componentes. Dessa forma, foram incubidos esforços para documentação interna e concluiu-se que sua utilização se mantém. A versão do serviço é a 8.6.2.

2.1.4 Armazenamento

O Redis é um sistema de armazenamento de dados em memória, de código aberto, baseado em uma estrutura chave-valor. Reconhecido por sua alta performance e baixa latência, é amplamente utilizado para cache, gerenciamento de sessões e controle de filas. No sistema, o Redis é empregado para enfileirar requisições mais lentas, garantindo desempenho consistente mesmo diante de múltiplos acessos simultâneos à mesma funcionalidade. Atualmente, está em uso a versão 7.0.11.

Além do Redis, duas outras tecnologias compõem a lógica de armazenamento: o sistema de gerenciamento de banco de dados PostgreSQL, responsável pelo armazenamento dos metadados, e o DSpace, uma solução de código aberto dedicada exclusivamente ao armazenamento dos recursos educacionais. Em termos práticos, quando um novo recurso é adicionado ao sistema, ele passa por três etapas: (1) é cacheado e enfileirado no Redis, (2) seus metadados — como título, autor, data de publicação e descrição — são registrados nas tabelas do PostgreSQL, e (3) o recurso propriamente dito — como um vídeo ou um arquivo PDF — é armazenado no DSpace.

Apesar da velocidade, disponibilidade e escalabilidade proporcionadas pela arquitetura atual, o uso do DSpace apresenta limitações significativas. Originalmente, o DSpace foi projetado como uma solução para criação de repositórios institucionais voltados à preservação digital de longo prazo, com forte ênfase na exposição de metadados por meio do protocolo OAI-PMH, o que facilita a interoperabilidade e a indexação por serviços externos.

No contexto da MECRED, embora o uso do DSpace tenha sido inicialmente vantajoso —

especialmente para permitir a indexação automatizada de conteúdo —, sua adoção acabou se tornando um ponto de gargalo. O sistema é mais complexo do que o necessário para os objetivos da aplicação atual, sendo subutilizado: na prática, ele funciona apenas como um repositório em nuvem para armazenar os recursos educacionais, sem explorar suas demais funcionalidades. Além disso, o DSpace opera, por padrão, sobre o Apache Tomcat, o que o torna menos compatível com a infraestrutura moderna adotada pelo C3SL, que utiliza majoritariamente o Nginx como servidor de proxy reverso, aliado a outras camadas de roteamento e segurança. Isso torna a manutenção do DSpace mais difícil.

Outro fator relevante é que, apesar de sua capacidade de indexação, essa funcionalidade não se mostra útil no contexto atual do sistema, que se baseia em um modelo mais dinâmico de criação, compartilhamento e curadoria de recursos pelos próprios usuários — diferentemente do modelo de preservação institucional que o DSpace pressupõe.

Por fim, vale destacar que a versão do DSpace atualmente em uso está desatualizada, e um processo de atualização representa riscos consideráveis à estabilidade do sistema, pois todos os recursos educacionais estão armazenados nesse sistema. É característico do DSpace sua dificuldade de migração de para uma nova versão.

Portanto, foi decidido substituir o DSpace por uma tecnologia mais confiável, moderna e compatível com o propósito do sistema.

Ainda no que diz respeito ao armazenamento, embora o PostgreSQL seja uma tecnologia robusta e amplamente reconhecida para gerenciamento de bancos de dados relacionais, a estrutura atual do banco de dados revelou-se redundante e problemática. Ao longo de mais de dez anos de evolução e reestruturações sucessivas, parte da lógica implementada no banco tornou-se confusa, pouco documentada e de difícil compreensão. Atualmente, o sistema conta com 69 tabelas, das quais aproximadamente 10 não possuem registros; além disso, algumas apresentam dependências circulares ou relacionamentos encadeados que violam princípios de normalização e boas práticas de modelagem relacional.

É importante destacar que todas essas tabelas foram geradas automaticamente por meio do ActiveRecord, o ORM do framework Ruby on Rails. Embora essa abordagem tenha acelerado o desenvolvimento nas fases iniciais do projeto, ela reduziu significativamente o controle granular da equipe sobre a modelagem dos dados, além de tornar a manutenção da lógica do banco complexa, propensa a erros e diretamente encadeada com o framework do backend.

Adicionalmente, análises mais refinadas realizadas no contexto de estudos de integração com mecanismos de aprendizado de máquina evidenciaram graves inconsistências de dados, como valores ausentes; ausência de referenciamento por chaves estrangeiras; inconsistência de tipos; presença de dados órfãos ou fantasmas e tabelas obsoletas, sem valor estatístico real. Diante desse cenário, concluiu-se que a estrutura atual do banco de dados representa um ponto crítico de falha e que uma reestruturação completa é necessária para garantir a integridade, escalabilidade e manutenibilidade dos componentes. Por fim, a tabela 2.1.4 possui as versões das tecnologias empregadas, para fins de resumo da seção.

Tecnologia	Versão
Redis	7.0.11
DSpace	6.4
PostgreSQL	16.6

Tabela 2.1.4: Tecnologias utilizadas no armazenamento atual do sistema.

3. Propostas de Melhoria

Este capítulo propõe uma série de melhorias identificadas para os pontos críticos descritos no capítulo anterior.

3.1 Nova Arquitetura

A nova arquitetura pode ser visualizada na Figura 3.1.

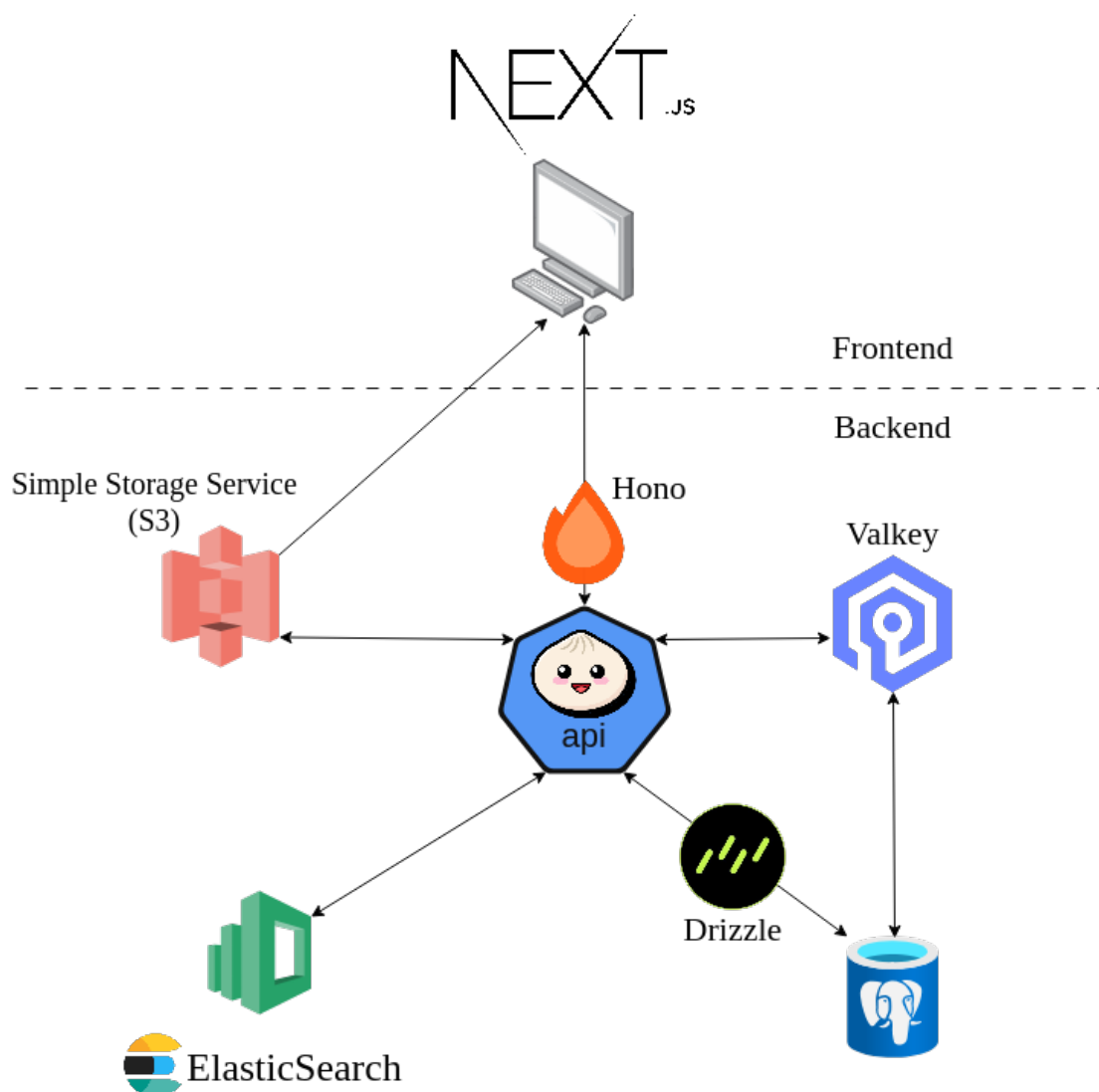


Figura 3.1: Nova proposta de arquitetura para o MECRED.

Cada componente será explicado nas subseções a seguir.

3.2 Backend

Propõe-se a reestruturação da API utilizando TypeScript, uma linguagem fortemente tipada baseada em JavaScript, que contribui para a redução de erros durante o desenvolvimento. Essa reescrita será realizada com o uso do Bun.js, um motor de execução JavaScript moderno, de alto desempenho e em rápida ascensão na comunidade. Para o roteamento da aplicação, será empregado o framework Hono, uma solução leve e eficiente, que permite a integração fluida entre backend e frontend, além de facilitar a inclusão de middlewares de autenticação e a documentação automática da API por meio do Swagger e do OpenAPI — ferramentas amplamente utilizadas para padronizar e descrever, de forma clara e interativa, os endpoints de uma API.

Em síntese, a nova proposta baseia-se em um conjunto de tecnologias modernas e robustas, consideradas de ponta no desenvolvimento web atual. Essa escolha garante maior longevidade, escalabilidade e manutenibilidade para o sistema. Além disso, por se tratar de TypeScript — uma extensão do JavaScript —, há um compartilhamento de conhecimento entre frontend e backend, facilitando a manutenção mesmo por desenvolvedores com pouca experiência em desenvolvimento backend. Essa abordagem revela-se especialmente adequada à dinâmica rotativa do C3SL.

Para a camada de persistência, foi adotado o Drizzle ORM, uma ferramenta moderna e em crescente adoção. O Drizzle permite uma modelagem mais segura e consistente, integrando validadores internos que aumentam a confiabilidade do código e simplificam a construção de consultas complexas. Ademais, oferece suporte à automação de logs, contribuindo significativamente para a depuração e o monitoramento do sistema.

Como discutido na seção sobre o motor de busca, considera-se viável a manutenção do Elasticsearch.

3.3 Armazenamento e Banco de Dados

Com o objetivo de substituir o DSpace por uma solução de armazenamento em nuvem mais alinhada à realidade do sistema, optou-se pela utilização do S3 (Simple Storage Service) — uma interface de armazenamento amplamente adotada, originalmente desenvolvida pela Amazon, mas também disponível em implementações de código aberto, como a que é mantida em instância própria no laboratório. O S3 possibilita o armazenamento e recuperação de objetos (como vídeos, PDFs e outros recursos digitais) de forma distribuída, segura e escalável, utilizando requisições HTTP simples.

Além disso, o S3 permite o fornecimento direto desses recursos ao frontend via streaming, eliminando a necessidade de intermediários ou camadas adicionais na API. Essa abordagem reduz a complexidade da aplicação, melhora a performance na entrega de conteúdo e facilita a manutenção e o monitoramento do ambiente. Vale destacar que esse cenário de uso está alinhado com o propósito original do S3, diferentemente do DSpace, cuja função vinha sendo extrapolada.

O Redis será substituído pelo Valkey, um fork compatível criado a partir do Redis original, que recentemente deixou de ser totalmente open source. O Valkey mantém o mesmo comportamento, desempenho e estrutura do Redis, permitindo uma migração tranquila. A lógica de cache e enfileiramento de requisições lentas permanece inalterada, assegurando a continuidade do desempenho otimizado do sistema.

No que diz respeito à base de dados, a tecnologia PostgreSQL será mantida, em razão de sua robustez e confiabilidade. No entanto, está sendo realizada uma reestruturação completa do banco. Parte significativa desse trabalho já foi concluída: das 69 tabelas originais, a estrutura foi reduzida para 35 tabelas normalizadas, eliminando redundâncias e inconsistências que dificultavam o desenvolvimento e comprometiam a integridade dos dados.

O principal desafio dessa etapa está na migração dos dados da estrutura antiga para o novo modelo, um processo delicado que requer atenção especializada para garantir integridade, consistência e fidelidade das informações. Essa atividade está sendo conduzida por uma equipe qualificada de Banco de Dados do C3SL, composta por estudantes de graduação, pós-graduação e doutorado. No momento da redação deste relatório, o novo modelo já se encontra consolidado e a migração dos dados da base antiga encontra-se em andamento.

Com essas mudanças, o sistema passa a contar com uma arquitetura mais moderna, eficiente e sustentável, alinhada às melhores práticas de desenvolvimento e gestão de dados. A substituição do Redis pelo Valkey, a manutenção do PostgreSQL com uma estrutura aprimorada e o cuidado com a integridade na migração evidenciam o compromisso do C3SL com a qualidade, escalabilidade e longevidade da solução desenvolvida. Essas melhorias estruturais não apenas resolvem gargalos históricos, como também preparam o ambiente para a evolução contínua do sistema nos próximos anos.

Tecnologia	Versão
Next.js	14.2.3
React	18.3.1
Node.js	21.6.1
pnpm	10.7.0
Bun.js	1.2.7
Hono	4.4.8
Drizzle ORM	0.31.2
PostgreSQL	16.6
Valkey	8.1.0
ElasticSearch	8.17.4

Tabela 3.3.1: Tecnologias utilizadas na nova proposta de sistema.