Luminos: Uma Plataforma Web de Gestão Pessoal com Foco em Clareza Mental

David Lucas Rodrigues Feitosa[1], João Arthur Rodrigues Pontes[2], João Igor de Sousa[3] Ferro, José Gabriel Soares dos Santos[4], Vinicius Rolim Aguiar Ibiapina[5]

[1]IFCE – Instituto Federal de Educação, Ciência e Tecnologia do Ceará – Campus Tianguá, Tianguá/CE, Brasil (david.lucas07@aluno.ifce.edu.br)
[2]IFCE – Instituto Federal de Educação, Ciência e Tecnologia do Ceará – Campus Tianguá, Tianguá/CE, Brasil (arthur.pontes09@aluno.ifce.edu.br)
[3]IFCE – Instituto Federal de Educação, Ciência e Tecnologia do Ceará – Campus Tianguá, Tianguá/CE, Brasil (joao.igor08@aluno.ifce.edu.br)
[4]IFCE – Instituto Federal de Educação, Ciência e Tecnologia do Ceará – Campus Tianguá, Tianguá/CE, Brasil (jose.santos07@aluno.ifce.edu.br)
[5]IFCE – Instituto Federal de Educação, Ciência e Tecnologia do Ceará – Campus Tianguá, Tianguá/CE, Brasil (vinicius.ibiapina07@aluno.ifce.edu.br)

Resumo: Este artigo apresenta o desenvolvimento do sistema web Luminos, uma plataforma de gerenciamento de anotações voltada à organização pessoal de conteúdos, compromissos e registros. A ferramenta reúne, em um único ambiente, funcionalidades inspiradas em diferentes soluções de produtividade, priorizando uma interface simples, intuitiva e de fácil utilização, com o objetivo de tornar a gestão de informações mais rápida, acessível e eficiente.

Palavras-chave: Sistemas *Web*; Gestão Pessoal; Gerenciamento de Anotações; Clareza Mental; Experiência do Usuário.

INTRODUÇÃO

Este artigo apresenta o desenvolvimento do sistema web Luminos, uma ferramenta de gestão pessoal desenvolvida para centralizar registros, compromissos e informações em uma interface intuitiva voltada à organização pessoal.

A principal motivação para este trabalho surgiu da observação do contexto atual da sociedade. A crescente transformação digital decorrente do uso de tecnologias no cotidiano tornou o excesso de informações e de estímulos uma característica constante da vida moderna. Dados do King's College London destacam que, em pesquisa realizada com 2.093 usuários de smartphones, 50% dos participantes afirmaram não conseguir deixar de verificar o aparelho mesmo quando deveriam estar concentrados em outras atividades, enquanto 49% consideram que sua capacidade de atenção é menor do que no passado (King's College London, 2022). Esse cenário pode contribuir para dificuldades relacionadas à organização das informações, à manutenção de uma rotina consistente e à sobrecarga cognitiva.

Somado a esse contexto, observa-se uma grande variedade de ferramentas voltadas à gestão pessoal, desenvolvidas para mitigar essa problemática, como o Notion e o Google Keep. Entretanto, embora sejam úteis em suas respectivas finalidades,é possível identificar, de modo geral, duas abordagens predominantes entre essas ferramentas. O primeiro grupo corresponde às ferramentas especializadas, que atendem a um propósito específico, podendo levar o usuário a recorrer a outras aplicações para atender diferentes necessidades de organização. O segundo grupo é composto por ferramentas mais completas que, por reunirem um grande número de funcionalidades, podem demandar maior tempo de aprendizagem, dificultando a exploração de todos os recursos disponíveis. Independentemente do grupo, uma questão torna-se evidente: o usuário é submetido a um esforço cognitivo maior, seja pela necessidade de gerenciar diversas ferramentas para funções distintas, seja pelo tempo necessário para explorar todo o potencial de uma única plataforma. Dessa forma, observa-se a necessidade de soluções que conciliam centralização de funcionalidades, simplicidade de uso e baixa carga cognitiva.

Nesse contexto, foi desenvolvido o Luminos, um sistema web de gestão pessoal voltado à organização de registros por meio da centralização das principais funcionalidades necessárias ao gerenciamento pessoal. Seu principal objetivo é oferecer uma

ferramenta baseada em uma interface limpa e centrada na experiência do usuário, tornando-se um auxílio para a organização cotidiana, e não mais uma tecnologia que demande esforço adicional de aprendizagem.

REFERENCIAL TEÓRICO

Esta seção apresenta os principais conceitos que fundamentam o desenvolvimento do sistema Luminos, abordando aspectos relacionados aos sistemas de organização pessoal, ao desenvolvimento de aplicações web, às arquiteturas modernas de software e ao armazenamento de dados em bancos relacionais.

Sistemas de Organização Pessoal

O avanço das tecnologias digitais modificou significativamente a forma como as pessoas organizam suas atividades acadêmicas, profissionais e pessoais. Atualmente, grande parte das informações do cotidiano é armazenada em ambientes digitais, tornando indispensável a utilização de ferramentas capazes de auxiliar no gerenciamento de tarefas, compromissos e registros pessoais. Segundo Allen (2015), a organização sistemática das informações reduz a sobrecarga mental, permitindo que o indivíduo concentre seus esforços na execução das atividades em vez de despender recursos cognitivos tentando lembrar de compromissos e pendências.

Entretanto, a grande quantidade de aplicações disponíveis para produtividade apresenta desafios relacionados tanto à fragmentação das informações entre diferentes plataformas quanto à elevada complexidade de algumas soluções. Conforme Norman (2013), sistemas digitais devem ser desenvolvidos considerando princípios de usabilidade e experiência do usuário, de forma que a interface seja intuitiva e reduza o esforço necessário para sua utilização. Nesse contexto, aplicações que centralizam funcionalidades essenciais e apresentam interfaces simplificadas tendem a proporcionar uma experiência mais eficiente aos usuários.

Desenvolvimento de Sistemas Web

Os sistemas de informação baseados na web consolidaram-se como uma das principais formas de disponibilização de serviços digitais devido à facilidade de acesso e manutenção. Diferentemente das aplicações desktop, sistemas web podem ser utilizados diretamente por meio de navegadores, independentemente do sistema operacional do dispositivo, reduzindo custos de instalação e atualização (Pressman; Maxim, 2021).

Outro aspecto relevante refere-se à arquitetura cliente-servidor, na qual a interface do usuário comunica-se com um servidor responsável pelo processamento das regras de negócio e pelo acesso aos dados armazenados. Essa separação favorece a escalabilidade, a manutenção e a evolução do software ao longo do tempo (Sommerville, 2018).

Com a evolução das aplicações web, tornou-se comum a utilização de arquiteturas desacopladas, nas quais o Front-end e o Back-end comunicam-se por meio de APIs (Application Programming Interfaces). Segundo Fielding (2000), o estilo arquitetural REST (Representational State Transfer) estabelece princípios que tornam essa comunicação mais simples, padronizada e escalável, sendo atualmente amplamente empregado no desenvolvimento de aplicações web modernas.

Tecnologias para Aplicações Web Modernas

O desenvolvimento de interfaces modernas tem sido amplamente realizado utilizando bibliotecas JavaScript como o React, cuja arquitetura baseada em componentes favorece a reutilização de código, a organização da aplicação e a construção de interfaces dinâmicas (React, 2025). Essa abordagem contribui para melhorar a experiência do usuário e facilita a manutenção do software durante sua evolução.

No desenvolvimento do lado do servidor, frameworks como o Django destacam-se por fornecer recursos voltados à segurança, autenticação, gerenciamento de usuários e integração com bancos de dados relacionais. Além disso, sua arquitetura baseada no padrão Model-View-Template (MVT) promove maior organização do código e separação de responsabilidades entre os componentes da aplicação (Django Software Foundation, 2025).

Em relação ao armazenamento das informações, os bancos de dados relacionais continuam sendo amplamente utilizados devido à sua capacidade de garantir integridade, consistência e confiabilidade dos dados. Entre eles, destaca-se o PostgreSQL,

reconhecido por sua conformidade com os padrões SQL, estabilidade e elevado desempenho em aplicações de diferentes portes (Elmasri; Navathe, 2016).

Dessa forma, a integração entre tecnologias como React, Django, APIs REST e PostgreSQL constitui uma arquitetura moderna amplamente empregada no desenvolvimento de sistemas web, permitindo a construção de aplicações escaláveis, seguras e de fácil manutenção.

SOLUÇÃO IMPLEMENTADA

A solução implementada neste trabalho envolve o desenvolvimento da infraestrutura de back-end, da API (Application Programming Interface) e da arquitetura de banco de dados de um sistema web modular voltado à produtividade e organização acadêmica denominado Luminos. O foco principal do desenvolvimento esteve concentrado na garantia da persistência robusta dos dados, no isolamento de regras de negócio complexas e no fornecimento de endpoints assíncronos capazes de alimentar uma interface reativa no lado do cliente.

A metodologia de pesquisa aplicada para a concepção da infraestrutura seguiu os preceitos da Dissertação-Projeto (Ribeiro; Zabadal, 2010), viabilizada por meio da modelagem relacional, implementação em nuvem e teste de um protótipo funcional integrado em ambiente de desenvolvimento local.

Levantamento de Requisitos do Sistema

Para subsidiar a modelagem arquitetural e a criação das tabelas, estabeleceu-se um levantamento focado em requisitos funcionais (RF) de armazenamento e controle lógico (Cysneiros, 2004):

Como cadastro de usuários, armazenando apenas as credenciais após a criptografia, garantindo privacidade e segurança a dados sensíveis como senhas. A possibilidade de personalização de elementos dentro do sistema. Definição de prazos, atualização de anotações, entre outros...

Modelagem e Arquitetura de Dados

A arquitetura de dados adota o modelo relacional para garantir a consistência das informações e relacionamento lógico entre as tabelas. O mapeamento seguiu um Diagrama Entidade-Relacionamento que é amplamente utilizado pela sua dinamicidade e pela sua capacidade de simplificar a complexidade de lidar com dados. (Barboza e Freitas, 2018).

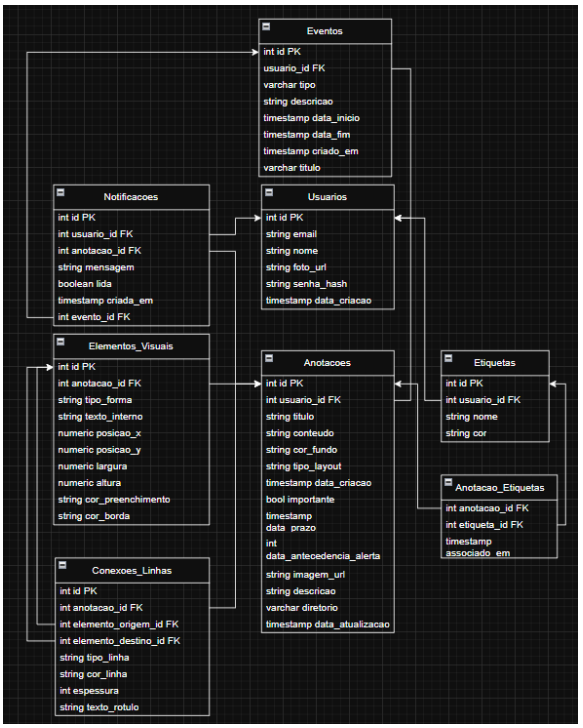


Figura 1: Diagrama E-R do Sistema Luminos (Fonte: Os Autores, 2026)

A entidade “usuário” tem ligações com diversas outras tabelas e, juntamente com a entidade “anotações”, forma a base do sistema, tanto que ambas foram as primeiras a ser implementadas. A partir delas, conseguimos elaborar e incrementar a estrutura deste modelo.

O banco conta com uma estrutura projetada para acomodar diversas funcionalidades de que o sistema precisa, como: adição de etiquetas, notificações, eventos, relações lógicas, entre outras. Essas ligações lógicas contribuem para a eficiência do sistema; como, por exemplo, o relacionamento muitos-para-muitos entre “anotações” e “etiquetas”, que permite que uma anotação possua mais de uma etiqueta e que uma etiqueta seja vinculada a mais de uma anotação.

Analisar desde os componentes necessários até a experiência do usuário ajuda na elaboração de um banco de dados eficiente, que atenda às necessidades do público e proporcione flexibilidade e espaço para o desenvolvimento dos programadores.

O banco de dados cresceu de forma proporcional à necessidade de se implementar novas funcionalidades. Também são aplicados conceitos

fundamentais, como o armazenamento apenas de URLs de imagens e de senhas em formato hash. Dessa forma, o banco não conhece a senha real do usuário, pois recebe apenas a sua versão criptografada.

O sistema conta com propriedades que evitam a permanência de dados soltos ocupando memória desnecessária. Um exemplo de caso de uso: se o usuário possui várias anotações e decide excluir sua conta (ou se ela é deletada por qualquer outro motivo), todas as tabelas que armazenam informações vinculadas a ele também são esvaziadas. Ou seja, os dados desse usuário são deletados assim que o sistema deixa de ter uma referência ativa para ele. Evitando uso desnecessário de memória.

Foi utilizado o Neon Console juntamente com o PostgreSQL para elaborar a parte de dados do sistema. O Neon Console é uma ferramenta de banco de dados baseada em nuvem que carrega instâncias do PostgreSQL de forma sem servidor, contando com um painel de controle onde você consegue gerenciar seu banco de dados PostgreSQL. Além disso, ele é o serviço que nos possibilita hospedar nosso banco de forma simplificada e acessível principalmente na fase de desenvolvimento e testes do software.

Protótipo do Back-end e Fluxo da API

No lado do servidor, utilizou-se o padrão de projeto MVT (Model-View-Template) adaptado para serviços de API. A camada de modelagem interage diretamente com o banco de dados PostgreSQL, garantindo a persistência dos estados gerados pelo usuário.

As requisições enviadas pela interface em React são interceptadas pelos controladores de rotas (Views), desenvolvidos sob decoradores @api_view do Django REST Framework. Os dados brutos passam por uma camada de serialização (Serializers) responsável por converter os relacionamentos complexos do banco em coleções JSON.

O fluxo das rotas foi otimizado: as funções api_cadastro e api_login gerenciam a segurança, enquanto a api_anotacoes realiza filtragens ativas diretamente em nível de banco de dados (query params). Interações do mapa mental acionam endpoints de persistência geométrica (api_salvar_elemento_visual e api_salvar_conexao), traduzindo dados visuais do front-end em dimensões cartesianas no PostgreSQL.

Desenvolvimento da Interface do Usuário

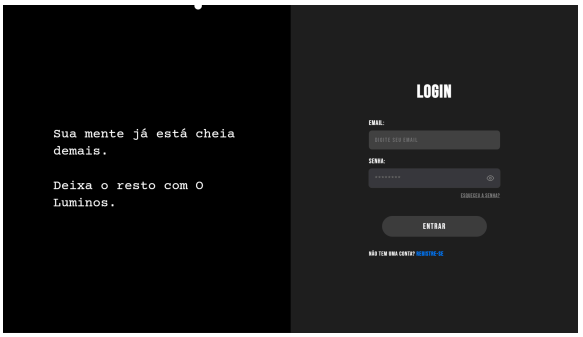
Paralelamente ao desenvolvimento da infraestrutura do sistema, foi implementada a interface do usuário do Luminos, responsável por proporcionar uma interação intuitiva com as funcionalidades disponibilizadas pela API. O processo iniciou-se com a elaboração de protótipos na plataforma Figma, permitindo definir previamente a identidade visual, os fluxos de navegação e a organização dos componentes antes da implementação do código.

A identidade visual foi projetada priorizando simplicidade, acessibilidade e conforto visual. Para isso, adotou-se uma interface em modo escuro, utilizando tons escuros como plano de fundo e cores de destaque para evidenciar informações importantes, reduzindo o cansaço visual durante períodos prolongados de utilização. A definição tipográfica buscou equilibrar estética e legibilidade, favorecendo a leitura e a navegação entre as funcionalidades do sistema.

Após a validação dos protótipos, iniciou-se a implementação da interface React. A aplicação foi estruturada seguindo o princípio da separação de responsabilidades, organizando componentes reutilizáveis, páginas e serviços responsáveis pela comunicação com a API desenvolvida em Django. Essa organização favoreceu a manutenção do código, a reutilização de componentes e a escalabilidade da aplicação ao longo do desenvolvimento do projeto.

A comunicação entre o front-end e o back-end foi realizada por meio de requisição à API REST do sistema, permitindo que todas as operações de autenticação, gerenciamento de usuários, criação de anotações, edição de conteúdos, organização por etiquetas e gerenciamento de eventos fossem executadas de forma dinâmica. Para manter o estado de autenticação do usuário, o identificador único retornado durante o processo de login é armazenado localmente no navegador é utilizado para autenticar as requisições subsequentes realizadas pela aplicação.

Entre as interfaces desenvolvidas destaca-se a tela de autenticação, responsável por permitir o cadastro de novos usuários e o acesso seguro às informações pessoais armazenadas no sistema. As Figuras 2 e 3 ilustram as telas de login e registro implementados.



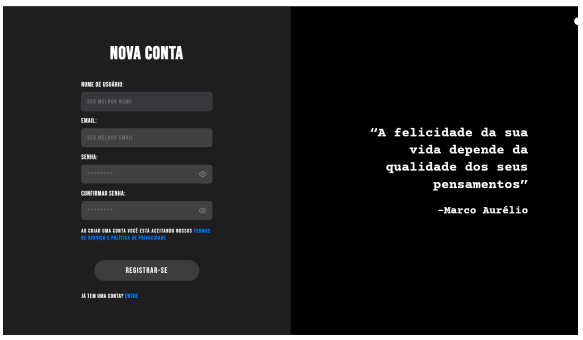


Figura 2 e 3: Telas de login e registro do Sistema Luminos (Fonte: Os Autores, 2026)

A tela inicial da aplicação corresponde ao painel principal, responsável por reunir as principais informações do usuário em um único ambiente. Nesse espaço são apresentadas as anotações recentes, compromissos cadastrados e uma saudação personalizada conforme o horário de acesso, oferecendo uma navegação simples e intuitiva. A Figura 4 apresenta essa interface.

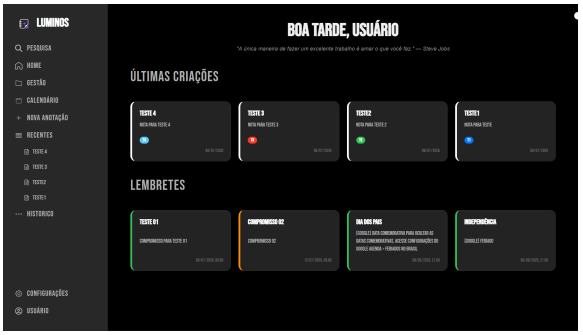


Figura 4: Tela de painel principal do Sistema Luminos (Fonte: Os Autores, 2026)

O Espaço de Criação representa o principal módulo do Luminos. Nele, o usuário pode produzir anotações completas contendo título, descrição e conteúdo textual, além de inserir imagens ilustrativas, organizar informações por meio de etiquetas personalizadas e construir mapas mentais utilizando elementos gráficos interligados. Essas funcionalidades permitem que diferentes formas de organização do conhecimento sejam utilizadas em um único ambiente, favorecendo tanto o planejamento de atividades quanto o registro de informações importantes. A Figura 4 apresenta essa funcionalidade.

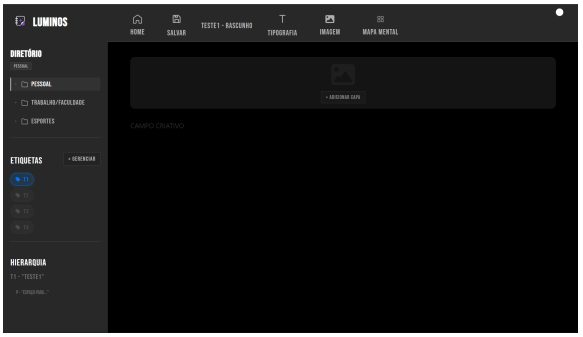


Figura 5: Tela de criação do Sistema Luminos (Fonte: Os Autores, 2026)

Além do gerenciamento de conteúdos, foi implementada uma área destinada às configurações da conta, permitindo ao usuário atualizar informações pessoais, como nome e senha, sem a necessidade de criar um novo cadastro. Essa funcionalidade contribui para a autonomia do usuário e para a manutenção da segurança de acesso à plataforma. A Figura 5 apresenta essa funcionalidade.

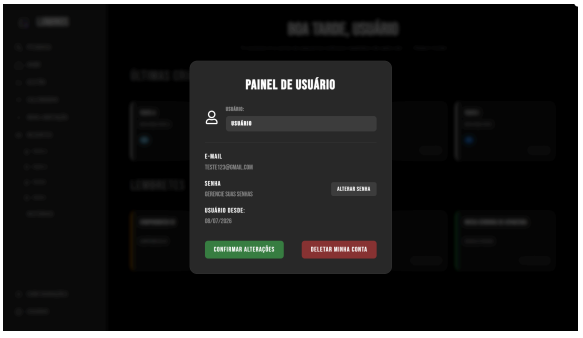


Figura 6: Tela de gerenciamento de conta do Sistema Luminos (Fonte: Os Autores, 2026)

Por fim, o sistema foi disponibilizado em ambiente de produção por meio da plataforma Render. O processo de implantação foi integrado ao repositório hospedado no GitHub, permitindo que novas versões da aplicação sejam publicadas automaticamente

sempre que alterações são incorporadas à ramificação principal do projeto.

Tecnologias Empregadas no Desenvolvimento

Para a consolidação das funcionalidades do servidor e persistência de dados, foram empregadas:

- **Linguagem Python e Django Framework:** Utilizados no desenvolvimento da lógica estrutural, processamento de regras de negócio e mapeamento ORM (Djangoproject.com, 2024).
- **Django REST Framework (DRF):** Empregado para a aceleração do desenvolvimento de APIs estáveis e controle de rotas.
- **PostgreSQL:** SGBD relacional robusto escolhido por sua capacidade de lidar com restrições lógicas e chaves estrangeiras complexas (Postgresql.org, 2024).
- **Neon (Serverless Postgres):** Plataforma de hospedagem em nuvem que separa computação de armazenamento, escalando sob demanda (Neon, 2026).
- **Django Auth Hashers:** Biblioteca utilizada no tratamento de segurança, convertendo senhas em hashes PBKDF2.
- **JavaScript** – linguagem utilizada no desenvolvimento da interface do usuário.
- **React** – biblioteca utilizada para construção da interface web, permitindo o desenvolvimento de componentes reutilizáveis e uma navegação dinâmica entre as telas.
- **Render** – plataforma utilizada para hospedagem da aplicação web e publicação das versões em produção.

CONCLUSÃO

O Luminos surgiu a partir da observação da necessidade de uma ferramenta que oferecesse uma experiência de usuário limpa e reunisse as principais funcionalidades da gestão pessoal em um único sistema, priorizando uma baixa carga cognitiva e uma reduzida curva de aprendizado, de modo a não se tornar mais uma ferramenta que exigisse um esforço adicional para sua utilização.

Durante o desenvolvimento, percebeu-se que a criação de uma identidade própria, de forma a evitar que o sistema se tornasse apenas uma réplica de ferramentas já existentes, o equilíbrio entre simplicidade e centralização de funcionalidades, a implantação da aplicação em um ambiente gratuito, a integração das diferentes tecnologias utilizadas e o cumprimento das metas de desenvolvimento em paralelo às atividades da graduação constituíram os principais desafios do projeto.

Como perspectivas futuras, pretende-se ampliar o conjunto de funcionalidades presentes no Luminos, permitindo ao usuário utilizar recursos como listas de afazeres, quadros Kanban, mapas mentais e *flashcards*. Além disso, pretende-se realizar avaliações da ferramenta em cenários reais, a fim de validar sua aplicação e orientar sua evolução com base nos resultados obtidos.

Por fim, acredita-se que o Luminos representa uma alternativa desenvolvida com foco na simplicidade, na centralização de funcionalidades e em uma baixa curva de aprendizado para usuários que enfrentam dificuldades relacionadas à organização das informações e ao excesso de estímulos presentes no cotidiano. Além de ter representado um exercício prático para a aplicação dos conhecimentos adquiridos ao longo da graduação, especialmente nas áreas de Engenharia de *Software*, desenvolvimento *web* e modelagem de banco de dados, o projeto possibilitou a construção de uma ferramenta que pode contribuir para a organização das atividades do dia a dia.

AGRADECIMENTOS

Os autores agradecem à Professora Cynthia Pinheiro Santiago pela orientação e pelo apoio durante o desenvolvimento deste trabalho.

REFERÊNCIAS

ALLEN, David. A arte de fazer acontecer: o método GTD – Getting Things Done. Rio de Janeiro: Elsevier, 2015.

BARBOZA, Fabrício Felipe Meleto; FREITAS, Pedro Henrique Chagas. Modelagem e desenvolvimento de banco de dados. SAGAH, Porto Alegre, 2018.

CYSNEIROS, L. Requisitos Não Funcionais: Da Elicitação ao Modelo Conceitual. PUC-Rio, Rio de Janeiro, 2004a.

CYSNEIROS, L. Requisitos Não Funcionais: Da Elicitação ao Modelo Conceitual. 2004.

DJANGO SOFTWARE FOUNDATION. Django Documentation. Disponível em: https://docs.djangoproject.com. Acesso em: 14 jul. 2026.

DJANGOPROJECT.COM. Django makes it easier to build better web apps more quickly and with less code. Django Software Foundation, Eletrônico, 2024.

DUFFY, Bobby; THAIN, Marion. Do we have your attention? How people focus and live in the modern information environment. London: King's College London, 2022. Disponível em: https://www.kcl.ac.uk/policy-institute/assets/how-people-focus-and-live-in-the-modern-information-environment.pdf. Acesso em: 14 jul. 2026.

ELMASRI, Ramez; NAVATHE, Shamkant B. Fundamentals of Database Systems. 7. ed. Boston: Pearson, 2016.

FIELDING, Roy Thomas. Architectural Styles and the Design of Network-based Software Architectures. 2000. Tese (Doutorado em Ciência da Computação) – University of California, Irvine, 2000.

GÓIS, A. S. Arquitetura de Sistemas Distribuídos e o Modelo Cliente-Servidor. Líder, Belo Horizonte, 2002a.

GÓIS, L. A. Implementação dos Serviços de Comunicação e Monitoração em um Ambiente para Desenvolvimento de Sistemas Distribuídos. Programa de Pós-Graduação em Ciência (Dissertação de Mestrado). UFSCar, São Paulo, 2002b.

NEON. Serverless Postgres Built for the Cloud. Neon Tech Documentation, Eletrônico, 2026.

NORMAN, Donald A. The Design of Everyday Things. Revised and Expanded Edition. New York: Basic Books, 2013.

PRESSMAN, Roger S.; MAXIM, Bruce R. Engenharia de Software: uma abordagem profissional. 9. ed. Porto Alegre: AMGH, 2021.

REACT. React Documentation. Disponível em: https://react.dev. Acesso em: 14 jul. 2026.

RIBEIRO, M. S.; ZABADAL, C. S. Metodologia de Pesquisa Aplicada à Computação: A Dissertação-Projeto. Sul Editores, Porto Alegre, 2010a.

RIBEIRO, V. G.; ZABADAL, J. R. Pesquisa em Computação. UniRitter, Porto Alegre, 2010b.

SOMMERVILLE, Ian. Engenharia de Software. 10. ed. São Paulo: Pearson, 2018.