Protótipo Funcional de um Portal para Armazenamento e Consulta de Trabalhos de Conclusão de Curso

José Diego Cosmo da Silva[1]

[1]Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, Campus Hortolândia - CEP 13183-091 – Hortolândia – SP – Brasil

j.diego@ifsp.edu.br

Abstract. *The final course projects from the Federal Institute of So Paulo, Hortolândia campus, are available on the institutional website in PDF format. Currently, these documents can be found through searches on the institutional website, using keywords related to the authors' names (students or faculty) or the project title. However, these filters are often insufficient to efficiently locate relevant works. To enhance document accessibility and optimize the search process, this study developed a functional prototype of a digital repository. The prototype introduces advanced filtering options, enabling searches based on specific criteria, key information, and broader queries, facilitating document retrieval.*

Resumo. *Os Trabalhos de Conclusão de Curso do Instituto Federal de São Paulo, campus Hortolândia, estão disponíveis no site institucional em formato PDF. Atualmente, esses documentos podem ser encontrados por meio de buscas realizadas diretamente no site institucional, utilizando palavras-chave relacionadas ao nome dos autores (discente ou docente) ou ao título do trabalho. No entanto, esses filtros nem sempre são suficientes para localizar um trabalho correlato de forma eficiente. Com o objetivo de melhorar o acesso a esses documentos e otimizar o processo de busca, este estudo desenvolveu um protótipo funcional de um repositório digital. O protótipo oferece novos filtros de pesquisa, permitindo consultas mais precisas baseadas em critérios específicos, informações-chave ou buscas generalistas, facilitando a recuperação dos documentos.*

1. Introdução

O Trabalho de Conclusão de Curso (TCC) constitui-se numa atividade curricular, de natureza técnico-científica, em campo de conhecimento que mantenha correlação direta com o curso do graduando, sendo um trabalho acadêmico de caráter obrigatório, que compõe a carga horária do curso (IFSP, 2024). Atualmente os TCCs do curso Tecnólogo em Análise de Desenvolvimento de Sistemas (ADS) ficam disponíveis no formato PDF no site do campus como pode ser visto na Figura 1.

Quando novos alunos começam uma nova pesquisa, é comum que os professores do curso solicitarem que os alunos acessem o portal do campus para leitura e análise de TCCs que já foram apresentados. Sendo assim, se deseja procurar um trabalho similar ou ainda sobre um tópico específico é necessário que algum professor do curso, que saiba sobre o referido tema, oriente o aluno na sua busca informando o nome do aluno ou o título do trabalho. Essa dificuldade faz também com que alunos do curso não acessem o repositório de TCCs do curso para aprenderem ou estudarem temas de interesse ou similares aos que estão pesquisando.

Considerando o avanço tecnológico que existe, a possibilidade de acesso a textos e busca em documentos, assim como os bancos de dados não relacionais, o objetivo deste trabalho foi a criação de um portal para os TCCs do Câmpus Hortolândia. O portal seria

similar a uma página do repositório cuja finalidade é centralizar todas as informações dos TCCs concluídos e facilitar o acesso à informação a esses trabalhos. Como objetivos secundários, tem-se a facilidade de ajudar na busca dos trabalhos por alunos que estão pesquisando sobre temas específicos e a criação de um novo tópico no site institucional do câmpus.

Figura 1. Tela do IFSP com os TCCs disponíveis (Instituto, 2024)

Este trabalho apresenta o Referencial Teórico na Seção 2, onde são abordados conceitos fundamentais para o desenvolvimento da aplicação, como requisitos funcionais e não funcionais, arquitetura de software (incluindo o padrão MVC), além da metodologia incremental. A Seção 3 os trabalhos correlatos são citados. A Seção 4 descreve a metodologia utilizada, detalhando o processo de levantamento de requisitos, prototipação e desenvolvimento. Na Seção 5, são descritos os incrementos implementados ao longo do projeto, destacando a construção das principais funcionalidades do sistema e as tecnologias utilizadas. Por fim, a Seção 6 apresenta a conclusão, onde são discutidos os resultados alcançados, as limitações encontradas e as possibilidades de trabalhos futuros relacionados ao sistema.

2. Referencial Teórico

Nesta Seção serão introduzidos os conceitos da arquitetura “Modelo-Visão-Controle” (MVC) utilizada no processo de desenvolvimento deste trabalho. O desenvolvimento *web* da aplicação foi criado com base na abordagem incremental para sua criação e a utilização como prototipação de um esboço inicial.

2.1. Requisitos Funcionais e Não Funcionais

Os requisitos funcionais representam as funcionalidades e serviços que um sistema deve fornecer, definindo suas operações principais e a forma como interage com os usuários. Segundo Pressman (2016), esses requisitos especificam o comportamento esperado do software, determinando as entradas, os processamentos e as saídas da aplicação. Eles incluem, por exemplo, a capacidade do sistema de armazenar dados, processar transações e gerar relatórios.

Por outro lado, os requisitos não funcionais descrevem as restrições e características de qualidade do sistema, como desempenho, segurança, usabilidade e portabilidade. Esses requisitos influenciam diretamente a experiência do usuário e a eficiência operacional do software. Pressman (2016) destaca que os requisitos não funcionais não determinam o que o

sistema faz, mas sim como ele deve operar, assegurando compatibilidade com diferentes plataformas, confiabilidade na execução e eficiência no uso de recursos computacionais.

Ao estabelecer requisitos funcionais e não funcionais bem definidos, o desenvolvimento da aplicação se torna mais estruturado e alinhado às necessidades dos usuários e às exigências técnicas do sistema (Pressman, 2016).

Exemplificação dos conceitos informados como não funcionais presentes são requisitos não funcionais ligados a desempenho: tempo de resposta, tempo de processamento, requisitos não funcionais de confiabilidade: disponibilidade, taxa de falhas, recuperação de falhas, requisitos de usabilidade: facilidade de uso, acessibilidade, documentação, requisitos não funcionais de Segurança: Confidencialidade, integridade, autenticidade, requisitos não funcionais de manutenibilidade: facilidade de correção, adaptabilidade, escalabilidade e requisitos não funcionais de portabilidade: capacidade de ser transferido para diferentes ambientes de hardware ou software (Glinz,2007).

2.2.Responsividade

A responsividade é fundamental no desenvolvimento *web*, garantindo que a interface se adapte dinamicamente a diferentes dispositivos e resoluções (Pressman, 2016). Neste projeto, foi implementada com Bootstrap, proporcionando ajuste automático para telas de dispositivos móveis, *tablets* e *desktops*, mantendo acessibilidade e usabilidade (Twbs, 2020).

Além do aspecto visual, a eficiência no processamento é essencial para a experiência do usuário. O Laravel otimiza o desempenho ao reduzir a carga no servidor por meio de roteamento eficiente e pré-processamento de consultas, diminuindo o tempo de resposta da aplicação (Otwell, 2011).

Ao combinar responsividade e otimização de processamento, a aplicação desenvolvida melhora a experiência do usuário, oferecendo maior fluidez, rapidez e eficiência operacional (Pressman, 2016).

2.3. Arquitetura de software

A arquitetura de software define a estrutura organizacional de um sistema, especificando seus componentes, interações e princípios de desenvolvimento (SOMMERVILLE, 2020). Dentro desse contexto, o modelo *Model-View-Controller* (MVC) (Figura 2) é amplamente utilizado em aplicações *web* devido à separação clara entre a lógica de negócio (Modelo), a interface do usuário (Visão) e o controle das interações do usuário (Controle) (Pressman, 2016).

A implementação do MVC no sistema desenvolvido seguiu as boas práticas preconizadas na literatura. O modelo gerencia a lógica de acesso aos dados, a visão exibe os resultados processados e o controlador atua como intermediário entre as solicitações do usuário e a resposta do sistema (Pressman, 2016).

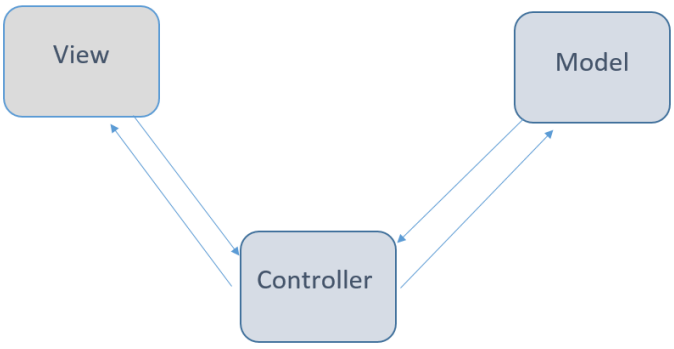

Figura 2. Arquitetura MVC (AppsLanka, 2023)

2.4. Modelo Incremental

O modelo incremental (Figura 3) é um método iterativo que divide o desenvolvimento em pequenas entregas progressivas, permitindo a adição contínua de funcionalidades ao sistema. Cada incremento amplia e aprimora o software, possibilitando a validação contínua dos requisitos e a incorporação de *feedback* ao longo do processo. Essa abordagem favorece entregas frequentes de versões utilizáveis do sistema, garantindo maior flexibilidade e adaptação às necessidades dos usuários (Pressman, 2016).

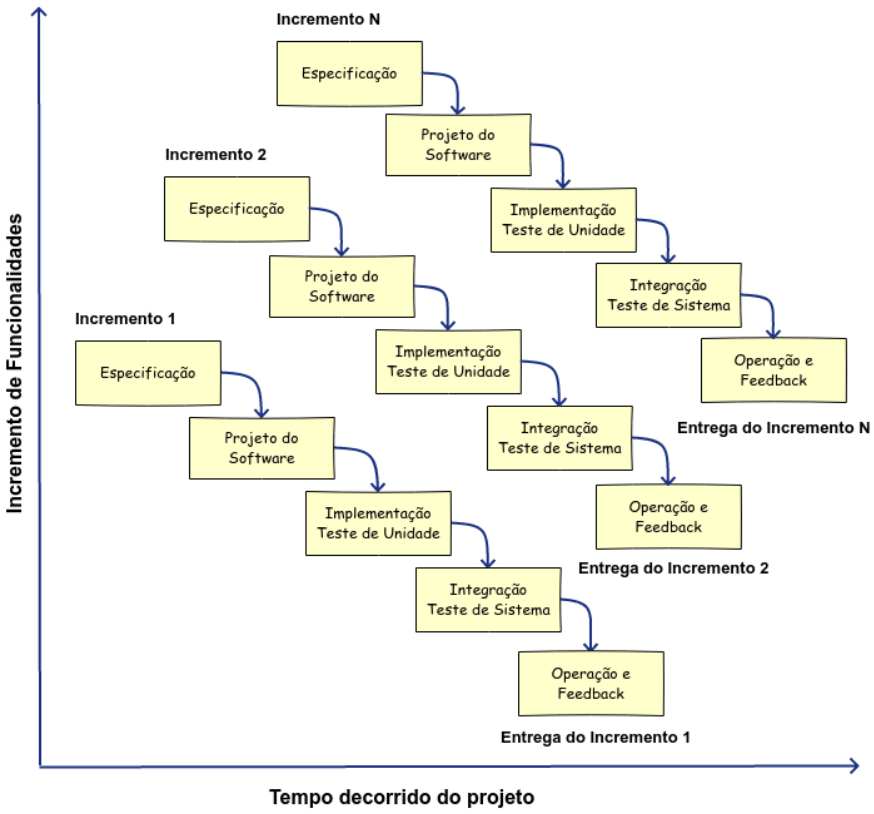

Figura 3. Exemplo do Modelo Incremental (Dias, 2019)

O modelo incremental é uma abordagem iterativa de desenvolvimento, onde o sistema é implementado em pequenos incrementos. Cada ciclo adiciona novas funcionalidades, permitindo testes e ajustes antes da entrega final (SOMMERVILLE, 2020).

Esse modelo foi aplicado no desenvolvimento do repositório digital de TCCs, possibilitando a evolução progressiva do sistema e garantindo maior flexibilidade e adaptação às necessidades dos usuários.

Na etapa de projeto, define-se a arquitetura do sistema, incluindo a organização dos componentes, a escolha de tecnologias e padrões de codificação. Em seguida, na implementação, o projeto é transformado em código, garantindo coesão entre os módulos e funcionalidade integrada. Testes unitários são aplicados para validar cada componente, assegurando conformidade com os requisitos estabelecidos.

Essas etapas garantem que o software seja desenvolvido de forma estruturada, mantendo qualidade e alinhamento com as expectativas definidas desde o início do processo.

3. Trabalhos Correlatos

A análise de plataformas *web* similares permitiu a identificação de requisitos essenciais para o desenvolvimento do repositório digital de TCCs. Foram estudadas as funcionalidades do

Google Acadêmico, ResearchGate e Academia.edu, que serviram de base para a implementação de busca, filtragem e organização dos documentos.

O Google Acadêmico influenciou a busca por termos-chave, autores e títulos; o ResearchGate contribuiu para a criação de filtros avançados por área do conhecimento e instituição; e o Academia.edu inspirou a categorização e exibição dos documentos.

Além das funcionalidades incorporadas, identificaram-se limitações nesses sistemas, motivando melhorias no repositório desenvolvido. Diferente do Google Acadêmico, que não permite *upload* direto de documentos, a plataforma proposta centraliza todos os TCCs internamente. Enquanto o ResearchGate exige cadastro para acesso, o repositório desenvolvido é aberto e sem restrições. Já o Academia.edu, que impõe barreiras pagas a algumas funcionalidades, foi adaptado para oferecer acesso gratuito e irrestrito aos trabalhos acadêmicos.

A análise desses sistemas permitiu a criação de um repositório otimizado, garantindo maior acessibilidade, eficiência na busca e democratização do conhecimento acadêmico.

3.1 Google Acadêmico

A plataforma Google Acadêmico (Figuras 4 e Figura 5) é amplamente utilizada por pesquisadores, estudantes universitários e profissionais de diversas áreas do conhecimento, permitindo o acesso e a contribuição para o avanço da ciência. Além de facilitar o acesso à literatura científica, o Google Acadêmico oferece ferramentas para criação de alertas de novas publicações relevantes e para a organização da biblioteca pessoal de artigos dos usuários (Alphabet, 2004).

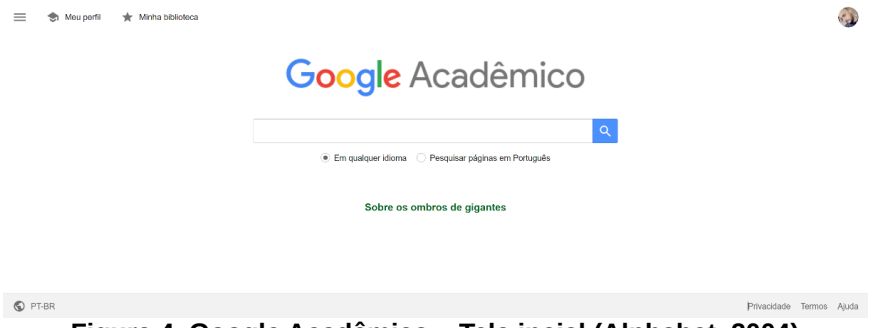

Figura 4. Google Acadêmico – Tela incial (Alphabet, 2004)

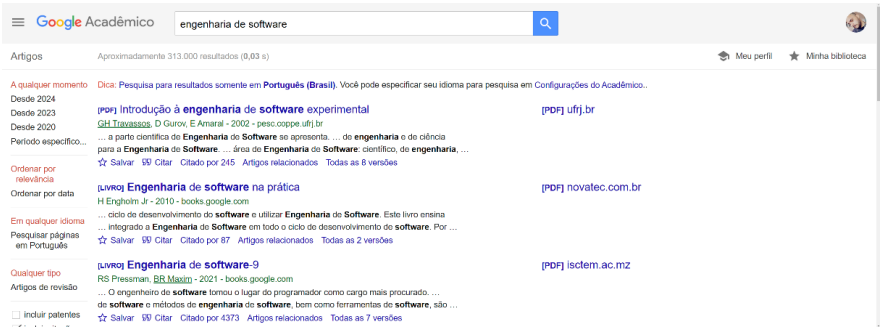

Figura 5. Google Acadêmico – Tela de pesquisa (Alphabet, 2004)

3.2. ResearchGate

O ResearchGate é uma plataforma online destinada à comunidade acadêmica e científica como mostrado nas figuras 6 e 7, foi projetada para facilitar o compartilhamento de pesquisas, colaboração entre pesquisadores e acesso a artigos científicos.

Esta plataforma permite que pesquisadores publiquem suas descobertas, submetam artigos revisados por pares e interajam com colegas de todo o mundo em seus respectivos campos de estudo. Além disso, oferece recursos para criação de perfis acadêmicos detalhados, onde os usuários podem listar suas publicações, projetos de pesquisa e interesses acadêmicos.

O ResearchGate também funciona como um repositório onde os pesquisadores podem disponibilizar versões pré-impressas ou pós-impressas de seus artigos, promovendo o acesso aberto à literatura científica. Com sua interface intuitiva e ferramentas de networking, a plataforma facilita a descoberta de novas colaborações e oportunidades de pesquisa entre os membros da comunidade científica global.

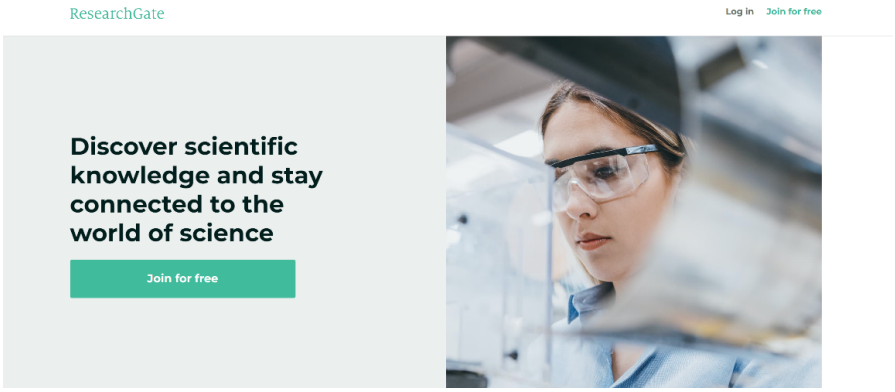

Figura 6. ResearchGate – Tela de início (Madisch, 2008)

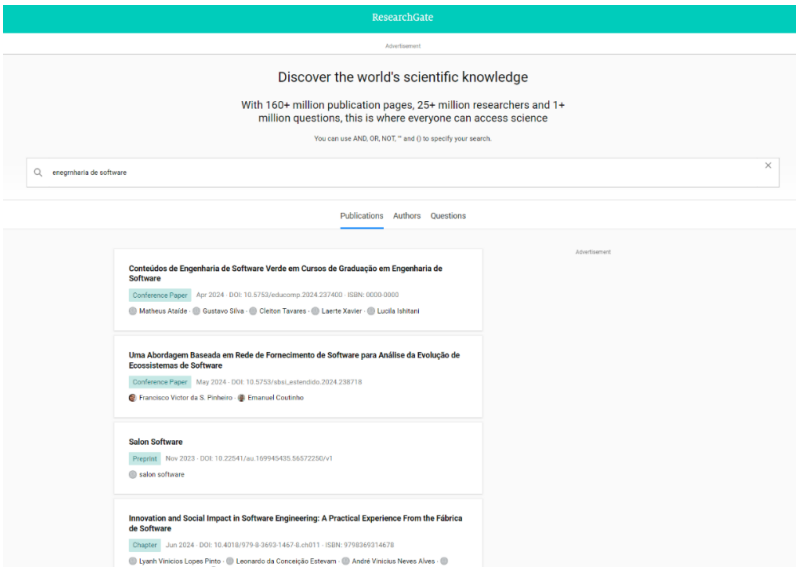

Figura 7. ResearchGate – Tela de pesquisa (Madisch, 2008)

3.3. Academia.edu

O Academia.edu (Figuras 8 e 9) é uma plataforma online voltada para a comunidade acadêmica, que facilita o compartilhamento e acesso a artigos científicos, teses, dissertações e outros tipos de pesquisa.

A plataforma permite que os usuários criem perfis acadêmicos detalhados, onde podem listar suas publicações, projetos de pesquisa e interesses acadêmicos. Além disso, oferece recursos para o *upload* e compartilhamento de artigos completos, proporcionando acesso aberto à literatura acadêmica para a comunidade científica global.

Os membros do Academia.edu podem seguir outros pesquisadores, formar redes acadêmicas e receber atualizações sobre novas pesquisas e publicações relevantes em suas

áreas de interesse. Isso facilita a descoberta de novas colaborações, oportunidades de pesquisa e o estabelecimento de conexões significativas dentro da comunidade acadêmica.

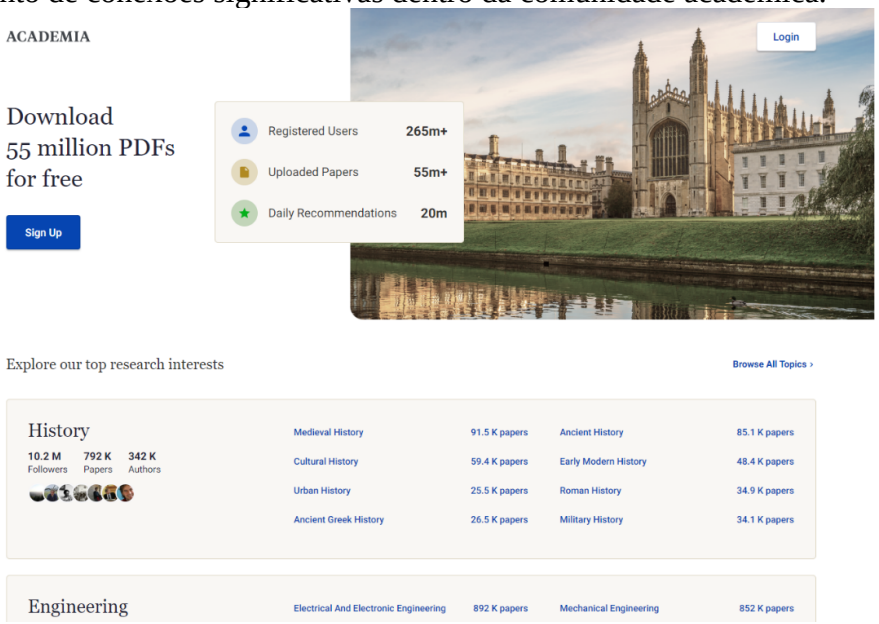

Figura 8. Academia.edu – Tela de inicio (Price, 2008)

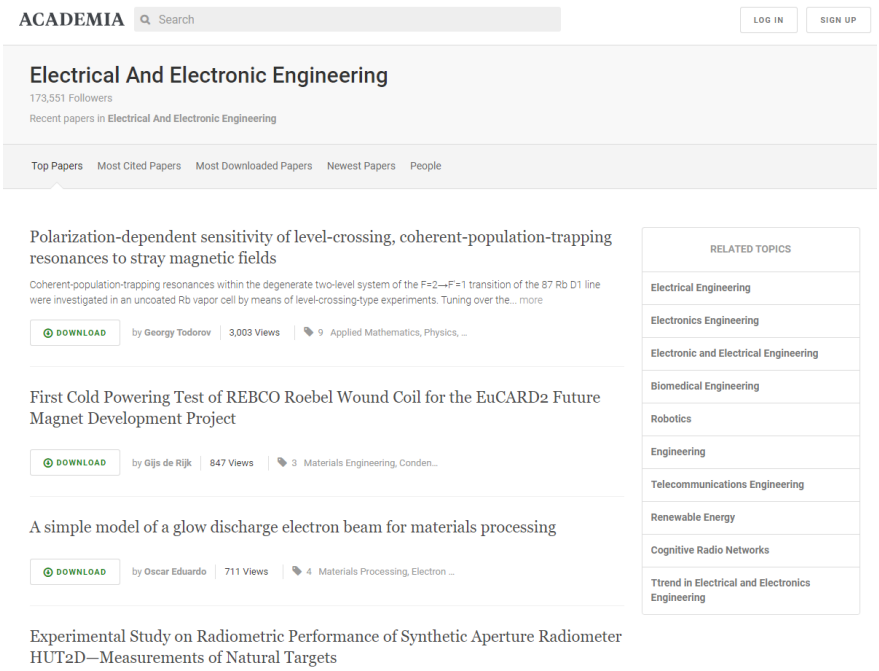

Figura 9. Academia.edu – Tela de pesquisa (Price, 2008)

3.4 Comparativo entre trabalhos correlatos

Com base nos trabalhos correlatos informados, foi possível rastrear as funcionalidades esperadas para o repositório a ser criado, foram utilizados os conceitos gráficos otimizados do Google como referência da interface esperada de usuário e os demais conceitos de funcionalidade e busca dos outros trabalhos sendo Academia.eu e ReasearchGate, uma representação gráfica pode ser vista na Tabela 1.

Tabela 1. Comparação dos trabalhos correlatos

Trabalhos cortelatos	Download de PDF	Estrutura resumida	Busca por termos	Busca por autores	Subida e edição apartir de fonte externa ou codigo	Download gratuito direto da ferramenta
Google Acadêmico	Não	Não	Sim	Sim	Não	Não
ResearchGate	Não	Sim	Sim	Sim	Não	Não
Academia Edu	Sim	Sim	Sim	Sim	Sim	Sim
Repositório de TCC's	Sim	Sim	Sim	Sim	Sim	Sim

O desenvolvimento do Repositório de TCCs foi influenciado pelos *layouts* e funcionalidades de plataformas como Google Acadêmico, ResearchGate e Academia.edu, adaptando suas características para oferecer uma experiência mais completa.

O Google Acadêmico não permite o *download* direto de PDFs, nem possui ferramentas para edição e upload de documentos. O ResearchGate, embora conte com busca avançada por termos e autores, também carece de funcionalidades para download gratuito e edição de documentos externos.

O Academia.edu por sua vez se destaca por integrar todas essas funcionalidades, permitindo busca eficiente, *upload* de arquivos e *download* gratuito de PDFs. Com base nesse modelo, o Repositório de TCCs foi desenvolvido para oferecer uma plataforma robusta e acessível, possibilitando a consulta, o gerenciamento e o compartilhamento de trabalhos acadêmicos de forma intuitiva e eficiente.

4. Metodologia

O desenvolvimento deste trabalho seguiu uma abordagem incremental, permitindo a implementação progressiva do sistema. Inicialmente, um protótipo da interface foi criado no Canva para validar o *design* (Canva, 2024). Em seguida, o Laravel foi utilizado para estruturar a aplicação e integrá-la ao banco de dados MySQL, escolhido por sua confiabilidade e suporte (Oracle, 2024).

No *front-end* foram utilizados Bootstrap (TWBS, 2020) e CSS (W3C, 1996), permitindo a criação de uma interface responsiva que se adapta a diferentes tamanhos de tela e dispositivos

No *back-end*, o Laravel foi adotado por oferecer recursos como autenticação, gerenciamento de rotas e otimização do banco de dados (Otwell, 2011). Para garantir adaptação e usabilidade, a interface foi desenvolvida utilizando Bootstrap, com elementos como:

- container: Centraliza o conteúdo da página.
- row e col-*: Estruturam os elementos em um *layout* responsivo.
- d-flex: Facilita o alinhamento flexível dos componentes.
- text-center: Centraliza textos dentro de seções da interface.

O levantamento de requisitos foi realizado com base em informações coletadas a partir dos trabalhos correlatos Google Academico item 3.1, ResearchGate item 3.2 e Academia.edu item 3.3. Com esses dados, foram criados diagramas de casos de uso e Diagrama de modelagem de banco de dados para documentar o sistema.

Inicialmente, considerou-se o uso de um banco não relacional, mas devido à sua complexidade, optou-se pelo MySQL, que proporciona maior eficiência na organização e recuperação de dados. Durante o desenvolvimento, foram aplicadas boas práticas no Laravel, incluindo:

- Gerenciamento de rotas: Definição precisa de URLs para otimização da funcionalidade.
- Organização do código: Aplicação correta do MVC, garantindo modularidade e manutenção simplificada.
- Eficiência no banco de dados: Utilização do Eloquent ORM para facilitar consultas.
- Segurança: Validação de requisições para evitar SQL Injection e XSS.

- Adaptação visual com Bootstrap: Uso de *container* para centralização, row e col- para responsividade, d-flex para distribuição flexível dos elementos e text-center para alinhamento do conteúdo gráfico*.
- Além dessas otimizações, foi implementada a conversão de arquivos PDF com a biblioteca PdfToText, permitindo a extração de palavras-chave e indexação automática dos documentos para facilitar a busca e recuperação no sistema (Vigh, 2024).

5. Desenvolvimento do trabalho

O primeiro passo para o desenvolvimento do trabalho foi o levantamento de requisitos funcionais e não funcionais, foram analisadas as funcionalidades que serão desenvolvidas ao longo do trabalho, cada uma delas foi detalhadamente revisada para garantir um entendimento claro sobre o que deveria ser desenvolvido, isto a partir de uma tela prototipada (Figura 10) utilizando a ferramenta Canva (Canva, 2024).

Figura 10. Protótipo

Além disso, a aplicação *web* foi idealizada para ser responsiva, a responsividade do sistema foi garantida utilizando o *framework* Bootstrap. O *layout* foi estruturado com *container* para centralização e d-flex text-center para alinhamento.

O formulário de pesquisa utiliza row e col-12 col-sm-10 col-md-8 col-lg-6 para ajustar dinamicamente os elementos conforme o tamanho da tela. Isso permite que o sistema se adapte automaticamente a diferentes dispositivos, garantindo acessibilidade e experiência otimizada para o usuário.

Podendo ser utilizado por diversos dispositivos, atendendo assim aos requisitos não funcionais. Outro requisito não funcional foi a facilidade de utilização, visto que no projeto existem diferentes tipos de dispositivos podem acessar a página e é importante que o software seja fácil de usar e acessível a qualquer público.

Após o término da análise, foi desenvolvido um diagrama de caso de uso da Figura 11, para melhor visualização dos requisitos. Neste diagrama estão especificados os requisitos funcionais da aplicação e quais são os atores do sistema (usuário e usuário administrador). Esse modelo ajudou na organização e entendimento de como o sistema seria estruturado em termos de interações dos usuários com a aplicação.

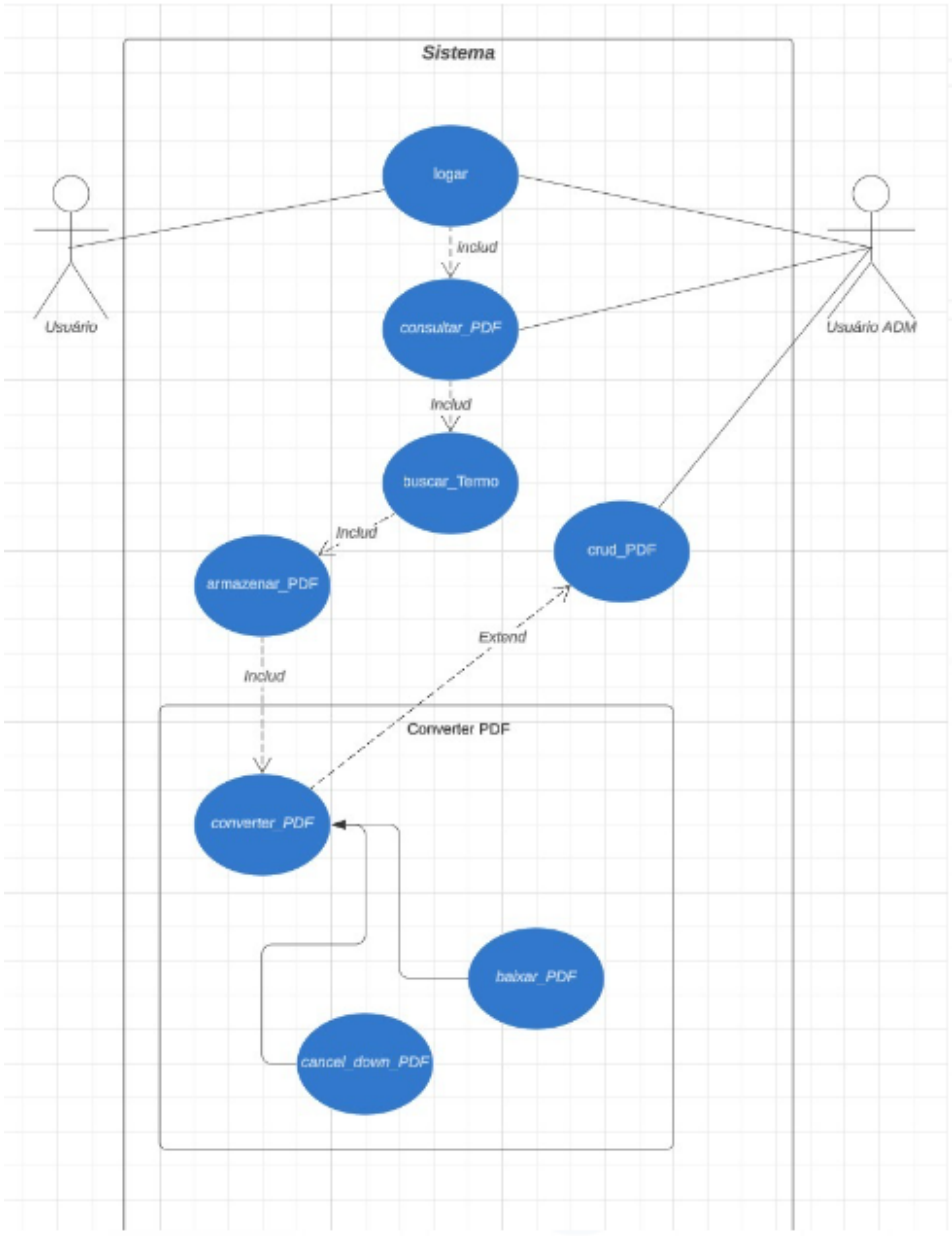

Figura 11. Diagrama de caso de uso

No diagrama da Figura 11, fica evidente a funcionalidade disponível para os diferentes usuários do sistema, definidos pelos seus papéis. Neste trabalho, os dois principais atores são o administrador e o usuário comum, cada qual com acessos e permissões distintas, de acordo com as necessidades do sistema.

No diagrama apresentado na Figura 12, fica evidente a estrutura de modelagem do banco de dados, composta por duas tabelas principais: trabalhos e usuários. A tabela trabalhos armazena informações relacionadas aos Trabalhos de Conclusão de Curso (TCC), já a tabela “usuários” armazena informações sobre os usuários do sistema. Este trabalho apresentou um protótipo navegável, permitindo validar conceitos fundamentais para um futuro sistema completo.

O protótipo desenvolvido demonstra funcionalidades essenciais, mas requer aprimoramentos antes da implementação final. Fases futuras do projeto incluem testes mais robustos, validação com usuários e aprimoramento da arquitetura do sistema. Conforme os princípios de Nielsen (1993), a avaliação da usabilidade deve ser um processo contínuo, visando aprimorar a interação entre o usuário e o sistema por meio de *feedback* iterativo.

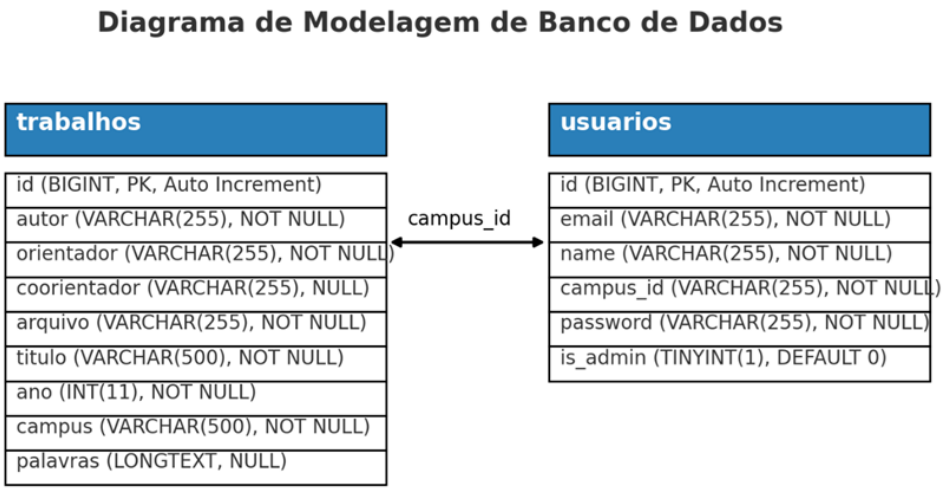

Figura 12. Diagrama de modelagem de banco de dados

O administrador possui acesso a funcionalidades abrangentes, incluindo o gerenciamento de usuários, trabalhos cadastrados e termos (representados por “palavra” no diagrama). Ele pode realizar operações como cadastro de registros e geração de relatórios das pesquisas, por termos, nomes e demais dados representados que são essenciais para a gestão do sistema e para busca de arquivos.

No entanto, foi tomada a decisão estratégica de restringir as opções de edição e remoção de dados exclusivamente ao usuário com acesso direto ao banco de dados. Essa escolha visa garantir maior segurança, evitando alterações indevidas e protegendo a integridade das informações armazenadas.

Por outro lado, os usuários comuns têm funcionalidades mais restritas, adequadas às suas necessidades. Eles podem realizar ações como buscas de trabalhos, *download* dos arquivos e consultas por termos, sendo os dados buscados do banco de dados de trabalhos disponíveis. Esse escopo limitado busca manter a segurança do sistema e assegurar que apenas as operações apropriadas sejam executadas por esses atores.

A partir dessa análise de requisitos, foi possível identificar as necessidades principais do sistema, como o cadastro de usuários e trabalhos, além do gerenciamento eficiente de categorias e permissões. Com base nessas funcionalidades, foi elaborado um modelo lógico de banco de dados que inclui as tabelas especificadas no diagrama de modelagem para armazenar informações de usuários, trabalhos e orientador por exemplo. Esse modelo assegura a organização e a integridade dos dados, atendendo aos requisitos estabelecidos para o sistema.

O diagrama resultante reflete a divisão clara de responsabilidades entre os diferentes atores do sistema, além de evidenciar a estrutura lógica para armazenar e gerenciar os dados. Dessa forma, as funcionalidades foram desenvolvidas para atender às principais demandas identificadas, garantindo uma solução eficiente e segura para o sistema como um todo.

5.1 Implementação do Primeiro Incremento

A partir da tela de protótipo da Figura 13, foi desenvolvida a tela inicial do projeto, correspondente à tela de pesquisa. Essa tela foi criada inicialmente sem qualquer integração com o back-end ou adição de lógicas de interação com o usuário. A implementação foi feita utilizando apenas HTML básico, sem a inclusão de scripts, CSS ou *frameworks* como o Bootstrap, inspirando-se no design minimalista da página de busca do Google.

No primeiro esboço da tela de pesquisa da Figura 14, os campos básicos para entrada de termos, autores ou títulos e botões de pesquisa simples. Essa estrutura inicial serviu de base para o desenvolvimento de futuras funcionalidades, neste esboço é somente uma página vazia.

Figura 13. Primeiro esboço da tela de pesquisa

A página inicial do projeto a Figura 14, teve depois sua estrutura representada no código o arquivo index.php. Este arquivo inicializa o sistema, carregando o autoload (autoload.php) e foram criadas rotas para registrar as rotas do projeto no arquivo routes.php mostrado na figura 15. A rota principal (/) chama o método index da classe TrabalhoController, localizada em app/controllers/TrabalhoController.php.

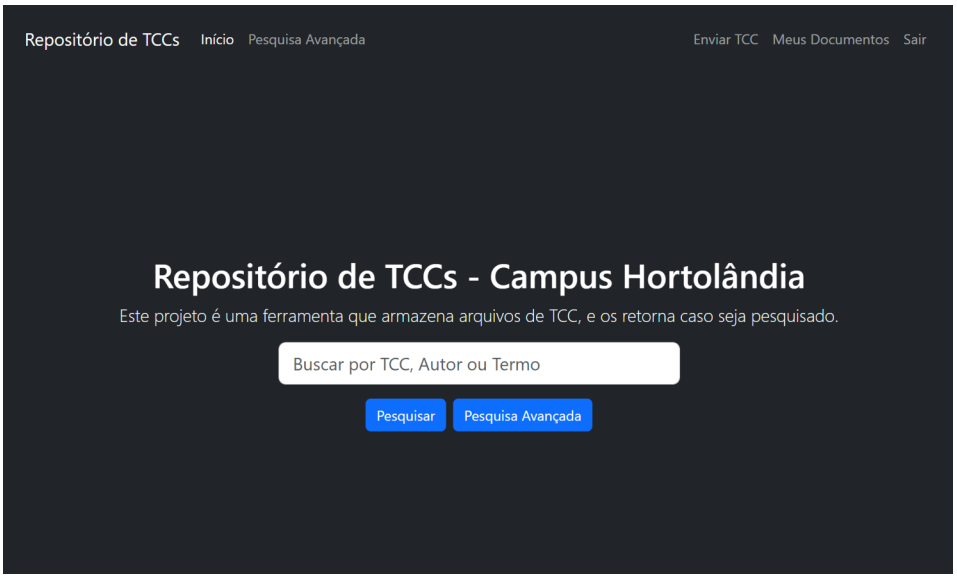

Figura 14. Tela inicial de pesquisa com estilização

Para garantir a responsividade do sistema, foi utilizado o *framework* Bootstrap, que permite a criação de interfaces ajustáveis a diferentes dispositivos. O *layout* do portal foi estruturado com *container* para centralização e d-flex para distribuição flexível dos elementos. O campo de busca foi projetado utilizando row e col-12, garantindo uma disposição adequada em diferentes tamanhos de tela.

Para garantir uma interface visual estruturada e responsiva, foi utilizada a linguagem CSS (Cascading Style Sheets), que possibilita a personalização dos elementos da interface do

sistema. A estilização foi aplicada tanto por meio de arquivos CSS externos quanto pelo uso de classes predefinidas do *framework* Bootstrap (W3C, 2020).

O CSS foi fundamental para definir cores, tipografia, espaçamentos e alinhamentos, proporcionando uma experiência de navegação mais intuitiva e acessível. Além disso, foram aplicados media queries, garantindo a adaptação do *layout* a diferentes tamanhos de tela, tornando o sistema compatível com dispositivos móveis, tablets e desktops (W3C, 2020).

```
<link rel="stylesheet" href="styles.css">
```

Figura 15. O trecho da inclusão do CSS na estrutura do projeto

O trecho da figura 15 centraliza as regras de estilização do sistema, permitindo a manutenção eficiente do código e a aplicação de padrões visuais consistentes.

5.2 Implementação do Segundo Incremento

A escolha do *framework* Laravel foi motivada pela sua estrutura baseada no padrão MVC. O Controller gerencia as requisições do usuário, encaminhando as interações para o *Model*, que processa os dados no banco MySQL. Os resultados são renderizados na *View*, utilizando Blade Templates.

As principais preocupações durante a implementação envolveram:

- Gerenciamento de rotas: Definição clara das URL endpoints;
- Organização modular do código: Separando lógica, interface e controle;
- Segurança: Implementação de proteção contra SQL Injection e XSS.

No segundo incremento, foi desenvolvida a página de Pesquisa Avançada da Figura 16, cujo comportamento é mapeado pela rota /search e pelo método advanced_search da figura 17 no TrabalhoController. A funcionalidade permite a busca refinada utilizando filtros adicionais, como autor, orientador, coorientador e palavras-chave.

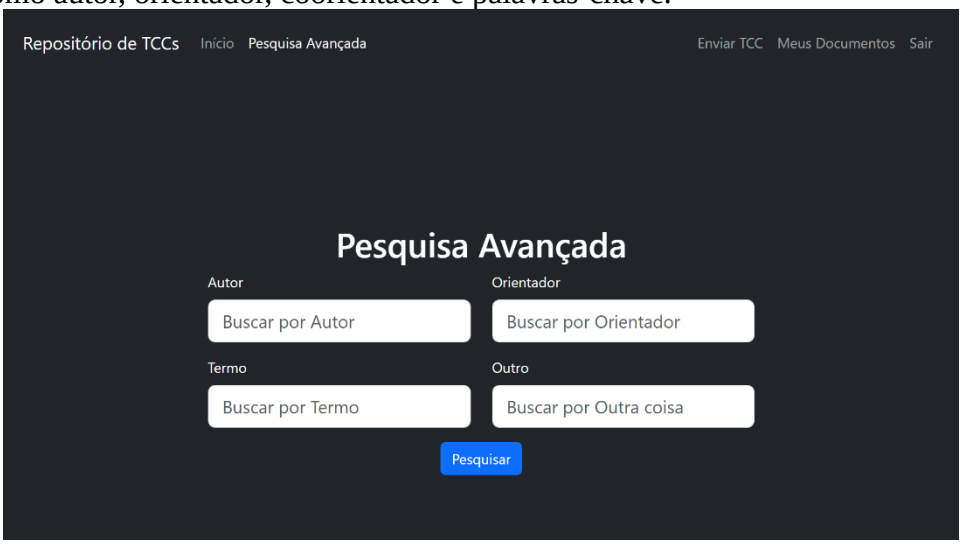

Figura 16. Pesquisa avançada

O arquivo search.php, localizado na pasta views/pages, é o responsável por renderizar os campos do formulário de pesquisa avançada, enquanto a lógica de manipulação dos filtros é gerenciada pelo *back-end* no TrabalhoController como mostrado na Figura 17.

O código apresentado na Figura 17 é referente a função advanced_search(), pertencente ao TrabalhoController, retorna a *view* "pages.advanced_search", permitindo que os usuários realizem buscas avançadas por título, autor, orientador e palavras-chave.

```php
    $result = Trabalho::fetchAllWhere($query, $params);
    return view("pages.results", ['results' => $result]);
}

public static function advanced_search() {
    return view("pages.advanced_search");
}

public static function view_file($file) {
    $root_path = __DIR__ . DIRECTORY_SEPARATOR. ".." . DIRECTORY_SEPARATOR;
    $filename = $root_path ."storage". DIRECTORY_SEPARATOR . "pdf" . DIRECTORY_SEPARATOR . $file;

    if(file_exists($filename)) {
        header('Content-Description: File Transfer');
        header('Content-Type: application/octet-stream');
        header("Cache-Control: no-cache, must-revalidate");
        header("Expires: 0");
        header('Content-Disposition: attachment; filename="'.basename($filename).'"');
        header('Content-Length: ' . filesize($filename));
        header('Pragma: public');

        flush();
        readfile($filename);
        die();
    }
}
```

Figura 17. Pesquisa avançada

Já a função view_file($file) gerencia o *download* de TCCs armazenados. Ela busca o arquivo na pasta "storage/pdf/", configura os headers HTTP e usa readfile($filename) para transferir o conteúdo ao usuário, finalizando o processo com flush(); e die(); para evitar erros.

O TrabalhoController atua como intermediário entre a interface e o banco de dados, garantindo busca eficiente e acesso seguro aos arquivos. A *view* de pesquisa, possivelmente estruturada com Bootstrap, assegura responsividade. Essa abordagem facilita futuras melhorias, como integração com sistemas de autenticação e aprimoramento da interface.

5.3 Implementação do Terceiro Incremento

O terceiro incremento incluiu a criação da página de login da Figura 18 para controle de acesso ao sistema. A página é mapeada pela rota /login, com os métodos login e post_login do UserController, localizado em app/controllers/UserController.php.

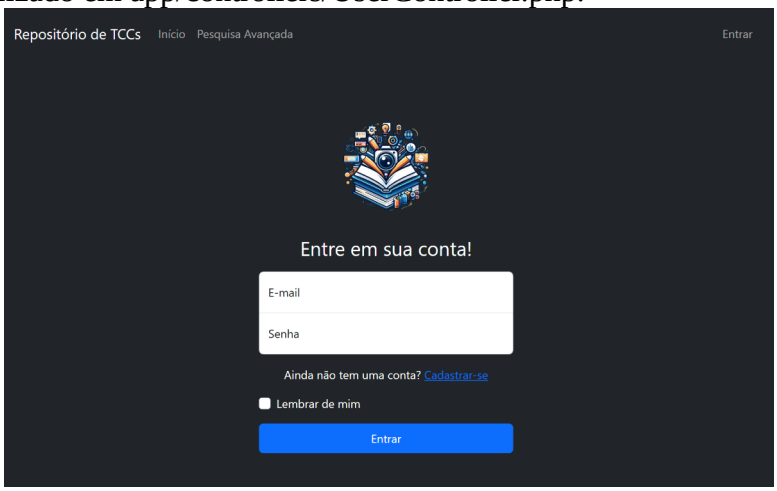

Figura 18. Página de Login

A criação da tela de cadastro de usuário da Figura 19, permitindo que novos usuários possam se registrar no sistema. Essa funcionalidade é essencial para garantir que diferentes

perfis de usuários tenham acesso ao sistema com as permissões apropriadas. A página é mapeada pela rota /register e utiliza os métodos register e post_register do UserController, localizado no arquivo app/controllers/UserController.php.

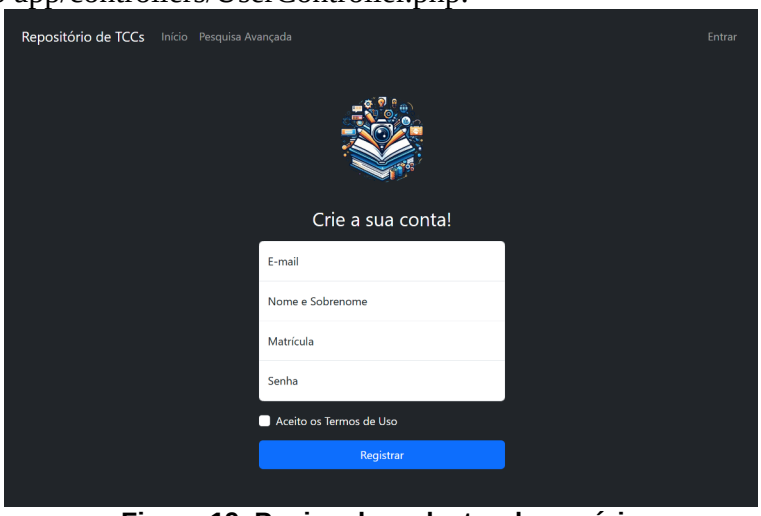

Figura 19. Pagina de cadastro de usuário

A Figura 20 ilustra a implementação das rotas no padrão MVC (Model-View-Controller), que direcionam as requisições do usuário para os controladores adequados, durante a execução das telas de busca anteriormente vistas na figura 16. No código, cada rota é definida por Route::add(), vinculando URLs a métodos no TrabalhoController. A rota / chama index(), carregando a página inicial, enquanto /search direciona para advanced_search(), que processa consultas refinadas. Outras rotas, como view_file() e upload(), lidam com visualização e envio de arquivos. Embora a camada Model (banco de dados) e View (interface gráfica) não estejam na imagem, elas complementam o MVC, garantindo separação de responsabilidades. Essa organização modular facilita a manutenção e escalabilidade do sistema.

```
3   namespace Tcc\App;
4   include(__DIR__."/router.php");
5
6   use \Tcc\App\Bases\BaseModel;
7   use \Tcc\App\Route;
8   use \Tcc\Controllers\TrabalhoController;
9
10  BaseModel::connectDb('mysql:host=127.0.0.1;dbname=repositorio_tcc', 'root', '');
11
12  Route::add('/', function() {
13    TrabalhoController::index();
14  }, 'get');
15
16  Route::add('/results', function() {
17    TrabalhoController::results();
18  }, 'get');
19
20  Route::add('/search', function() {
21    TrabalhoController::advanced_search();
22  }, 'get');
```

Figura 20. Configuração das rotas e métodos do controlador no MVC

Na página de cadastro da Figura 21, desenvolvida em register.php, o *front-end* válida os campos obrigatórios, como nome, e-mail e senha. Após a validação inicial no navegador, os dados são enviados ao *back-end*, que utiliza o modelo Usuario.php para armazenar as

informações no banco de dados. Este modelo segue os princípios do padrão MVC, assegurando a separação das responsabilidades e a integridade das informações.

```php
$busca_usuario = Usuario::fetchAllWhere("UPPER(email) = UPPER(?)", [$data['email']['value'] ?? '']
if (sizeof($busca_usuario) > 0) {
    $is_valid = false;
    $data['email']['valid'] = false;
    $data['email']['valid_message'] = "Email já está em uso.";
}

if ($is_valid) {
    $usuario = new Usuario([
        "email"     => $data["email"]['value'],
        "name"      => $data["name"]['value'],
        "campus_id" => $data["campus_id"]['value'],
        "password"  => password_hash($data["password"]['value'], PASSWORD_BCRYPT),
        "is_admin"  => 0,
    ]);
    $usuario->save();

    header("Location: /login");
    exit();
} else {
    return view("pages.register", ["data" => $data]);
}
```

Figura 21. Cadastro de usuário na página UserController.php

Os *Controllers* no padrão MVC atuam como intermediários entre as requisições do usuário e a lógica da aplicação. No sistema desenvolvido, o método results() do TrabalhoController gerencia a pesquisa de TCCs, processando os filtros de busca e retornando os resultados para a *View* correspondente. Os valores da figura 22 são enviados pelo usuário ao realizar a pesquisa, o argumento “??” evita erros caso o parâmetro não seja passado.

```php
public static function results() {
    $geral      = $_GET['q'] ?? '';
    $autor      = $_GET['a'] ?? '';
    $orientador = $_GET['o'] ?? '';
    $titulo     = $_GET['t'] ?? '';
    $termo      = $_GET['p'] ?? '';
```

Figura 22. Captura os parâmetros da URL (GET)

A Figura 23 concatena diversos campos para buscar por ID, autor, orientador, título e outros atributos, o UPPER() torna a busca insensível a letras maiúsculas e minúsculas.

```php
$query = "(UPPER(CONCAT('|', CONCAT_WS('|', id, autor, orientador, coorientador, arquivo, titulo, ano, campus), '|')) like UPPER(?)) or
(UPPER(autor) like UPPER(?)) or
(UPPER(orientador) like UPPER(?)) or
(UPPER(titulo) like UPPER(?))";
```

Figura 23. Monta a consulta SQL para buscar os TCCs no banco de dados

A Figura 24 é referente a consulta SQL que é passada para o *Model* Trabalho, que executa a busca no banco de dados. Os TCCs encontrados são enviados para a página results.php, onde o usuário visualizará os resultados.

```php
    }

    $result = Trabalho::fetchAllWhere($query, $params);
    return view("pages.results", ['results' => $result]);
}
```

Figura 24. Executa a busca no banco de dados usando o *Model* e Retorna a *View* com os resultados da pesquisa

5.4 Implementação do Quarto Incremento

Nesse incremento, foi implementada a funcionalidade de exibição dos resultados de busca da Figura 25. A rota /results, definida em routes.php, chama o método results do TrabalhoController. Esse método busca os dados no banco de dados, utilizando a tabela trabalhos definida no modelo Trabalho.php.

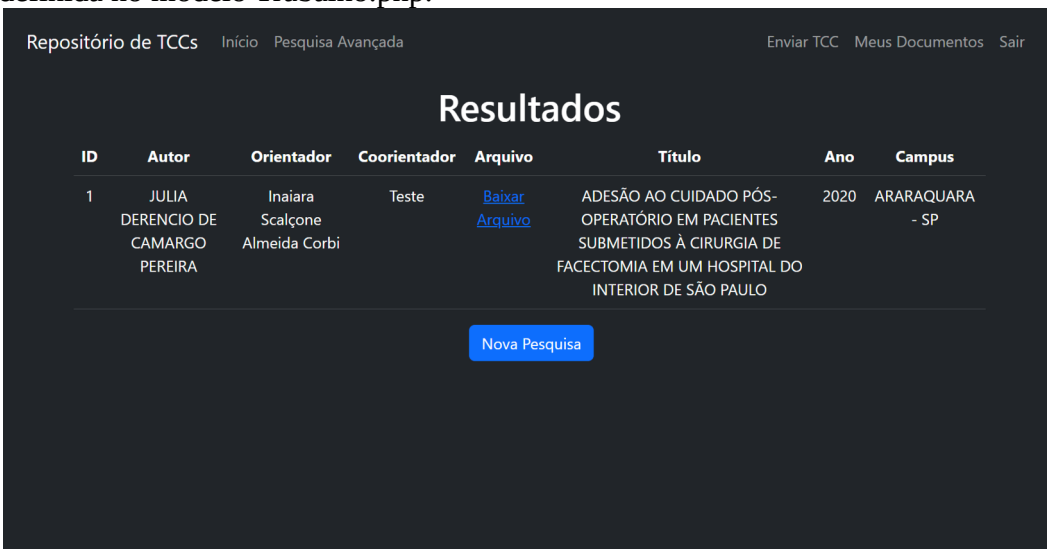

Figura 25. Página de Resultado

A página results.php, localizada em *views/pages*, apresenta os resultados ao usuário. Quando não há registros, uma mensagem de aviso é exibida junto ao botão para iniciar uma nova pesquisa, representado na figura 25.

Além das funcionalidades ainda podemos ver no código das figuras 26 e 27 a representação da implementação da responsividade na interface do repositório, utilizando Bootstrap para garantir uma adaptação eficiente aos diferentes tamanhos de tela. O layout foi estruturado com *container* para centralização e d-flex text-center para alinhamento. O formulário de pesquisa usa row e col-12 col-sm-10 col-md-8 col-lg-6, permitindo ajuste dinâmico conforme a tela do dispositivo. O campo de entrada possui form-control form-control-lg, garantindo dimensionamento adequado, enquanto os botões utilizam btn btn-primary me-1 e btn btn-primary ms-1 para manter espaçamento e estilo uniforme. Esses recursos permitem que a busca seja realizada de forma intuitiva em dispositivos móveis, tablets e desktops, melhorando a acessibilidade em diferentes dispositivos e a experiência do usuário, como parte desse critério de usabilidade e interação humana podemos utilizar de exemplo as cores dos botões e adaptação das telas aos dispositivos utilizados (Preece, 2015).

```php
public static function results() {
    $geral      = $_GET['q'] ?? '';
    $autor      = $_GET['a'] ?? '';
    $orientador = $_GET['o'] ?? '';
    $titulo     = $_GET['t'] ?? '';
    $termo      = $_GET['p'] ?? '';

    $query = "(UPPER(CONCAT('|', CONCAT_WS('|', id, autor, orientador, coorientador, arquivo, titulo, ano, campus), '|')) like UPPER(?)) or
    (UPPER(autor) like UPPER(?)) or
    (UPPER(orientador) like UPPER(?)) or
    (UPPER(titulo) like UPPER(?))";

    $params = [
        '%|%'. mb_convert_encoding($geral, 'ISO-8859-1', 'UTF-8') .'%|%',
        '%|%'. mb_convert_encoding($autor, 'ISO-8859-1', 'UTF-8') .'%|%',
        '%|%'. mb_convert_encoding($orientador, 'ISO-8859-1', 'UTF-8') .'%|%',
        '%|%'. mb_convert_encoding($titulo, 'ISO-8859-1', 'UTF-8') .'%|%'
    ];
```

Figura 26. Função implementa a lógica de busca avançada no banco de dados

Figura 27. Estrutura responsiva do campo de busca

6. Conclusão

Este trabalho apresentou a prototipação funcional de um repositório digital de TCCs, aplicando conceitos de responsividade, arquitetura MVC e modelagem incremental. O protótipo permitiu testes iterativos, validação de funcionalidades e aprimoramento progressivo, prevendo futuras otimizações nos algoritmos de busca e integração com bases de dados externas.

O objetivo foi criar um protótipo navegável para armazenar e consultar TCCs do IFSP – Campus Hortolândia, centralizando documentos e oferecendo busca avançada. O desenvolvimento seguiu um modelo de prototipação, possibilitando a validação de conceitos antes da implementação final. Cada incremento refinou funcionalidades como interface de pesquisa e organização de dados, considerando princípios de Interação Humano-Computador (IHC), além de integrar conhecimentos de Arquitetura de Software e Desenvolvimento *Web*.

Os principais desafios envolveram a aplicação de *frameworks* e configuração de rotas no banco de dados, superados com suporte de documentações e fóruns especializados. A princípio, foi avaliado o uso de um banco não relacional, mas optou-se pelo MySQL devido a melhor integração com as tecnologias utilizadas. O sistema foi estruturado com PHP, Laravel e Bootstrap, garantindo funcionalidade e adaptação responsiva.

A experiência adquirida proporcionou aprofundamento prático em Laravel, banco de dados, integração de *frameworks* e modelagem de sistemas, consolidando conhecimentos teóricos das disciplinas de Banco de Dados, Algoritmos, Programação, IHC e Metodologias Ágeis na construção de um protótipo funcional voltado à gestão de TCCs.

7. Trabalhos Futuros

Para aprimorar o sistema desenvolvido, sugere-se as seguintes melhorias:

- Expansão dos filtros de busca: Implementação de *frameworks* avançados para refinamento dos resultados, possibilitando pesquisas mais precisas e eficientes;
- Integração com repositórios acadêmicos: Conexão com bases de dados externas para permitir a recuperação e cruzamento de informações de diferentes fontes;
- Testes de usabilidade com usuários: Avaliação da experiência de navegação, incluindo adaptações para acessibilidade, como suporte a leitores de tela para usuários com deficiência visual.

Essas melhorias permitirão um refinamento da busca, uma integração mais ampla com bases de dados acadêmicas e um design mais acessível, garantindo que o sistema seja intuitivo e eficiente para diferentes perfis de usuários.

Referências

ALPHABET INC. Google Acadêmico. 2004. Disponível em: https://scholar.google.com.br/. Acessado em: 6 jan. 2024.

APPSLANKA. MVC Architecture. 2023.Disponível em: https://appslanka.lk/our-blog/mvc-architecture. Acesso em: dez. 2024.

CANVA. Canva: a melhor ferramenta de design gráfico online. 2024. Disponível em: https://www.canva.com. Acessado em: 3 dez. 2024.

DIAS, Ricardo. O modelo incremental. 2019. Disponível em: https://medium.com/contexto-delimitado/o-modelo-incremental-b41fc06cac04. Acesso em: dez. 2024.

GLINZ, M. On non-functional requirements. In: IEEE INTERNATIONAL REQUIREMENTS ENGINEERING CONFERENCE, 15., 2007, New Delhi. Anais... New Delhi: IEEE, 2007. p. 21-26. DOI: 10.1109/RE.2007.45.

INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO – IFSP. Normas e procedimentos para elaboração do Trabalho de Conclusão do Curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas. 2024. Disponível em: https://hto.ifsp.edu.br/cloud/s/PbkRTRpPHGrM44t. Acessado em: 10 jun. 2024.

MADISCH, I.; HOFMAYER, S. ResearchGate. 2008. Disponível em: https://www.researchgate.net/. Acessado em: 6 jan. 2024.

NIELSEN, Jakob. Usability engineering. Boston: Morgan Kaufmann, 1993.

ORACLE CORPORATION. MySQL. 2024. Disponível em: https://www.mysql.com/. Acessado em: 6 jan. 2024.

OTWELL, Taylor. Laravel: The PHP framework for web artisans. 2011. Disponível em: https://laravel.com/. Acessado em: 6 jan. 2024.

PREECE, J.; ROGERS, Y.; SHARP, H. Interaction Design: Beyond Human-Computer Interaction. 4. ed. New York: Wiley, 2015.

PRESSMAN, Roger S. Engenharia de Software: Uma Abordagem Profissional. 8. ed. Porto Alegre: AMGH, 2016.

PRICE, R. Academia.edu. 2004. Disponível em: https://www.academia.edu/. Acessado em: 6 jan. 2024.

SOMMERVILLE, Ian. Software Engineering. 10. ed. Boston: Pearson, 2020.

TWBS. Bootstrap: The most popular HTML, CSS, and JS framework for developing responsive, mobile-first projects on the web. 2020. Disponível em: https://getbootstrap.com. Acessado em: 3 dez. 2024.

VIGH, C. PdfToText: biblioteca para conversão de PDF para texto. Disponível em: https://github.com/christian-vigh-phpclasses/PdfToText/tree/master. Acessado em: 3 dez. 2024.

W3C. Cascading Style Sheets (CSS). 1996. Disponível em: https://www.w3.org/Style/CSS/. Acessado em: 3 dez. 2024.

Documento Digitalizado Público

Anexo I - Versão final do TCC

Assunto: Anexo I - Versão final do TCC
Assinado por: Daniela Marques
Tipo do Documento: Projeto
Situação: Finalizado
Nível de Acesso: Público
Tipo do Conferência: Documento Digital

Documento assinado eletronicamente por:

- **Daniela Marques**, **PROFESSOR ENS BASICO TECN TECNOLOGICO**, em 18/03/2025 10:22:55.

Este documento foi armazenado no SUAP em 18/03/2025. Para comprovar sua integridade, faça a leitura do QRCode ao lado ou acesse https://suap.ifsp.edu.br/verificar-documento-externo/ e forneça os dados abaixo:

Código Verificador: 1969956
Código de Autenticação: 7682071741