

Faculdade

XPe



RELATÓRIO

PROJETO
APLICADO

PÓS-GRADUAÇÃO

XP Educação
Relatório do Projeto Aplicado

Software para Gestão de Acervo de Biblioteca Escolar

Rafael Dainese Ravelli

Orientador(a): Marcos Prochnow

Segunda-feira, 25 de Setembro de 2023



RAFAEL DAINESE RAVELLI

XP EDUCAÇÃO

RELATÓRIO DO PROJETO APLICADO

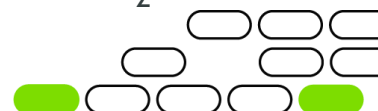
SOFTWARE PARA GESTÃO DE ACERVO DE BIBLIOTECA ESCOLAR

Relatório de Projeto Aplicado
desenvolvido para fins de conclusão do
curso Pós-graduação em Desenvolvimento
Full Stack.

Orientador (a): Marcos Prochnow

São Paulo, SP

25 de Setembro de 2023



Sumário

1. CANVAS do Projeto Aplicado	4
1.1 Desafio	4
1.1.1 Análise de Contexto	4
1.1.2 Personas	5
1.2 Solução	9
1.2.1 Objetivo SMART	9
1.2.2 Premissas e Restrições	10
1.2.3 Backlog de Produto	11
2. Área de Experimentação	13
2.1 Sprint 1 - BackEnd	13
2.1.1 Solução	13
• Evidência do planejamento:	13
• Evidência da execução de cada requisito:	13
• Evidência dos resultados:	13
2.1.2 Lições Aprendidas	20
2.2 Sprint 2 - FrontEnd	21
2.2.1 Solução	21
• Evidência do planejamento:	21
• Evidência da execução de cada requisito:	21
• Evidência dos resultados:	21
2.2.2 Lições Aprendidas	26
2.3 Sprint 3 - FrontEnd + Deploy	26
2.3.1 Solução	26
• Evidência do planejamento:	26
• Evidência da execução de cada requisito:	26
• Evidência dos resultados	29
2.3.2 Lições Aprendidas	29
3. Considerações Finais	29
3.1 Resultados	29
3.2 Contribuições	31
3.3 Próximos passos	31
4. Referências Bibliográficas	31



1. CANVAS do Projeto Aplicado

1.1 Desafio

1.1.1 Análise de Contexto

A internet passou a ser veículo motor da economia global nas últimas décadas, com crescimento exponencial em importância. O profissional dedicado à infraestrutura das redes, desenvolvimento de softwares e trato com clientes ganha destaque e passa a ser peça fundamental para a manutenção de sistemas de informação. Novas tecnologias surgem corriqueiramente e tornam as existentes obsoletas rapidamente:

‘O que era através da internet discada começou a ser utilizada a banda larga e conexão através do celular através da criação da rede 3G (e hoje a criação da rede 4G), hoje em dia a internet acabou virando uma necessidade diária tanto para o uso empresarial, quanto para o uso doméstico.’ (linkdesignbrasil.com, 2022)

Nesse contexto de otimização de serviços via tecnologia, surge o escopo do Desafio. A Escola Avanço está localizada no bairro do Ipiranga em São Paulo, capital. Possui cerca de 500 alunos, de Ensino Fundamental e Médio. Possui 10 anos de existência, sendo considerada iniciante em relação às escolas tradicionais. A Diretora Fernanda Cerqueira aposta em tecnologia e inovação para consolidar-se no mercado. Neste sentido propôs a melhoria do sistema de gestão da biblioteca, que ainda não fora informatizada.

Para a concepção do Produto, foram realizadas reuniões com a Diretoria e também com Conselho da Escola. As reuniões contextualizaram o Time de Tecnologia contratado, e valeram-se de técnicas de *brainstorm* e geraram os relatórios de referência para balisa do Projeto. As Figuras 2 e 3 ilustram o processo. A matriz CSD foi considerada como ponto de partida para analisar o problema/desafio. A ferramenta permitiu explorar questões essenciais sobre o projeto, contextualizando atores, cenários e regras de negócio, em comparação com expectativas dos clientes. O método POEMS permite uma visão sistêmica do ambiente ao qual o desafio está situado.

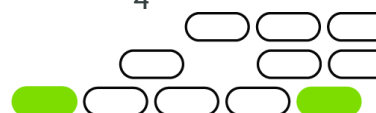
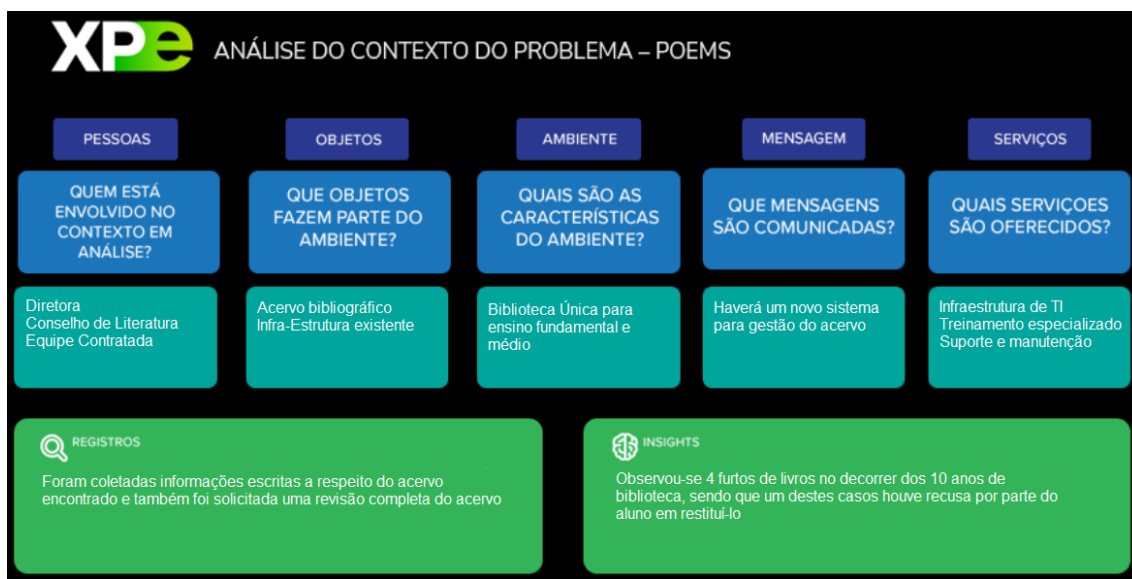


Figura 2 - Matriz CSD (adaptado de IGTI 2023)



Figura 3 - Análise do contexto do problema (POEMS) (adaptado de IGTI 2023)



1.1.2 Personas

O Projeto destinou uso aos funcionários da Biblioteca da Escola Avanço, a saber um estudante universitário de Biblioteconomia e a Bibliotecária oficial. Juntamente aos funcionários, a Conselheira de Literatura foi a Persona ligada responsável ao projeto e sua condução.

A Professora de Literatura e Conselheira Ana Paula Mendez é formada pela Universidade Federal de São Paulo, onde também obteve mestrado em Literatura

Clássica. Atua como professora há 20 anos, sendo responsável pela gestão das equipes de Literatura, Gramática e Redação.

A Equipe Contratada entrevistou a Professora Ana Paula e construiu um Mapa de Empatia. Neste mapa é possível observar critérios mentais do público-alvo, compreendendo desejos e como direcionar o produto. A Figura 4 a seguir apresenta a ferramenta:

Figura 4 - Mapa da Empatia (adaptado de IGTI 2023)



1.1.3 Benefícios e Justificativas

O Projeto justificou-se pela demanda do cliente em otimizar o serviço existente, melhorando qualidade, segurança e redução de custos. Do ponto de vista empresariais, foi uma solução com alta lastreabilidade, que pode ser estendida a novos clientes com facilidade.

Enquanto proposta de valor, observou-se que foram registrados furtos e maus tratos aos livros existentes na Biblioteca até o momento, o que deve ser revertido com a nova implementação. Ampliou-se o escopo do projeto também à segurança e manutenção do patrimônio da Escola, que se mostra atento às políticas de boa convivência da escola.

Por meio do Business Design Blueprint, foi possível observar o dia-a-dia do cliente, afetando a tomada de decisão quanto às particularidades do produto a ser desenvolvido. O método CANVAS, proposta de Valor, entendeu de qual maneira o produto proposto pôde adequar-se às necessidades do cliente, aproximando-os. Seguem nas Figuras 5 e 6 a seguir o Blueprint e CANVAS para a Escola Avanço:

Figura 5 - Blueprint (adaptado de Teixeira, 2017)

Itens	Detalhamento
Objetivos	Otimizar o serviço existente
Atividades	Apresentar os documentos oficiais, encontrar empresas capacitadas
Questões	Haverá dificuldade na manutenção do serviço? Funcionará online?
Barreiras	Dificuldade de relacionamento entre os funcionários da Biblioteca e a Conselheira de Literatura
Ações do cliente	
Funcionalidades	Listar todo o acervo da biblioteca, adicionando 'tags' para localização e filtragem
Interação	Interação com filtros e edição de registros
Mensagem	Logs de resposta para apresentação e atualizações de informações
Onde ocorre	Em ambiente web
Tarefas aparentes	Criação do serviço pela Equipe Contratada
Tarefas escondidas	Alocação de serviço em Cloud
Processos de suporte	Treinamento para uso do Produto
Saída desejável	Funcionamento pleno para a contratante

Figura 6 - CANVAS (O Analista de Modelo de negócios, sem data)



1.1.4 Hipóteses

A partir das reuniões com a Diretoria, Conselho e funcionários da biblioteca, foram levantadas as hipóteses que motivam o desenvolvimento e entrega da solução. Observou-se que o cliente está disposto a cooperar com o projeto, apresentando suas dependências físicas e disponibilizando todo o corpo docente e de funcionários.

Amparado nos balizadores para notas BASICO, critérios consagrados no teste de hipóteses e apresentados na Figura 7 a seguir, a equipe de tecnologia propôs a Matriz de Observações para hipóteses e a Matriz aplicada BASICO, observadas nas Figuras 8 e 9, também em sequência:

Figura 7 - Balizadores BASICO (Material de Apoio IGTI)

Escala	B - Benefícios	A - Abrangência	S - Satisfação	I - Investimentos	C - Cliente	O - Operacionalidade
5	De vital importância	Total (de 70 a 100%)	Muito grande	Pouquíssimo investimento	Nenhum impacto	Muito fácil
4	Significativo	Muito grande (de 40 a 70%)	Grande	Alguns investimentos	Impacto pequeno	Fácil
3	Razoável	Razoável (de 20 a 40%)	Média	Médio investimento	Médio impacto	Média facilidade
2	Poucos benefícios	Pequena (de 5 a 20%)	Pequena	Alto investimento	Impacto grande	Difícil
1	Alguns benefícios	Muito pequena	Quase não é notada	Altíssimo investimento	Impacto muito grande no cliente	Muito difícil

Figura 8 - Matriz de Observações (Autoria Própria)

OBSERVAÇÕES	HIPÓTESES
Os livros ficam desorganizados nas prateleiras	A falta de um sistema de nomenclatura para disposição dos livros está acarretando erros de disposição
Os livros voltam amassados, molhados, às vezes não são devolvidos	Um sistema antigo e obsoleto para arquivo e locação dos livros acarreta o mau uso
Faltam critérios para compra de novos livros	O Conselho de Literatura deveria reunir-se com mais frequência para abordar temas de referência
Os funcionários da livraria podem não conseguir adaptar-se ao novo sistema que será implementado	A biblioteca possui funcionários de idade e que podem demorar a incorporar as novas práticas



Figura 9 - Matriz aplicada BASICO (Autoria Própria)

IDEAIS	B	A	S	I	C	O	TOTAL	PRIORIDADE
Construir dashboard interativo para adição, edição, remoção e consulta aos livros cadastrados (CRUD)	5	5	5	5	5	5	30	1º
Encontrar padrões nos livros existentes para reuni-los em filtros aplicáveis no software	4	3	4	4	3	3	21	2º
Criar funcionalidade Blog para a interações sobre novos temas de estudo em literatura	2	3	2	2	2	5	16	3º
Criar rotas de acesso secundário para alunos acessarem o sistema e fazerem agendamentos da locação de livros	3	3	2	4	2	1	15	4º

1.2 Solução

1.2.1 Objetivo SMART

Baseado no conceito de Objetivo SMART, Specific Mensurable Attainable Relevant and Time Based, foi projetada uma solução que contemple o produto requisitado pela Escola Avanço. Partiu-se da necessidade de otimização do sistema de acervo existente para uma versão mais tecnológica, estável e segura. Foi estabelecido também prazos de manutenção do serviço e treinamentos sazonais para uso da ferramenta. Foram propostos treinamentos semanais no primeiro mês de implementação, seguido de treinamentos quinzenais até o término do primeiro mês de funcionamento. Após esse período, haverá treinamentos mensais.

O sistema contempla o que existe de mais moderno em relação a desenvolvimento de web softwares e deploy. A contratada abre hipóteses de ampliação do produto entregue para novas funcionalidades.



1.2.2 Premissas e Restrições

Buscando sucesso do projeto, foram estabelecidos requisitos balizadores das etapas do desenvolvimento. Considerou-se então custos de soluções, escopo de abrangência do softwares, qualidade do produto, estabilidade e manutenção. Todos os parâmetros foram norteados também pelos prazos estabelecidos, tornando a solução competitiva e economicamente viável.

A Equipe de Tecnologia da contratada optou pelo uso da linguagem de programação Node.js para desenvolvimento do Back-End da aplicação React.js para Front-End. A interface das duas linguagem será dada por API. Seguiu-se o padrão de projetos consagrado MVC (Model-View-Controller) e as métricas de rotas definidas em Controllers, Models, Repositories, Routes e Services. Os comandos CRUD seguiram padrão de API HTTP REST. A alocação de dados será feita em banco de dados relacional MySQL com backups mensais disponibilizados ao cliente. O código-fonte contém em diretório o SQL Query necessário para criação e alimentação (seeds) das tabelas. O código-fonte foi alocado em repositório GIT e será de uso exclusivo da contratada. O deploy da aplicação em ambiente web é cargo da contratada, utilizando servidores AWS.

Ao cliente foi disponibilizada interface gráfica web para uso do sistema. Constou de página de login com senha criptografada e pré-definida. A troca de senha eventual deve ser feita pela contratada. O dashboard de acesso constou de CRUD para visualização, adição, edição e remoção de registros, no caso o acervo existente da biblioteca. Foi implementada também a funcionalidade de filtros para pesquisa de dados.

O serviço foi vendido em regra de passou a ser de domínio do contratante após entrega e validação em POC (Prova de Conceito) de 2 meses (60 dias corridos).

Segue a seguir, na Figura 10, a Matriz de Riscos do Projeto:

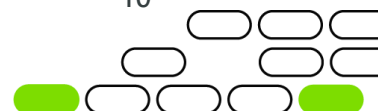


Figura 10 - Matriz de Riscos (Autoria Própria)

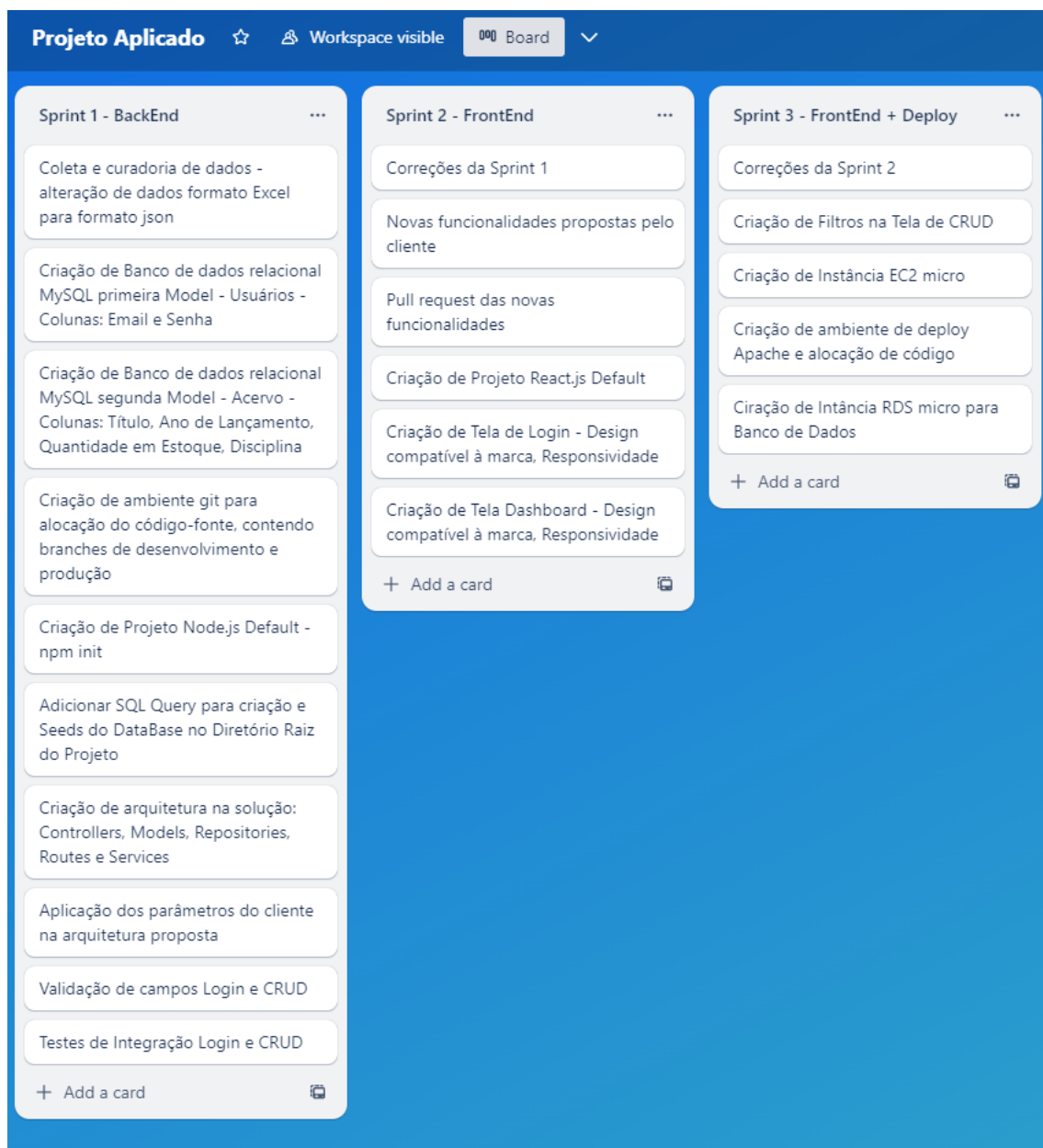
RISCO IDENTIFICADO	IMPACTO POTENCIAL	AÇÕES PREVENTIVAS	AÇÕES CORRETIVAS
Demora na entrega dos dados para análise	Atraso no cronograma de entregas	Realizar análises preliminares, já contabilizando informações-fantasia	Nova visita ao cliente para coleta de informações
Entrega não se mostra funcional ou apresenta problemas	Não atender o objetivo esperado pelo cliente	Particionar o projeto e fazer entregas preliminares dentro da lógica git, melhor definir as branches de produção e teste	Resetar os commits que apresentaram problemas

1.2.3 Backlog de Produto

Esta seção tem o objetivo de apresenta, o backlog de requisitos idealizados para o desenvolvimento da solução para a Escola Avanço. Foi utilizada a plataforma Trello como gerenciadora de tarefas. O planejamento previu a distribuição das tarefas em 3 sprints. A Figura 11 a seguir ilustra o processo:



Figura 11 - Planejamento das Sprints (trello.com)



2. Área de Experimentação

Esta seção tem o objetivo de apresentar as evidências do planejamento dos requisitos selecionados do Backlog de Produto, além de mostrar a maneira como eles foram desenvolvidos e registrar os resultados alcançados.

2.1 Sprint 1 - BackEnd

2.1.1 Solução

- **Evidência do planejamento:**

A primeira Sprint contemplou o escopo inicial do projeto, a partir da definição do produto, coleta e curadoria de dados e Desenvolvimento do BackEnd do Software Produto.

- **Evidência da execução de cada requisito:**

1. Coleta e curadoria de dados - alteração de dados formato Excel para formato json
2. Criação de Banco de dados relacional MySQL primeira Model - Usuários - Colunas: Email e Senha
3. Criação de Banco de dados relacional MySQL segunda Model - Acervo - Colunas: Título, Ano de Lançamento, Quantidade em Estoque, Disciplina
4. Criação de ambiente git para alocação do código-fonte, contendo branches de desenvolvimento e produção
5. Criação de Projeto Node.js Default - npm init
6. Adicionar SQL Query para criação e Seeds do DataBase no Diretório Raiz do Projeto
7. Criação de arquitetura na solução: Controllers, Models, Repositories, Routes e Services
8. Aplicação dos parâmetros do cliente na arquitetura proposta
9. Validação de campos Login e CRUD
10. Testes de Integração Login e CRUDv

- **Evidência dos resultados:**

1. Os dados recebidos foram tratados e estão apresentados no arquivo dados_tratados.json, que está localizado na pasta raiz do código-fonte. A Figura 12 apresenta dados contidos no arquivo. A estrutura do JSON já foi projetada atenta às regras do banco de dados. Os dados foram inseridos um a um no banco de dados. As labels do JSON foram construídas com snake_case.



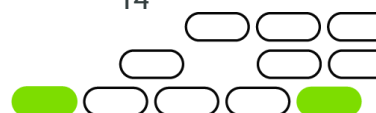
Figura 12 - Dados em formato JSON (IDE VSCode)

```
01. XPE Course Projects > TCC-projeto-aplicado > {} dados.json > {} 3 > disciplina
1  [
2      {
3          "id": 1,
4          "titulo": "Navegando pela História Moderna",
5          "ano_de_lancamento": "2020",
6          "quantidade_em_estoque": 3,
7          "disciplina": "História"
8      },
9      {
10         "id": 2,
11         "titulo": "Biologia para ensino fundamental",
12         "ano_de_lancamento": "2015",
13         "quantidade_em_estoque": 5,
14         "disciplina": "Biologia"
15     },
16     {
17         "id": 3,
18         "titulo": "Matemática de A a Z",
19         "ano_de_lancamento": "2018",
20         "quantidade_em_estoque": 10,
21         "disciplina": "Matemática"
22     },
23 ]
```

2. Segue na Figura 13 a seguir a SQL Query de criação da Model requisitada. Para tanto foi instalado o software MySQL e o client HeidiSQL.

Figura 13 - Model usuarios (HeidiSQL)

```
1 CREATE TABLE `usuarios` (
2   `id` INT(10) NOT NULL AUTO_INCREMENT,
3   `email` VARCHAR(50) NULL DEFAULT NULL COLLATE 'utf8mb4_0900_ai_ci',
4   `senha` VARCHAR(50) NULL DEFAULT NULL COLLATE 'utf8mb4_0900_ai_ci',
5   PRIMARY KEY (`id`) USING BTREE
6 )
7 COLLATE='utf8mb4_0900_ai_ci'
8 ENGINE=InnoDB
9 ;
10
```



3. Segue na Figura 14 a seguir a SQL Query de criação da Model requisitada. Segui-se a métrica do tópico 2.

Figura 14 - Model acervo (HeidSQL)

```

1 CREATE TABLE `acervo` (
2   `id` INT(10) NOT NULL AUTO_INCREMENT,
3   `titulo` VARCHAR(50) NULL DEFAULT NULL COLLATE 'utf8mb4_0900_ai_ci',
4   `ano_de_lancamento` VARCHAR(50) NULL DEFAULT NULL COLLATE 'utf8mb4_0900_ai_ci',
5   `quantidade_em_estoque` INT(10) NULL DEFAULT NULL,
6   `disciplina` VARCHAR(50) NULL DEFAULT NULL COLLATE 'utf8mb4_0900_ai_ci',
7   PRIMARY KEY (`id`) USING BTREE
8 )
9 COLLATE='utf8mb4_0900_ai_ci'
10 ENGINE=InnoDB
11 AUTO_INCREMENT=2
12 ;
13

```

4. Foi utilizado o servidor Github para alocação do código. Aproveitou-se repositório existente, onde foi criada a branch 'development'. Ao invés de 'master' branch, o servidor Github utilizada 'main' branch, que foi mantida.

Figura 15 - Projeto GIT (GitHub)

Projects-Node.js / 01. XPE Course Projects /

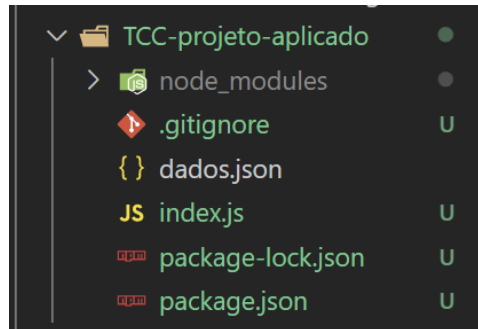
rafaelravelli12 Initial commit to ttc-projeto-aplicado and development branch creation

This branch is 1 commit ahead of `main`.

Name	Last commit message
..	
TCC-projeto-aplicado	Initial commit to ttc-projeto-aplicado and development branch creation
module-1-01-practical-work	Folders Rearrange
module-1-02-challenge	Folders Rearrange
module-2-01-practical-work	Folders Rearrange
module-2-02-challenge	Folders Rearrange
module-3-01-practical-work	Folders Rearrange
module-3-02-challenge	Folders Rearrange
.gitignore	Folders Rearrange
README.md	Folders Rearrange

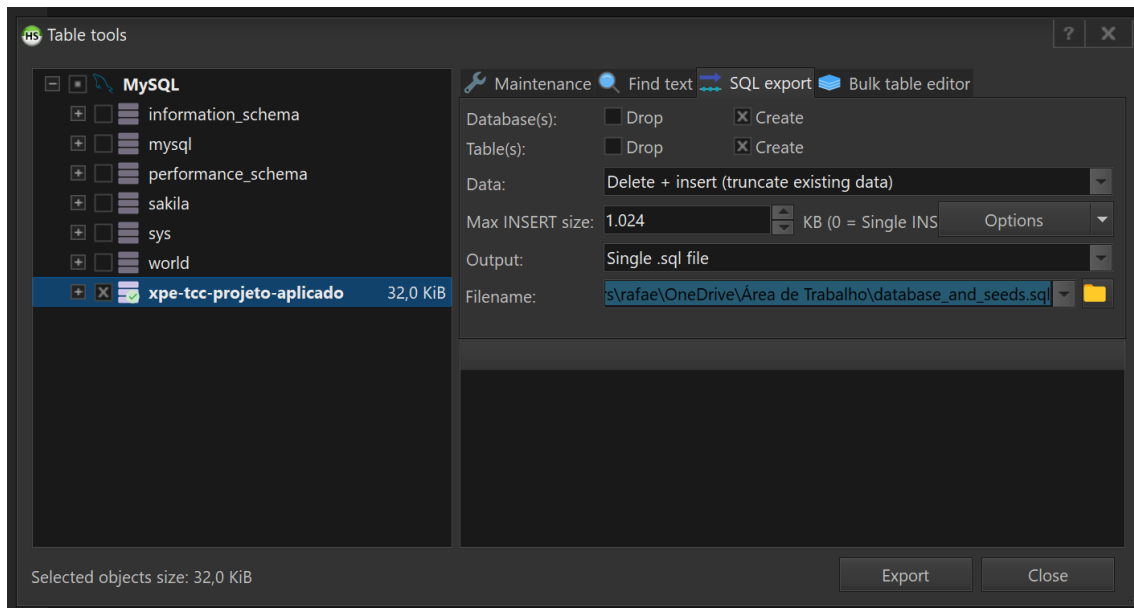
5. A criação do projeto base necessitou da instalação do software Node.js, do site homônimo. O comando de terminal 'npm init', junto à adição das dependências express e nodemon, geraram o projeto padrão mostra na Figura 16 a seguir:

Figura 16 - npm init (Node.js + Vscode)



6. O arquivo database_and_seeds.sql foi criado com o recurso disponibilizado no HeidiSQL e depois armazenada no diretório raiz do projeto. A Figura 17 elucida o processo:

Figura 17 - Criação de Database e Seeds (HeidiSQL)



7. A solução apresentou apenas Routes e Services como arquitetura de solução. Partiu-se de 3 páginas fundamentais: index.js, routes.js e service.js.

Index.js fez conexão com a API REST HTTP a partir da porta 3000 e chama as rotas. A Figura 18 mostra o resultado. routes.js definirá o CRUD (Create, Read, Update e Delete) a ser executado pela Service. Routes.js segue na Figura 19 e service.js na Figura 20. O software Insomnia foi utilizado para testar as rotas e está apontado na Figura 21.

Figura 18 - index.js (VSCode)

```
01. XPE Course Projects > TCC-projeto-aplicado > JS index.js > ...
1  import express from "express";
2  import { router } from "../routes/routes.js";
3
4  const app = express();
5  app.use(express.json());
6
7  app.use("/", router);
8
9  app.listen(3000, () => {
10 |     console.log("API Started");
11 | });
12
```

Figura 19 - routes.js (VSCode)

```
01. XPE Course Projects > TCC-projeto-aplicado > routes > JS routes.js
1  import express from "express";
2
3  import {
4      getBooks,
5      addBook,
6      updateBook,
7      deleteBook
8  } from "../service/service.js";
9
10 export const router = express.Router();
11
12 router.get("/getBooks", getBooks);
13 router.post("/addBook", addBook);
14 router.put("/updateBook", updateBook);
15 router.delete("/deleteBook", deleteBook);
16
```



Figura 20 - service.js (VSCode)

01. XPE Course Projects > TCC-projeto-aplicado > service > JS service.js > ...

```
1  import mysql from "mysql2/promise";
2
3  const pool = mysql.createPool({
4    host: "127.0.0.1", // or "localhost"
5    user: "root",
6    password: "root",
7    database: "xpe-tcc-projeto-aplicado"
8  });
9
10 export const getBooks = async (req, res) => {
11   try {
12     const connection = await pool.getConnection();
13     const [rows] = await connection.execute("SELECT * FROM acervo");
14     connection.release();
15
16     res.status(200).json(rows);
17   } catch (error) {
18     console.error("Error:", error);
19     res.status(500).send("Internal Server Error");
20   }
21 };
```

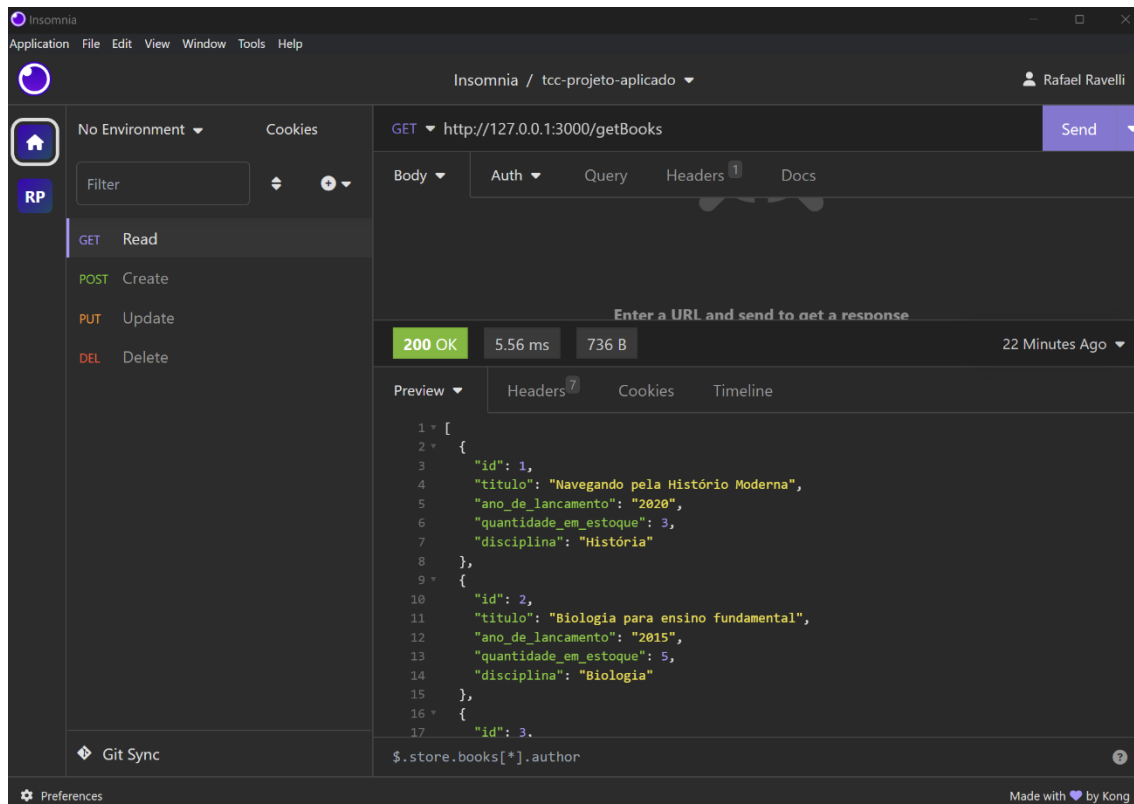
```
23 export const addBook = async (req, res) => {
24   try {
25     const { id, titulo, ano_de_lancamento, quantidade_em_estoque, disciplina } = req.body;
26
27     const connection = await pool.getConnection();
28     await connection.execute(
29       "INSERT INTO acervo (id, titulo, ano_de_lancamento, quantidade_em_estoque, disciplina) VALUES (?, ?, ?, ?, ?)",
30       [id, titulo, ano_de_lancamento, quantidade_em_estoque, disciplina]
31     );
32     connection.release();
33
34     res.status(201).send("Book added successfully");
35   } catch (error) {
36     console.error("Error:", error);
37     res.status(500).send("Internal Server Error");
38   }
39 }
40
```

```
41 export const updateBook = async (req, res) => {
42   try {
43     const { id, titulo, ano_de_lancamento, quantidade_em_estoque, disciplina } = req.body;
44
45     const connection = await pool.getConnection();
46     await connection.execute(
47       "UPDATE acervo SET titulo = ?, ano_de_lancamento = ?, quantidade_em_estoque = ?, disciplina = ? WHERE id = ?",
48       [titulo, ano_de_lancamento, quantidade_em_estoque, disciplina, id]
49     );
50     connection.release();
51
52     res.status(200).send("Book updated successfully");
53   } catch (error) {
54     console.error("Error:", error);
55     res.status(500).send("Internal Server Error");
56   }
57 };
```

```
59 export const deleteBook = async (req, res) => {
60   try {
61     const { id } = req.body;
62
63     const connection = await pool.getConnection();
64     await connection.execute(
65       "DELETE FROM acervo WHERE id = ?",
66       [id]
67     );
68     connection.release();
69
70     res.status(200).send("Book deleted successfully");
71   } catch (error) {
72     console.error("Error:", error);
73     res.status(500).send("Internal Server Error");
74   }
75 };
```



Figura 21 - Rotas testadas via Insomnia (Insominia)



8. O cartão 8 foi descartado devido a ter sido completado nos cartões anteriores
9. Cartão reposicionado para ser feito após frontEnd
10. Cartão reposicionado para ser feito após frontEnd

2.1.2 Lições Aprendidas

A primeira Sprint revelou-se o início dos trabalhos da Equipe de Tecnologia em Desenvolvimento. Foram observados critérios fundamentais para o desenvolvimento de um serviço web, como a criação de um Banco de Dados local para desenvolvimento, a criação de um ambiente GIT para armazenamento e evolução de código-fonte e a criação das primeiras regras de negócio do software em si. Pude exercitar ferramentas até então pouco conhecidas como a compatibilização de um Banco de Dados com uma aplicação Node.js. O código já está disponível em:

<https://github.com/rafaelravelli12/Projects-Node.js>

2.2 Sprint 2 - FrontEnd

2.2.1 Solução

- **Evidência do planejamento:**

A segunda Sprint contemplou eventuais ajustes da primeira Sprint e todo o desenvolvimento Front-End do Software Produto, até sua aprovação pela Diretor da Equipe de Tecnologia

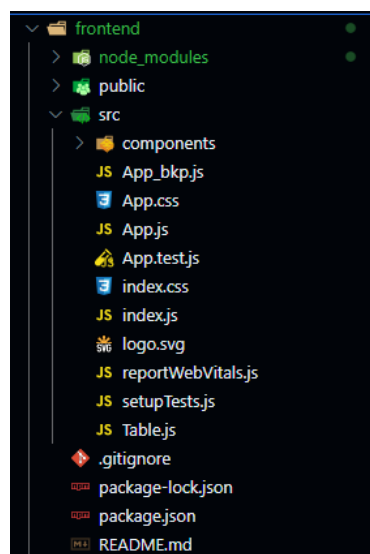
- **Evidência da execução de cada requisito:**

1. Correções da Sprint 1
2. Novas funcionalidades propostas pelo cliente
3. Pull request das novas funcionalidades
4. Criação de Projeto React.js Default
5. Criação de Tela de Login - Design compatível à marca, Responsividade
6. Criação de Tela Dashboard - Design compatível à marca, Responsividade

- **Evidência dos resultados:**

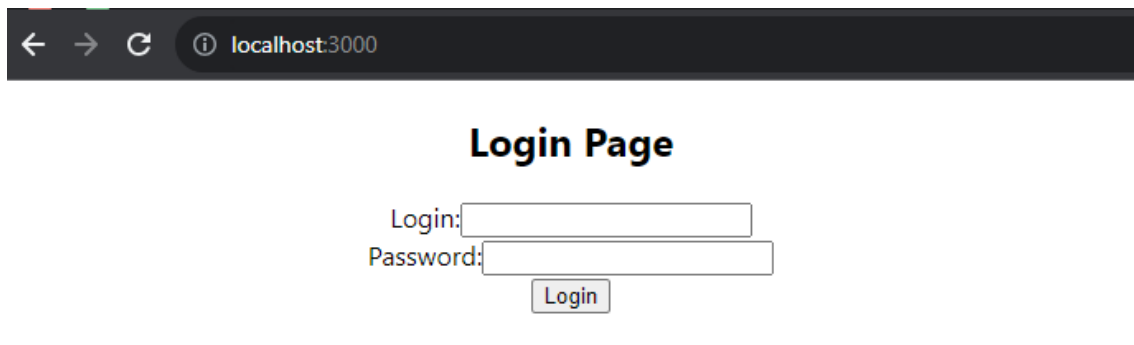
1. A Sprint 1 não incorreu em erros;
2. O cliente não propôs novas funcionalidades;
3. O pull request do commit resultante da Sprint 1 na branch de Desenvolvimento foi realizado ao término da Sprint 1;
4. O projeto padrão de React foi criado a partir das bibliotecas do npm, utilizando o comando: `npx create-react-app` . A Figura 22 mostra o resultado da criação das bibliotecas

Figura 22 - projeto padrão React (VSCode)



5. A primeira versão da tela de login foi apresentada segundo as Figuras 23, 24 e 25 onde vemô-la ainda sem responsividade

Figura 23 - página de Login React



← → ↻ ⓘ localhost:3000

Login Page

Login:

Password:

Login

Figura 24 - página de Login React Apps.js - Parte 1

```

1  import React, { useState } from 'react';
2  import './App.css';
3  import Table from './Table';
4
5  function App() {
6    // Create state variables for login and password
7    const [login, setLogin] = useState('');
8    const [password, setPassword] = useState('');
9
10   // Handle changes in the login and password inputs
11   const handleLoginChange = (event) => {
12     setLogin(event.target.value);
13   };
14
15   const handlePasswordChange = (event) => {
16     setPassword(event.target.value);
17   };
18
19   // Handle login form submission
20   const handleLoginSubmit = (event) => {
21     event.preventDefault();
22     // You can perform authentication logic here
23     console.log('Login:', login);
24     console.log('Password:', password);
25     // Reset the input fields
26     setLogin('');
27     setPassword('');
28   };
29

```

Figura 25 - página de Login React Apps.js - Parte 2

```
30     return (  
31       <div className="App">  
32         <div className="login-container">  
33           <h2>Login Page</h2>  
34           <form onSubmit={handleLoginSubmit}>  
35             <div className="form-group">  
36               <label htmlFor="login">Login:</label>  
37               <input  
38                 type="text"  
39                 id="login"  
40                 name="login"  
41                 value={login}  
42                 onChange={handleLoginChange}  
43                 required  
44               />  
45             </div>  
46             <div className="form-group">  
47               <label htmlFor="password">Password:</label>  
48               <input  
49                 type="password"  
50                 id="password"  
51                 name="password"  
52                 value={password}  
53                 onChange={handlePasswordChange}  
54                 required  
55               />  
56             </div>  
57             <button type="submit">Login</button>  
58           </form>  
59         </div>  
60       </div>  
61     );  
62   }  
63  
64   export default App;  
65 }
```

6. A primeira versão do Dashboard foi apresentada segundo as Figuras 26, 27 e 28, onde vemos o layout inicial, a rota App.js e a Component Table associada

Figura 26 - página de Login React Apps.js - Parte 2

← → ↻ ⓘ localhost:3000

Table Example

ID	Title	Release Year	Stock Quantity	Discipline	Actions
1	Book 1	2020	10	Science	<div>EditDelete</div>
2	Book 2	2019	15	History	<div>EditDelete</div>
3	Book 3	2022	5	Math	<div>EditDelete</div>

Adicionar Livro

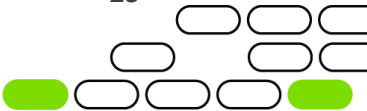


Figura 27 - Apps.js para Dashboard

```
1 // App.js
2
3 import React from 'react';
4 import logo from './logo.svg';
5 import './App.css';
6 import Table from './Table';
7
8 function App() {
9   return (
10     <div className="App">
11       <Table /> {/* Render the Table component here */}
12     </div>
13   );
14 }
15
16 export default App;
17
```



Figura 28 - Trecho da Component Table para Dashboard

```

49     return (
50       <div>
51         <h2>Table Example</h2>
52         <table style={tableStyle}>
53           <thead>
54             <tr>
55               <th style={thStyle}>ID</th>
56               <th style={thStyle}>Title</th>
57               <th style={thStyle}>Release Year</th>
58               <th style={thStyle}>Stock Quantity</th>
59               <th style={thStyle}>Discipline</th>
60               <th style={thStyle}>Actions</th> /* Add Actions header */
61             </tr>
62           </thead>
63           <tbody>
64             {data.map((item) => (
65               <tr key={item.id}>
66                 <td style={tdStyle}>{item.id}</td>
67                 <td style={tdStyle}>{item.titulo}</td>
68                 <td style={tdStyle}>{item.ano_de_lancamento}</td>
69                 <td style={tdStyle}>{item.quantidade_em_estoque}</td>
70                 <td style={tdStyle}>{item.disciplina}</td>
71                 <td style={tdStyle}>
72                   <button>Edit</button> /* Edit button */
73                   <button>Delete</button> /* Delete button */
74                 </td>
75               </tr>
76             ))}
77           </tbody>
78         </table>
79         <button style={buttonStyle} onClick={openModal}>
80           Adicionar Livro
81         </button>
82
83         /* Modal */
84         <Modal
85           isOpen={isModalOpen}
86           onRequestClose={closeModal}
87           contentLabel="Adicionar Livro Modal"
88         >
89           <h2>Adicionar Livro</h2>
90           /* Add insertable information inputs and form elements here */
91           <button onClick={closeModal}>Close Modal</button>
92         </Modal>
93       </div>
94     );
95   };

```



2.2.2 Lições Aprendidas

A Sprint 2 abordou a questão do FrontEnd utilizando a Linguagem React.js. Pude iniciar o projeto a partir das bibliotecas npm e observar as primeiras questões de rotas e componentização de métodos e classes para o React. A integração de Front com BackEnd segue para a próxima Sprint.

O código foi commitado e segue disponível no link a seguir:

<https://github.com/rafaelravelli12/Projects-Node.js>

2.3 Sprint 3 - FrontEnd + Deploy

2.3.1 Solução

- **Evidência do planejamento:**

A terceira Sprint contemplou eventuais ajustes da segunda Sprint, adição de filtros na tela de CRUD e o processo de Deploy da solução em ambiente Web.

- **Evidência da execução de cada requisito:**

- Correções da Sprint 2

- Dando prosseguimento ao término dos cards referentes à sprints dois, seguiu-se o desenvolvimento e usabilidade do frontEnd e sua integração ao backend. A tela de HomePage com a tabela necessária foi atualizada e está apresentando dados do banco de dados. A tela de adicionar elementos também foi atualizada e está funcionando corretamente, ou seja, a execução do método atualiza o banco de dados conforme esperado. As Figuras 29 e 30 ilustram o frontEnd enquanto as figuras 31 e 32 ilustram o método insert do backEnd.

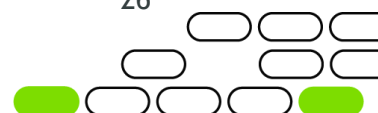


Figura 29 - HomePage (React.js)

- [Home](#)
- [Add Book](#)

Acervo Biblioteca

ID	Título	Ano de Lançamento	Quantidade disponível	Disciplina	Ações
1	Navegando pela História Moderna	2020	3	História	<div>Edit</div> <div>Delete</div>
2	Biologia para ensino fundamental	2015	5	Biologia	<div>Edit</div> <div>Delete</div>
3	Matemática de A a Z	2018	10	Matemática	<div>Edit</div> <div>Delete</div>
4	Manual da Língua Portuguesa	2022	18	Português	<div>Edit</div> <div>Delete</div>
5	English Course	2021	8	Inglês	<div>Edit</div> <div>Delete</div>
6	Física Moderna	2020	4	Física	<div>Edit</div> <div>Delete</div>

Figura 30 - Adicionar Livro ao acervo (React.js)

- [Home](#)
- [Add Book](#)

Add Book

Title:

Release Year:

Stock Quantity:

Discipline:



Figura 31 - BackEnd para execução do método insert (Node.js)

```

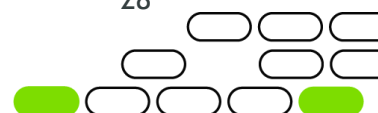
40 export const addBook = async (req, res) => {
41   try {
42     const { titulo, ano_de_lancamento, quantidade_em_estoque, disciplina } = req.body;
43
44     const connection = await pool.getConnection();
45     await connection.execute(
46       "INSERT INTO acervo (titulo, ano_de_lancamento, quantidade_em_estoque, disciplina) VALUES (?, ?, ?, ?)",
47       [titulo, ano_de_lancamento, quantidade_em_estoque, disciplina]
48     );
49     connection.release();
50
51     res.status(201).send("Book added successfully");
52   } catch (error) {
53     console.error("Error:", error);
54     res.status(500).send("Internal Server Error");
55   }
56 };

```

Figura 32 - FrontEnd para execução do método INSERT (React.js)

```

11   const handleSubmit = async (e) => {
12     e.preventDefault();
13     try {
14       const response = await fetch('http://localhost:4000/addBook', {
15         method: 'POST',
16         headers: {
17           'Content-Type': 'application/json',
18         },
19         body: JSON.stringify({
20           titulo: titulo || null,
21           ano_de_lancamento: ano_de_lancamento || null,
22           quantidade_em_estoque: quantidade_em_estoque || null,
23           disciplina: disciplina || null,
24         }),
25       });
26       if (response.ok) {
27         console.log('Book added successfully');
28       } else {
29         console.error('Failed to add book.');
```



- Criação de Filtros na Tela de CRUD
- Criação de Instância EC2 micro
- Criação de ambiente de deploy Apache e alocação de código
- Criação de Instância RDS micro para Banco de Dados

- **Evidência dos resultados**

Os demais métodos faltantes serão apresentados na entrega final

2.3.2 Lições Aprendidas

Nesta Sprint pude observar o funcionamento de um banco de dados relacional MySQL gerido por uma aplicação Node.js + express funcionar em conjunto com o frontEnd em React. Foram desenvolvidos elementos e componentes para estruturar as classes existentes. O commit do código na branch de desenvolvimento foi disponibilizado junto à entrega da Sprint.

3. Considerações Finais

3.1 Resultados

A Entrega final do Relatório buscou finalizar cartões das Sprints passadas. O software então passou a apresentar os elementos necessários da tela de Login e Dashboard CRUD. As Figuras 33 e 34 mostram o layout final observado na entrega para essas telas.

A tela de login compõe autenticação de rotas e validação de password a partir dos dados do database. A tela de Dashboard apresenta a todos os elementos CRUD necessários (ver, criar, atualizar e deletar itens), além de uma rota de logout. A Figura 35 apresenta a rota de Atualizar Livros, a título de exemplificação.

A consolidação desses elementos encerraram todos os cards referentes às Sprints 1 e 2.

Figura 33 - Página de Login (React.js)



← → ↻ ⓘ localhost:3000

Página de Login - Escola AVANÇO



Figura 34 - Página de Dashboard (React.js)

[←](#)
[→](#)
[↻](#)
localhost:3000

- [Dashboard](#)
- [Adicionar Livro](#)
- [Logout](#)

Acervo Biblioteca - Escola AVANÇO

ID	Título	Ano de Lançamento	Quantidade disponível	Disciplina	Ações
1	Navegando pela História Moderna	2020	4	História Nova	Edit Delete
2	Biologia para ensino fundamental	2015	5	Biologia	Edit Delete
3	Matemática de A a Z	2018	10	Matemática	Edit Delete
4	Manual da Língua Portuguesa	2022	18	Português	Edit Delete
5	English Course	2021	8	Inglês	Edit Delete
7	a	a	3	a	Edit Delete
10	a	a	1	a	Edit Delete
11	b	c	1	b	Edit Delete

Figura 35 - Página de Atualizar Livro (React.js)

[←](#)
[→](#)
[↻](#)
localhost:3000/edit/1

- [Dashboard](#)
- [Adicionar Livro](#)
- [Logout](#)

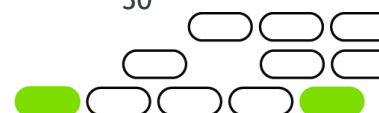
Edit Book

Title:

Release Year:

Stock Quantity:

Discipline:



A aplicação de filtros para busca de conteúdos na tabela principal da aplicação foi colocada em projetos Futuros.

Os cartões relacionados ao deploy na Sprint 3 foram colocados em projetos Futuros e deixaram de fazer parte do escopo do projeto na data de 25/09/2023. Ao invés, foi criado um arquivo README.me na raiz do projeto com explicações de como fazê-lo rodar localmente.

O código atual foi commitado na branch de Desenvolvimento e em sequência foi feito o ‘merge’ com a branch master.

3.2 Contribuições

O Desafio Proposto foi portanto resolvido conforme pretendido. Partindo-se do estudo de caso e da coleta de dados, foi possível levantar as principais dores da Escola AVANÇO e propor uma solução eficiente. Sua biblioteca passou a ser catalogada inteiramente digitalizada. Seu acervo atual foi listado e inserido na Database de referência.

Houve inovação e melhorias alcançadas em código, ao integrar duas linguagens de programação diferentes, prática comum ao mercado de trabalho. Node.js trabalhou como Backend, regendo regras de negócios e consultas ao banco de dados, enquanto o React.js trabalhou como Frontend, incorporando o usuário final ao sistema.

3.3 Próximos passos

Os próximos passos apontam para otimização da aplicação existente, visando melhor usabilidade. A integração entre telas pode ser melhorada, com rotas de retorno melhores definidas. Cartões existentes que foram postergados para Projetos Futuros também poderão ser executados.

4. Referências Bibliográficas

<https://www.linkdesignbrasil.com/a-evolucao-da-internet-ate-os-dias-atuais/>, acessado 31 de julho de 2023

Material de Apoio IGTI, acessado 04 de agosto de 2023

