UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

Gabriel Haddad Vieira

UM SISTEMA PARA APOIO A DECISÃO EM SPES DA CONSTRUÇÃO CIVIL

Santa Maria, RS
2023

Gabriel Haddad Vieira

UM SISTEMA PARA APOIO A DECISÃO EM SPES DA CONSTRUÇÃO CIVIL

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Ciência da Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Bacharel em Ciência da Computação**.

Orientador: Prof. Giovani Rubert Librelotto

Santa Maria, RS
2023

Gabriel Haddad Vieira

UM SISTEMA PARA APOIO A DECISÃO EM SPES DA CONSTRUÇÃO CIVIL

Trabalho de Conclusão de Curso apresentado ao Curso de Bacharelado em Ciência da Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Bacharel em Ciência da Computação**.

Aprovado em 17 de julho de 2023:

Giovani Rubert Librelotto, Dr. (UFSM)
(Presidente/Orientador)

Jonas Bulegon Gassen, Dr. (UFSM)

Rafael Teodósio Pereira, Dr. (UFSM)

Santa Maria, RS
2023

AGRADECIMENTOS

Primeiramente, sou extremamente grato a meus pais, Chalia e Romualdo. Nesta longa formação acadêmica, com seus altos e baixos, ambos sempre estiveram a meu lado trazendo toda a ajuda que podiam, sem medir esforços para permitir que eu concluísse essa graduação. Por muitos momentos foram a força que motivava a seguir adiante e fazer o melhor. Vocês são pessoas exemplares, e eu devo tudo à vocês.

Com muito carinho aos meus amigos Carlos Eduardo e Lana. Vocês foram imprescindíveis para que este trabalho fosse concluído. O apoio e companheirismo de vocês durante os últimos meses fez total diferença para que o caos não tomasse conta. Também à minha médica e, principalmente, amiga Claudia que, por muitas vezes, foi fonte de calmaria e tranquilidade quando mais se fez necessário.

Books dont need batteries.

(Nadine Gordimer)

RESUMO

UM SISTEMA PARA APOIO A DECISÃO EM SPES DA CONSTRUÇÃO CIVIL

AUTOR: Gabriel Haddad Vieira
Orientador: Giovani Rubert Librelotto

As Sociedades de Propósito Específico (SPEs) foram criadas para reduzir riscos financeiros, limitando seu objeto social a um único empreendimento. Na construção civil, as SPEs são usadas na construção de prédios comerciais e residenciais, proporcionando maior segurança aos sócios. Com a mesma tributação de construtoras tradicionais, elas isolam os riscos, gerando confiança entre as partes envolvidas. As SPEs frequentemente precisam adquirir materiais, serviços ou produtos durante o projeto. Com o Sistema Nacional da Nota Fiscal Eletrônica (NF-e) e a Nota Fiscal de Serviço Eletrônica (NFS-e), é possível padronizar a emissão das notas fiscais, facilitando a análise das movimentações financeiras e identificando informações relevantes sobre gastos e investimentos. Para elevar a segurança das SPEs e fortalecer a confiança dos sócios, propõe-se o desenvolvimento de uma plataforma que forneça suporte à tomada de decisão financeira. Essa plataforma web interativa apresentaria informações, gráficos e relatórios de gastos, destacando dados relevantes com base nas notas fiscais eletrônicas. O objetivo é construir uma plataforma que auxilie a tomada de decisão das SPEs. O desenvolvimento da aplicação envolveu um estudo de caso abrangendo as SPEs, notas fiscais e tecnologias disponíveis. Foram realizados planejamento, prototipação, desenvolvimento inicial, testes, melhorias e criação de novos módulos. A análise de diferentes cenários permitiu simular as operações dos usuários e fornecer métricas para avaliar o resultado final da proposta. O objetivo principal foi comparar o resultado obtido com os objetivos estabelecidos.

Palavras-chave: Sociedade de Propósito Específico. Nota Fiscal Eletrônica. Desenvolvimento Web.

ABSTRACT

TÍTULO DO TRABALHO EM INGLÊS

AUTHOR: Gabriel Haddad Vieira
ADVISOR: Giovani Rubert Librelotto

Specially Purpose Entities (SPEs) were created to reduce financial risks by limiting their social object to a single project. In the construction industry, SPEs are commonly used for the development of commercial and residential buildings, providing greater security for the stakeholders. With the same taxation as traditional construction companies, they isolate risks, fostering trust among the involved parties. SPEs often need to acquire materials, services, or products during project execution. With the implementation of the National Electronic Invoice System (NF-e) and the Electronic Service Invoice (NFS-e), a standardized regulation for invoice issuance is established. This enables the consolidation of financial transactions through invoices, facilitating analysis and identification of relevant information regarding project expenditures and investments. To enhance security in SPEs and solidify trust with stakeholders, the proposal suggests the development of a decision support platform for financial transactions. This interactive web platform would provide informative content, graphs, and expense reports, highlighting project-specific information derived from collected electronic invoices. By collecting, grouping, and presenting this information in a standardized format, the platform aims to assist and support decision-making processes within SPEs. The development of the proposed application involved a case study covering key topics related to SPEs, as well as the invoices documenting their financial transactions and the available technologies for implementing the decision support platform. It included planning, prototyping, and initial development, followed by testing, improvements, and the creation of new modules. Different usage scenarios were analyzed to simulate user operations and provide metrics for evaluating the final outcome of the proposal, comparing it against the established objectives.

Keywords: Specially Purpose Entity. National Electronic Invoice System. Web Development.

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE SIGLAS

API	Application Programming Interface
BSON	Binary JSON
CFOP	Código Fiscal de Operações e Prestações
CNPJ	Cadastro Nacional da Pessoa Jurídica
CRUD	Create Read Update Delete
CSS	Cascading Style Sheets
GTIN	Global Trade Item Number
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
ISS	Imposto sobre Produtos e Serviços
JS	JavaScript
JSON	JavaScript Object Notation
NCM	Nomenclatura Comum do Mercosul
NF-e	Nota Fiscal Eletrônica
NFS-e	Nota Fiscal de Serviço Eletrônica
OFX	Open Financial Exchange
REST	Representational State Transfer
SPE	Sociedade de Propósito Específico
SPED	Sistema Público de Escrituração Digital
XML	Extensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

Criada para reduzir riscos financeiros, a Sociedade de Propósito Específico - SPE - traz como característica principal ter seu objeto social limitado a apenas um empreendimento. Por não assumir dívidas ou encargos maiores do que seu capital social, sua forma de existência traz um maior nível de segurança para sócios e demais participantes da sociedade. Na construção civil as SPEs têm sido utilizadas para construção de prédios comerciais e residenciais. Possuindo mesma tributação de uma construtora tradicional, é capaz de isolar os riscos envolvidos trazendo segurança e confiança entre as partes envolvidas. Se faz dessa forma, pois o sócio investidor deve passar a confiar num administrador representador do negócio.

As SPEs na construção civil frequentemente necessitam adquirir materiais, serviços ou produtos durante sua existência em projeto. Com o Sistema Nacional da Nota Fiscal Eletrônica (NF-e) e Nota Fiscal de Serviço Eletrônica (NFS-e) a criação de um regulamento para padronização de emissão de NF-e e NFS-e passa a existir. Com isso, é possível agrupar as movimentações financeiras das SPEs por meio de suas notas. Contendo um padrão, é facilitada sua análise e estudo. Dessa forma vem a possibilidade de elencar informações de interesse em relação a gastos e investimentos do projeto.

A fim de elevar o nível de segurança presente em SPEs e solidificar a confiança entre esta e os sócios investidores, é fornecido o apoio para a tomada de decisão de SPEs em suas movimentações financeiras. A criação de uma plataforma que possa prover informações de interesse, que forneça esse apoio adicional, surge como proposta neste trabalho. Trazendo informativos, gráficos e relatórios de gastos, uma plataforma web interativa destaca informações pertinentes ao projeto na qual a SPE atua e as movimentações financeiras com base nas notas fiscais recolhidas. Coletando, agrupando e exibindo essas informações num padrão específico, se busca construir uma plataforma que ajude e apoie a tomada de decisão das SPEs.

Para o desenvolvimento da aplicação proposta, foi realizado um estudo de caso dos principais tópicos atrelados a essa proposta. Desde as SPEs, como também as notas fiscais que registram suas movimentações financeiras, assim como as tecnologias disponíveis para a implementação da plataforma de apoio à tomada de decisão. Na sequência, foi realizado o planejamento, prototipação e desenvolvimento da plataforma em uma versão primordial. O que se seguiu foram testes e melhorias aplicados na versão inicial obtida para o refinamento de suas funcionalidades e desenvolvimento de novos módulos.

Os métodos aplicados no desenvolvimento da proposta foram analisados em diferentes cenários. Esses cenários objetivaram simular as principais operações de um usuário da plataforma que busca obter o apoio à tomada de decisão. Com isso, as análises dos cenários de uso da aplicação proveram as métricas para o resultado definitivo da proposta.

Dessa forma foi possível observar sua construção como um todo e comparar o resultado obtido com o objetivo proposto.

O Capítulo 2 apresenta a proposta do projeto. Com isso, no Capítulo 3 é encontrada a revisão teórica e bibliográfica acerca dos principais tópicos de pesquisa para o trabalho. O Capítulo 4 demonstra a metodologia utilizada para pôr em prática o desenvolvimento proposto, de forma que no Capítulo 5 existem cenários que exploram a aplicação desenvolvida e seus resultados. Ao final, no Capítulo 6 é feita a análise dos resultados obtidos em relação à proposta realizada.

2 PROPOSTA

Atualmente, na Construção Civil, tem-se utilizado significativamente de Sociedades de Própositos Específicos (SPE) para a construção de prédios comerciais e residenciais. Nessa área, a SPE funciona com base em um relacionamento de confiança entre as partes envolvidas, pois o sócio investidor deverá confiar em um administrador, o qual irá representar o negócio perante os fornecedores e o fisco. Apesar de possuir a mesma tributação que uma construtora tradicional possui, a SPE se mostra como o tipo de sociedade mais aceita entre os investidores no ramo da Construção Civil.

O Sistema Nacional da Nota Fiscal de Serviço eletrônica (NFS-e) regulamenta um padrão nacional para emissão de NFSe. Não somente, mas também a construção de um repositório para o controle das NFS-e expedidas e disponibilização de um emissor de nota público. O projeto beneficia altamente as administrações tributárias, trazendo uma padronização e melhoramento na qualidade e informações. Além disso, gera maior eficiência na atividade fiscal e facilita análise de seus dados. Com a padronização e armazenamento de suas NFS-e, é possível criar um acervo de gastos envolvidos no projeto, e com isso analisar gargalos e problemas de percurso.

O apoio para tomada de decisão em SPEs traz a possibilidade de elevar ainda mais a segurança e confiança na relação de uma SPE e o sócio investidor. A possibilidade se dá com o desenvolvimento de uma aplicação Web para SPEs. Se baseado na análise de requisitos pertinentes a algumas necessidades básicas de uma SPE, a plataforma Web busca trazer esse apoio à tomada de decisão para SPEs da construção civil. O apoio trazido na forma de relatórios e informativos, mune as SPEs de maiores detalhes para apoiar sua tomada de decisão. Com isso, tanto facilita e provê mais informações para as necessárias fundamentar suas tomadas de decisões em seu caminho, quanto traz maior segurança e confiança entre a SPE e seus sócios investidores

Este trabalho apresenta uma proposta de desenvolvimento de plataforma web para apoio à tomada de decisão de SPEs em construção civil. Sua construção busca trazer, em relatórios e informativos, informações pertinentes que suportem as SPEs em suas decisões. Para tal, as informações pertinentes dos créditos e débitos da sociedade empresarial são necessárias. Tais informações serão chave para a análise de custos e gastos, produzindo relatórios e informativos importantes para a plataforma. Para isso, serão utilizadas notas fiscais de serviço eletrônicas de uma SPE de construção civil. Possuindo dados reais, a construção da plataforma se aproxima de uma verossimilhança com sua utilização em grande escala

3 REVISÃO TEÓRICA E BIBLIOGRÁFICA

Este capítulo apresenta uma revisão teórica e bibliográfica dos principais conceitos aplicados neste trabalho. A primeira seção explora as Sociedades de Propósito Específico e sua função, seguida diretamente por uma sessão onde se analisa a estrutura da NF-e e como seus dados podem ser de interesse. Na sequência são analisadas as principais tecnologias envolvidas para representação de dados do projeto. O capítulo é finalizado por uma análise da arquitetura REST, a qual é base para a construção do *back-end* da aplicação.

3.1 SOCIEDADES DE PROPÓSITO ESPECÍFICO NA CONSTRUÇÃO CIVIL

A indústria da construção civil frequentemente lida com projetos complexos que exigem uma gestão cuidadosa e uma estrutura de negócios adequada. Uma abordagem que tem ganhado destaque nesse setor é a criação de Sociedades de Propósito Específico (SPEs) (ALVES; FREITAS, 2020). Nesta seção, explora-se o conceito de SPEs na construção civil, seus benefícios e como elas podem contribuir para o sucesso de empreendimentos de grande porte.

Sociedades de Propósito Específico, como visto em Matias et al. (2007), são uma figura jurídica especial que constituem uma sociedade cujos ativos, funcionamento e gestão são determinados com objetivos específicos. Essas sociedades são constituídas para realizar uma atividade específica, como a construção de um edifício, uma ponte ou uma usina, e são dissolvidas após a conclusão do projeto ou o cumprimento de determinadas condições.

Os benefícios das SPEs na Construção Civil podem ser definidos como (BATISTA; MACHADO, 2021):

- Responsabilidade limitada: ao criar uma SPE, os envolvidos no projeto podem limitar sua responsabilidade aos ativos e passivos da própria entidade, protegendo seus outros interesses e patrimônio pessoal. Isso proporciona uma camada adicional de segurança para os investidores e acionistas envolvidos no empreendimento.
- Gestão especializada: as SPEs permitem que uma equipe de gestão altamente especializada seja designada para o projeto específico. Essa equipe pode ser composta por especialistas em engenharia, arquitetura, finanças e outras áreas relevantes, que trabalham em conjunto para garantir o sucesso do empreendimento.
- Captação de recursos: a criação de uma SPE pode facilitar a captação de recursos financeiros específicos para o projeto em questão. Investidores interessados em

financiar projetos de construção civil podem estar mais dispostos a participar de uma SPE, pois ela oferece maior transparência e clareza sobre como seus recursos serão utilizados.

- Controle e governança: as SPEs têm uma estrutura de governança clara e definida, o que ajuda a garantir que as decisões sejam tomadas de forma eficiente e transparente. Os acionistas e investidores podem participar ativamente nas deliberações estratégicas e no acompanhamento do progresso do projeto.
- Risco compartilhado: ao utilizar uma SPE, os riscos associados ao projeto são compartilhados entre os parceiros envolvidos. Isso permite que cada parte assuma uma parcela justa de riscos e recompensas, reduzindo assim a exposição a eventos imprevistos e incertezas.

As SPEs também podem ser utilizadas na construção civil para projetos de grande escala, como a construção de aeroportos, estádios, hospitais e complexos industriais. Empresas de engenharia, construtoras e investidores podem estabelecer uma SPE para garantir uma estrutura de negócios adequada, facilitar a gestão do projeto e promover a colaboração entre as partes interessadas.

Em resumo, as Sociedades de Propósito Específico desempenham um papel crucial na indústria da construção civil, permitindo que projetos complexos sejam executados de maneira eficiente e estruturada. Elas oferecem benefícios como responsabilidade limitada, gestão especializada, captação de recursos, controle e governança, além de compartilhamento de riscos. Ao adotar essa abordagem, os envolvidos podem minimizar os riscos, aumentar a eficiência e garantir o sucesso de empreendimentos desafiadores na construção civil.

3.2 NF-E

Afim de buscar viabilizar a integração concreta e padronização das informações entre os órgãos fiscalizadores, como o Ministério da Fazenda e a Receita Federal, em 2007 foi instituído o Sistema Público de Escrituração Digital (SPED) (RUSCHEL; FREZZA; UTZIG, 2011). Para este trabalho, será focado em apenas um dos quatro módulos do SPED: A Nota Fiscal Eletrônica (NF-e). A NF-e existe prévio a criação do próprio sistema do SPED, datada em 2005 (BRANCO, 2012). Sua existência age como um documento eletrônico para substituir a nota fiscal em papel, trazendo consigo todos os dados pertinentes da operação realizada. A NF-e não traz somente eficiência na facilitação da verificação de irregularidades fiscais, como também uma notável redução do espaço de armazenamento de documentos (GERON et al., 2011).

As notas fiscais eletrônicas, como também as notas fiscais de serviço eletrônicas, possuem papel principal no desenvolvimento da aplicação. São elas que marcam as transações de produtos e serviços, nesse caso as NFS-es, de uma SPE na construção civil. As notas fiscais são, em seu conceito, um arquivo no formato XML. Como dito anteriormente, as notas trazem consigo as informações necessárias à operação realizada em sua emissão, e as *tags* do formato XML são responsáveis por nomear e organizar esses campos. Nas Figuras 1, 2, 3, 4 temos trechos de uma nota fiscal que marca a venda de um produto.

Para a aplicação desenvolvida, essas quatro seções da NF-e indicadas nas seguintes figuras são de extrema importância e seus campos são amplamente utilizados. Na seção de informações, tem-se a capacidade da ordenação das notas fiscais por sua data e hora de emissão (*tag* <dhEmi>), bem como a classificação referente à natureza da operação. As seções de emitente e destinatário registram os fornecedores e prestadores de serviço ao destinatário da nota, nesse caso a SPE da construção civil. Por fim a seção de produtos é a responsável por prover as informações utilizadas nas gerações de gráficos e relatórios dos gastos com cada produto durante o projeto.

Figura 1 – Trecho de informações de uma NF-e.

```
<ide>
    <cUF>43</cUF>
    <cNF>89438335</cNF>
    <natOp>VENDA DE PRODUCAO DO ESTABELECIMENTO</natOp>
    <mod>55</mod>
    <serie>1</serie>
    <nNF>130576</nNF>
    <dhEmi>2020-05-29T14:13:55-03:00</dhEmi>
    <tpNF>1</tpNF>
    <idDest>1</idDest>
    <cMunFG>4310538</cMunFG>
    <tpImp>1</tpImp>
    <tpEmis>1</tpEmis>
    <cDV>1</cDV>
    <tpAmb>1</tpAmb>
    <finNFe>1</finNFe>
    <indFinal>1</indFinal>
    <indPres>1</indPres>
    <procEmi>0</procEmi>
    <verProc>CIGAM 180813.fRC117</verProc>
</ide>
```

Fonte: Autor.

3.2.1 Informações Gerais da NF-e

Os campos indicados na Figura 1 se destinam a informações gerais da nota. É de destacar os campos marcados pelas *tags* <natOp> que simboliza a natureza da operação da nota, equanto o campo na *tag* <dhEmi> indica a data e hora da emissão da nota fiscal.

Figura 2 – Trecho do emitente de uma NF-e.

```
<emit>
    <CNPJ>01234567890123</CNPJ>
    <xNome>NOME EMITENTE</xNome>
    <xFant>NOME FANTASIA EMITENTE</xFant>
    <enderEmit>
        <xLgr>LOGRADOURO EMITENTE</xLgr>
        <nro>012</nro>
        <xBairro>BAIRO EMITENTE</xBairro>
        <cMun>0123456</cMun>
        <xMun>MUNICIPIO EMITENTE</xMun>
        <UF>UF</UF>
        <CEP>01234567</CEP>
        <cPais>0123</cPais>
        <xPais>PAIS EMITENTE</xPais>
        <fone>01234567890</fone>
    </enderEmit>
</emit>
```

Fonte: Autor.

3.2.2 Emitente e Destinatário da NF-e

Inicialmente, para demonstrativo e afim de manter a discrição das partes envolvidas na transação comercial registrada pela NF-e, informações pertinentes das partes emitentes e destinatárias foram substituídas por informações genéricas. Tais informações pertinentes incluem CNPJ, nome (incluindo nome fantasia), bem como informações relacionadas a endereço e contato.

A orquestração das informações na seção emitente da NF-e também será encontrada na seção de destinatário, que possui basicamente as mesmas informações pertinentes ao documento. Ambas seções terão seus próprios CNPJ, nome e endereço registrados na nota.

3.2.3 Produto da NF-e

As *tags* XML encontradas na Figura 4 marcam cada item de interesse da padronização da NF-e em relação à seção do produto da nota. Em primeiro momento é fácil obter o produto da nota, identificado pela *tag* <xProd>, na qual para a nota referenciada temos "Brita N2 ou Brita 1". Na sequência temos a unidade comercial e tributável do produto, identificadas respectivamente com as *tags* <uCom> e <uTrib>, bem como a quantidade comercial e tributável, identificadas respectivamente com as *tags* <qCom> e <qTrib>. O valor unitário de comercialização do produto é marcado pela *tag* <vUnCom>, assim como seu valor unitário de tributação é marcado pela *tag* <vUntrib>. Por fim, o valor total da nota, a multiplicação do valor unitário de comercialização do produto pela quantidade comercial do mesmo é identificada pela *tag* <vProd>. Com esses valores iniciais já é possível ter um

Figura 3 – Trecho do destinatário de uma NF-e.

```
<dest>
    <CNPJ>01234567890123</CNPJ>
    <xNome>NOME DESTINATARIO</xNome>
    <enderDest>
        <xLgr>LOGRADOURO DESTINATARIO</xLgr>
        <nro>012</nro>
        <xBairro>BAIRO DESTINATARIO</xBairro>
        <cMun>0123456</cMun>
        <xMun>MUNICIPIO DESTINATARIO</xMun>
        <UF>UF</UF>
        <CEP>01234567</CEP>
        <cPais>0123</cPais>
        <xPais>PAIS DESTINATARIO</xPais>
        <fone>01234567890</fone>
    </enderDest>
</dest>
```

Fonte: Autor.

compreensão geral da movimentação financeira registrada pela NF-e.

Adicionalmente há mais campos na seção de produtos em uma NF-e. A *tag* <cProd> representa um código interno do produto referenciado, assim como a *tag* <cEANTrib> identifica o número referente ao código de barras do produto. No caso do produto exemplo, não há a presença de GTIN (Global Trade Item Number - Número Global do Item Comercial), o número diretamente abaixo do código de barras. Além disso, a *tag* CFOP classifica os produtos da nota de acordo com operação, entrada ou saída, enquanto a *tag* NCM é referente ao código do produto em relação à Nomenclatura Comum do Mercosul (NCM).

3.3 TECNOLOGIAS UTILIZADAS

Durante o desenvolvimento da aplicação analisada no trabalho, onde uma alta quantidade de dados é manipulada e armazenada a fim de registro e estudo, é necessário abordar e destacar duas estruturas cruciais que desempenham papéis fundamentais.

Inicialmente, para a organização e documentação das notas fiscais, foi identificada a necessidade de manipular documentos com a estrutura XML. As notas fiscais utilizadas nesse contexto abrangem as transações fiscais e fornecem informações essenciais para a geração de relatórios e gráficos informativos. Essas notas fiscais seguem um padrão de definição nacional que se baseia na estrutura XML. A estrutura XML é particularmente adequada para a organização das notas fiscais, pois permite a representação hierárquica dos dados. Por meio do uso de tags bem definidas, é possível expressar com precisão cada elemento presente nas notas fiscais, como informações de identificação, detalhes do produto, valores, datas e outros campos relevantes. Essa estrutura oferece flexibilidade

Figura 4 – Trecho do produto de uma NF-e.

```
<prod>
    <cProd>01100000007</cProd>
    <cEAN>SEM GTIN</cEAN>
    <xProd>BRITA N 2  OU BRITA 1</xProd>
    <NCM>25171000</NCM>
    <CFOP>5101</CFOP>
    <uCom>TN</uCom>
    <qCom>95.8700</qCom>
    <vUnCom>52.0000000000</vUnCom>
    <vProd>4985.24</vProd>
    <cEANTrib>SEM GTIN</cEANTrib>
    <uTrib>TN</uTrib>
    <qTrib>95.8700</qTrib>
    <vUnTrib>52.0000000000</vUnTrib>
    <indTot>1</indTot>
</prod>
```

Fonte: Autor.

para a manipulação e a extração de dados, facilitando a análise posterior. Ao aderir às tags e aos padrões definidos e devidamente documentados, é possível mapear os campos contidos nessas notas fiscais para um segundo formato que também desempenha um papel crucial na aplicação: o formato JSON.

No entanto, para alguns propósitos específicos, como integração com outros sistemas e facilidade de leitura e processamento em determinados cenários, o formato JSON se mostra altamente vantajoso. Ele é amplamente utilizado na troca de dados entre sistemas, especialmente em aplicações web. A representação dos dados em JSON é simples e compacta, tornando-a mais legível e fácil de ser interpretada tanto por humanos quanto por máquinas. Tanto o *front-end* quando *back-end* da aplicação são baseados na linguagem de programação JavaScript. A notação de objeto JavaScript (JSON) foi utilizada extensamente e é como as informações e dados são manipulados e gerenciados pelo sistema.

Tanto o XML quanto o JSON são formatos de representação de dados estruturados. Ambos possuem uma hierarquia de elementos e atributos que podem ser facilmente mapeados entre si. Num documento XML, as *tags* com seus atributos podem ser mapeadas para as chaves e valores de um objeto JSON.

3.3.1 XML

XML significa eXtensible Markup Language (CONSORTIUM et al., 2006), linguagem de marcação extensível ao ser traduzida para o português. A fundamentação de um documento XML se baseia na simplicidade e a facilidade de leitura. XML é um subproduto da SGML, Standard Generalized Markup Language, uma metalinguagem na qual se pode

definir linguagens de marcação para documentos. Sua estrutura permite que seus usuários definam padrões de marcação para documentos ou usem esquemas automatizados para tais definições (NURSEITOV et al., 2009). Sua estrutura de marcação com tags de fácil leitura, como projetado, tem como objetivo ser diretamente usável através da internet, sendo razoavelmente clara (CONSORTIUM et al., 2006). Na Figura 5 ilustra-se um exemplo simples da um esquema de *tags* em XML para identificar uma pessoa, utilizando seu nome, sobrenome e idade.

Figura 5 – Exemplo de esquema XML.

```
<pessoa>
    <nome> Gabriel </nome>
    <sobrenome> Haddad </sobrenome>
    <idade> 24 </idade>
</pessoa>
```

Fonte: Autor.

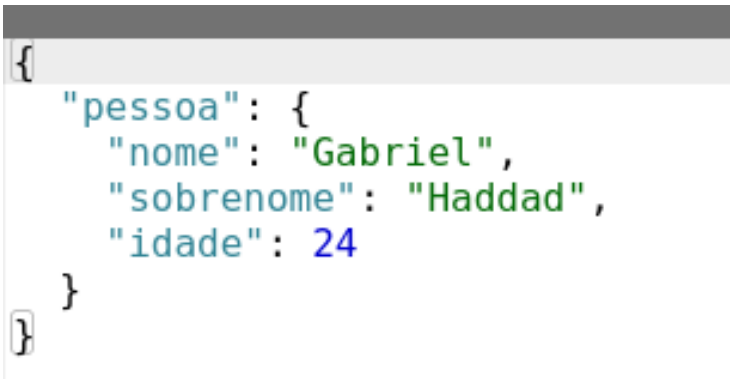
No decorrer deste trabalho, o tópico referente a NF-es, as notas fiscais eletrônicas, será constante e de importância. Seu padrão nacional é definido por um esquema em XML, na qual as *tags* identificam seções e informações de interesse da transação na qual a NF-e é emitida. Existem *tags* para identificar o emitente e destinatário da nota, bem como informações gerais e produtos envolvidos na transação. A arquitetura da nota no formato XML, e sua migração para o formato JSON, serão tópicos discutidos em seções futuras.

3.3.2 JSON

JSON é sigla para *JavaScript Object Notation*, ou notação de objeto JavaScript, um formato de dados baseado nos tipos de dados da linguagem de programação JavaScript (BRAY, 2014). Nos últimos anos, JSON ganhou grande popularidade entre os desenvolvedores de aplicações web, se tornando o principal formato para troca de informações através da rede (PEZOA et al., 2016). Além disso, sua estrutura de fácil entendimento ambos por máquina quanto desenvolvedor, possui importância em sua popularidade. Na Figura 6 ilustra-se o mesmo exemplo demonstrado na seção anterior, a criação de um objeto JSON para identificar uma pessoa utilizando seu nome, sobrenome e idade.

Ambas estruturas possuem campos autoexplicativos, as *tags* XML e as chaves do JSON, cada um com seus respectivos valores. A facilidade do mapeamento dessas estruturas entre si é de suma importância para o desenvolvimento da aplicação. Enquanto a aplicação seria desenvolvida em sua extensão e utilizando a linguagem de programação JavaScript, as notas fiscais utilizadas estariam no formato XML. Em outros cenários, e uti-

Figura 6 – Exemplo de objeto JSON.



Fonte: Autor.

lizando outros formatos de representação de dados, o mapeamento de um formato a outro poderia se mostrar desafiador e complexo. Ao observar os exemplos de ambas Figuras 5 e 6, temos formatos intercambiáveis entre si próprios para a aplicação desenvolvida.

3.4 ARQUITETURA REST

O conceito da arquitetura Representational State Transfer (REST) - ou em português, tansferência de estado representacional - foi primeiramente apresentado por Roy Fielding, no capítulo homônimo de seu livro *Architectural Styles and the Design of Network-based Software Architectures* (FIELDING, 2000). Como as demais arquiteturas, REST possui seus próprios princípios e requisitos que necessitam ser seguidos. No livro supracitado são listados os princípios e requisitos de uma arquitetura RESTful. Para o desenvolvimento da API utilizada no sistema, são de destaque e de necessidade detalhar alguns dos princípios da arquitetura.

3.4.1 *Client-Server*

Existe uma limitação na relação arquitetônica cliente-servidor. O principal ponto que define a limitação cliente-servidor é o conceito de independência mútua. Ao se separar as necessidades da interface do usuário das necessidades do servidor, cria-se um aumento no nível de portabilidade e escalabilidade. O aumento no nível de portabilidade se baseia no cliente necessitando saber apenas o mínimo, e podendo existir em múltiplas plataformas. Enquanto isso o aumento em escalabilidade é trazido pela simplificação dos componentes do servidor. Com essa limitação e construção, há a possibilidade de evolução mútua, mas independente, de ambas pontas da aplicação.

3.4.2 *Stateless*

Mais uma limitação na relação cliente-servidor é o conceito de não haver estado. A comunicação, ou requisição, de um cliente para um servidor necessita conter todas as informações necessárias para a compreensão da chamada. Não se deve poder usar de nenhuma informação previamente armazenada no servidor. O estado da sessão acaba por ficar totalmente na ponta cliente da aplicação, sendo responsável por construir uma requisição com as informações necessárias. Na adição dessa limitação, criam-se propriedades de visibilidade e confiabilidade, além de também aumentar a escalabilidade da aplicação.

A visibilidade é inserida com maior enfoque no campo de monitoramento do sistema. Apenas uma única requisição é necessária ser analisada para compreender sua execução por completo, já que a requisição traz toda informação pertinente ao serviço de sua chamada. A confiabilidade vem ao facilitar a tarefa de recuperar-se de falhas parciais (KENDALL et al., 1994). E, por fim, o aumento na escalabilidade se dá ao não precisar armazenar o estado da aplicação entre uma requisição e outra. Isso simplifica a implementação na ponta servidor da aplicação, ao não precisar gerenciar tais recursos, além de permitir um acesso mais veloz aos recursos solicitados.

Contudo, essa escolha arquitetônica traz suas desvantagens. Partindo do fato em que informação nenhuma é deixada no servidor em uma forma compartilhada para uso futuro, pode ocorrer uma diminuição do desempenho da rede, visto que múltiplas requisições serão feitas, muitas com informações replicadas. Além disso, há o problema de deixar a responsabilidade do estado da aplicação na ponta cliente. Isso acarreta num servidor sem um controle consistente do comportamento da aplicação, já que agora ela depende da implementação correta dentre múltiplas versões do cliente

4 METODOLOGIA

A metodologia para desenvolvimento do trabalho foi dividida em duas etapas. A primeira dessas consiste no estudo das tecnologias existentes e suas vantagens e desvantagens. É de destaque a importância dessa fase antes da programação de fato da aplicação. Com inúmeras tecnologias para implementação de aplicações web, surgia a necessidade de uma análise preemptiva das que proveriam mais benefícios para a aplicação. Com isso em mente, foram estudadas alternativas para a interface de usuário, bem como o servidor que gerenciaria as informações e o banco de dados que as armazenaria.

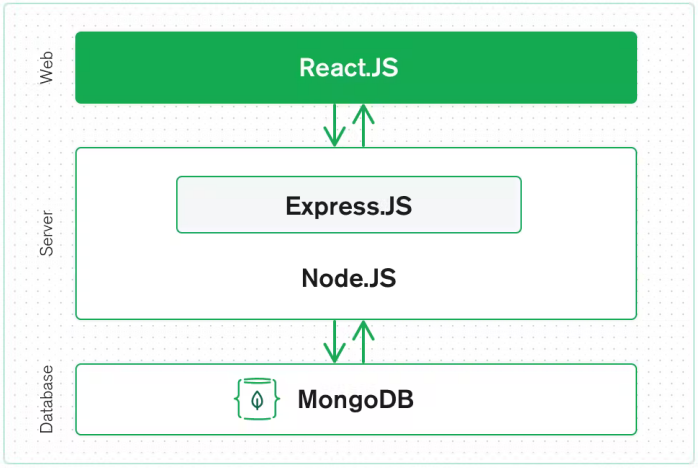
Para a interface de usuário, foi escolhida a linguagem de programação JavaScript combinada a sua biblioteca React. Além de ser altamente usada no mercado, quando combinada com sua biblioteca React que traz componentes reutilizáveis e demais recursos, JavaScript se mostrou uma benéfica alternativa para a aplicação. Além disso, a experiência prévia do autor no desenvolvimento com a linguagem para este fim, bem como a linguagem JavaScript e outras bibliotecas, teve impacto na decisão.

No que tange o servidor da aplicação, foi escolhido o ambiente Node.js. Sua estrutura, além de ser baseada também em JavaScript, traz foco em operações I/O (input/output, entrada e saída). Além disso, seu *framework Express* para gerenciamento de rotas seria de grande auxílio para a escalabilidade e flexibilidade do trabalho. Para finalizar o ecossistema do projeto, manteve-se a escolha da linguagem JavaScript no banco de dados MongoDB. Mongo utiliza em sua estrutura coleções e documentos que respeitam a arquitetura JSON, e com isso seria facilmente agregado ao projeto. Além disso, seu alto desempenho traria ainda mais performance à aplicação. A Figura 7 demonstra a arquitetura da plataforma dadas as tecnologias escolhidas.

A segunda etapa da metodologia do projeto foi o desenvolvimento da aplicação como um todo. De início foi realizado o projeto e especificação do trabalho, buscando levantar os requisitos básicos da aplicação e as necessidades do usuário. Com esses fatores enumerados, foi possível desenvolver uma versão básica da aplicação que proveria as funcionalidades levantadas. O sistema possuindo seus requisitos básicos para manipulação de notas fiscais, a sequência foi dada para a geração de relatórios informativos ao usuário. Para isso foram projetados e desenvolvidos relatórios com parâmetros definidos pelo usuário os quais proveriam o apoio à tomada de decisão.

Ainda nessa mesma etapa, vieram o planejamento e execução de melhorias ao sistema. De forma que as funcionalidades desenvolvidas foram revisadas e otimizadas em busca de aumentar seu desempenho, como também analisar possíveis pontos de melhorias e novos desenvolvimentos. Com isso mais funcionalidades foram acrescidas à plataforma, trazendo ainda mais recursos e informações que elevassem ainda mais o apoio àquela SPE no que tange suas decisões financeiras. Por fim, com uma aplicação concreta

Figura 7 – Estrutura das tecnologias do projeto.



Fonte: https://www.mongodb.com/mern-stack

e sólida foram analisadas e planejadas futuras implementações, de recursos e melhorias.

Ambas etapas de metodologia para implementação da aplicação podem ser vistas com maior detalhe nas seções seguintes.

4.1 *BACK-END*

O *back-end* de uma aplicação web desempenha um papel crucial na sua funcionalidade e segurança. Ele é responsável pelo processamento de dados, lógica de negócios e interação com o banco de dados. Utilizando linguagens de programação como Python, Java, JavaScript, o *back-end* implementa algoritmos, manipula informações e estabelece conexões com outras partes do sistema. Além disso, ele possibilita a criação de APIs para integração com outros sistemas. Um *back-end* bem projetado é essencial para garantir a eficiência, segurança e escalabilidade de uma aplicação web.

O *back-end* da aplicação segue seu desenvolvimento como uma REST API. É de extrema importância o conhecimento do conceito de APIs RESTful para a compreensão do funcionamento do *back-end* da aplicação desenvolvida. Uma API define uma coletânea de regras e a especificação da comunicação de um cliente com os componentes de um serviço (REDDY, 2011). Uma API RESTful é uma API que segue os princípios, limitações e construção da arquitetura REST.

Para a implementação do *back-end* da aplicação, foi utilizado o ambiente de execução JavaScript Node.js. Sua arquitetura focada em operações I/O, baseada em uma

única *thread* que não bloqueia o cliente, além de seu grande desempenho foram pontos importantes para a escolha de desenvolvimento. Acompanhando Node.js também foi utilizado o *framework* para aplicações web Express. A escolha da edição do *framework* para a aplicação se baseia em seu funcionamento de simplificar o desenvolvimento no ambiente Node.js e trazer inúmeras mais funcionalidades.

Já para a escolha do banco de dados da aplicação, seguiu-se a implementação com a escolha do MongoDB. Sua metodologia baseada em NoSQL quando comparada aos tradicionais bancos SQL (BOICEA; RADULESCU; AGAPIN, 2012) ao se mostrar superior gerenciando grandes volumes de dados foi o principal ponto de escolha para a tecnologia. Além disso, sua construção baseada em documentos majoritariamente no estilo JSON, onde campos podem ser facilmente acrescidos e removidos traz uma consistência de linguagem, bem como a flexibilidade necessária para aplicação. Com um *front-end* e *back-end* baseados em JavaScript e manipuladores de alta quantidade de objetos JSON, MongoDB se mostrou o banco de dados pertinente à aplicação.

4.1.1 Node.js

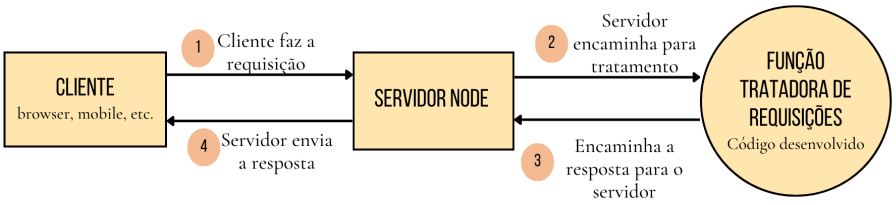
Para a implementação do servidor da aplicação foi utilizado o ambiente focado em operações I/O. Node.js é um ambiente, de código aberto e multiplataforma, de execução Javascript (OPENJS FOUNDATION, 2011). Seu funcionamento baseado numa única *thread* baseada em eventos reduz a sobrecarga gerada em sistemas baseados em múltiplas *threads*. Com isso, o ambiente se mostra extremamente capaz de rapidamente entregar informação de, e para, um servidor web (BANGARE et al., 2016).

Ao cliente fazer uma chamada para o servidor, por exemplo, não há bloqueio durante a espera de uma resposta. Desta forma, aplicações focadas em I/O são beneficiadas, enquanto aplicações focadas em CPU não terão o mesmo caminho. Por mais que o banco de dados da aplicação seja majoritariamente focado em I/O, balanceando a perda em relação ao servidor nesse caso, para essa aplicação este fato apenas aumenta a performance do desenvolvimento.

Na Figura 8 podemos compreender de forma simples o funcionamento do comportamento do Node.js. Inicialmente o cliente faz uma requisição diretamente ao servidor. Ele, por sua vez, encaminha a requisição para uma função tratadora de requisições, essa desenvolvida pelo programador. A requisição é processada e a função devolve uma resposta para o servidor. Por fim o servidor encaminha a resposta ao cliente.

Enquanto isso, na Figura 9, o trecho de código exemplifica a implementação de um simples servidor Node.js. Um servidor é facilmente criado com o método *createServer()* do módulo HTTP. O servidor é configurado para o par porta e nome do hospedeiro, e assim que é criado sua *callback* informa seu funcionamento. Ao receber uma requisição,

Figura 8 – Demonstrativo do funcionamento do Node.js.



Fonte: Autor.

um evento é disparado provendo dois objetos, o par requisição-resposta. O objeto requisição traz as informações da chamada, enquanto o objeto resposta é uma função *callback* utilizada para retornar o resultado da chamada para o cliente.

Figura 9 – Exemplo de código em Node.js.

```
const http = require('http');

const hostname = '127.0.0.1';
const port = 3000;

const server = http.createServer((req, res) => {
  res.statusCode = 200;
  res.setHeader('Content-Type', 'text/plain');
  res.end('Hello World\n');
});

server.listen(port, hostname, () => {
  console.log(`Server running at http://${hostname}:${port}/`);
});
```

Fonte: Autor.

Nesta aplicação, onde as chamadas carregam quantidades elevadas de informação, esse tipo de arquitetura é essencial para um nível elevado de desempenho. Além disso, em comparação com outras abordagens para *back-end*, como PHP ou Python, Node.js se mostra mais eficiente e com maior performance. Em Lei, Ma e Tan (2014) é possível obter dados que demonstram uma performance de Node.js duas vezes maior em comparação com PHP, e entre seis e sete vezes maior quando comparado com Python.

Além disso, seu funcionamento baseado em JavaScript foi fundamental para a escolha de um banco de dados compatível. Tal escolha buscava um recurso tão rápido e escalável quanto o próprio Node.js, a fim de alcançar o máximo de performance ao usuário possível. Para isso foi escolhido MongoDB. Seu sistema também funciona baseado em

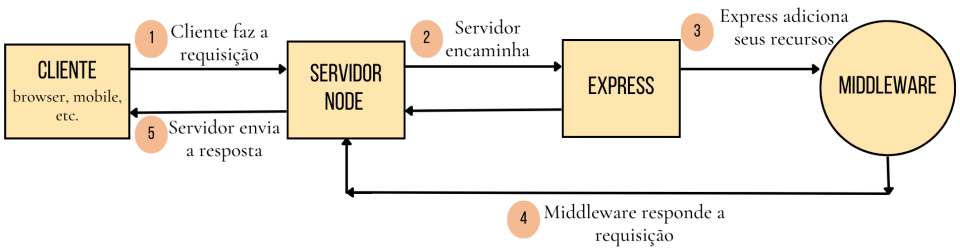
JavaScript, além de seus documentos serem registrados no formato BSON - um JSON binário. Com isso MongoDB se mostrou a melhor alternativa para banco de dados da aplicação.

4.1.1.1 Express

Express é um *framework* flexível para aplicações web com Node.js que provê um robusto conjunto de recursos para aplicações web e mobile (OPENJS FOUNDATION, 2022). Seu funcionamento como uma leve camada sobre o Node.js torna mais agradável o desenvolvimento de aplicações. Isso se dá por ser capaz de simplificar o funcionamento da API e abstrair muito de sua complexidade ao adicionar recursos úteis ao desenvolvimento. O envio de arquivos, por exemplo, não é uma tarefa simples utilizando Node.js puro. Ao utilizar o *framework* citado, o trabalho é facilmente reduzido e simplificado.

Em Hahn (2016), temos: "[...] permite refatorar um tratador de requisições único e monolítico em vários tratadores menores que atuam em partes e seções específicas. Isso é mais sustentável e mais modular". Essa capacidade de simplificação de código e quebra em partes menores e mais sustentáveis foi chave para a escolha do *framework* para a aplicação. Nesse desenvolvimento, que envolve um alto envio de arquivos e uma elevada quantidade de requisições, Express se mostrou chave para o desenvolvimento. Na Figura 10 é possível compreender melhor o espaço em que o *framework* habita na aplicação.

Figura 10 – Demonstrativo do funcionamento do Express.



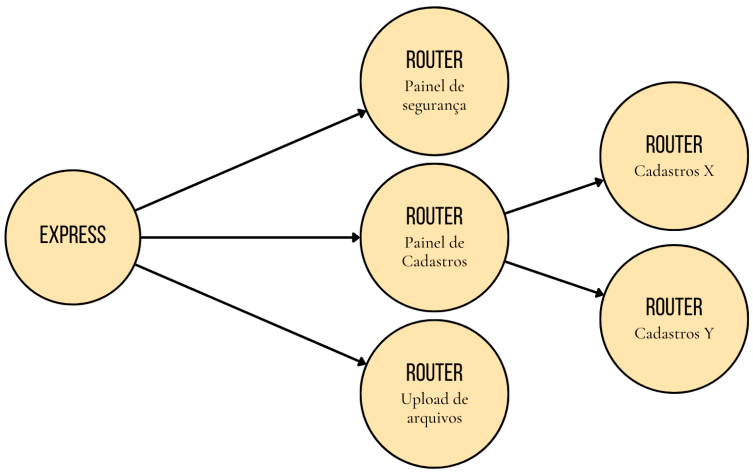
Fonte: Autor.

É perceptível a forma como os recursos do Express evitam a demorada implementação de uma estrutura (BROWN, 2019). Dos recursos providos pelo *framework* Express, o com destaque mais expressivo para se tornar chave para o desenvolvimento foi o con-

ceito de *routers*. Ao desenvolver uma aplicação maior, surge a necessidade de tornar o servidor mais escalável e sustentável. Por mais que o *framework* traga simplicidade, o conceito de flexibilidade é chave para escalar o servidor no qual opera. O conceito de *routers* quebra uma grande aplicação em pequenas aplicações numa orquestração construída pelo usuário.

Com isso, é possível criar um *router* que apenas trata de rotinas referentes ao *upload* de arquivos. Da mesma forma um outro *router* responsável por tratar cadastros da aplicação, ou múltiplos *routers* individuais para cada cadastro necessário. Essa flexibilidade na capacidade de reorquestrar e quebrar um código monolítico em partes menores e individuais se mostra indispensável na manutenção de uma grande aplicação ou uma aplicação em crescimento. Na Figura 11 vê-se um exemplo de arquitetura dos *routers* quando utilizados numa aplicação.

Figura 11 – Exemplo de arquiteturas de *routers* Express.



Fonte: Autor.

4.1.2 Mongo DB

> Bancos de dados relacionais são amplamente usados em um grande volume de aplicações e possuem boa performance quando tratam de uma limitada quantidade de dados. Para tratar um grande volume de informação como a internet, multimídia ou mídias sociais, o uso dos tradicionais bancos de dados relacionais é ineficiente. Para superar esse problema, o conceito de 'NoSQL' como metodologia foi introduzido. (GYŐRÖDI et al., 2015).

Para a aplicação desenvolvida que visa manipular um alto volume de dados e informações, além de buscar as melhores ferramentas para isto, essa definição impacta a escolha de

um banco de dados para o desenvolvimento. Um banco de dados baseado em NoSQL foi prontamente escolhido buscando a eficiência e desempenho no projeto.

Em oposição aos bancos de dados relacionais tradicionais, a capacidade de gerenciar informações desestruturadas, como documentos, e multimídia de forma eficiente é a principal vantagem do sistema NoSQL. É possível visualizar esse resultado tanto no trabalho de Khan e Mane (2013), como também no trabalho de Wei-Ping, Ming-Xin e Huan (2011). Seu funcionamento é baseado não num esquema fixo em tabelas, mas em chaves identificadoras, e a informação é encontrada através dessas chaves. Há algumas estratégias para o armazenamento de informações em bancos de dados não-relacionais como o utilizado na aplicação desenvolvida. Elas podem ser mais profundamente estudadas em (FRĂTEAN,). Para este documento será focado na estratégia baseada em documentos, a qual é utilizada no banco de dados escolhido.

MongoDB é um banco de dados NoSQL baseada em documentos. Seu funcionamento não utiliza o conceito de tabelas, comumente encontradas em bancos de dados tradicionais, mas sim documentos e coleções. Em suas coleções, seus documentos são baseados principalmente nos formatos JSON (Java Script Object Notation) e BSON (Binary JSON). Uma coleção não possui um esquema pré-definido como uma tabela tradicional, mas documentos possuidores de uma série de campos, documentos estes que podem ser vistos como linhas na coleção. Cada documento possuirá um campo de ID que será utilizado como chave primária.

Como visto anteriormente (WANG; TANG, 2012) é possível retificar os problemas de capacidade de bancos de dados tradicionais que gerenciam altos volumes de informação não estruturada utilizando de sistemas baseados em NoSQL, como o MongoDB ou, por exemplo, o Cassandra. Por seus documentos serem principalmente baseados nos formatos citados, seu funcionamento se uniria facilmente com as tecnologias baseadas em JavaScript empregadas para o front-end e *back-end* da aplicação. Além de sua flexibilidade, seu alto desempenho quando comparado a outros bancos de dados foi de extrema importância para a aplicação. No trabalho de Jose e Abraham (2020) é possível visualizar resultados que mostram o maior nível de performance de bancos de dados NoSQL, no caso MongoDB, quando comparados com bancos de dados tradicionais, como o MySQL. Além disso, como apresentado em Boicea, Radulescu e Agapin (2012) também podemos ver como MongoDB se destaca ao manipular altos volumes de dados quando comparado a um banco de dados baseado em SQL como o Oracle.

4.2 *FRONT-END*

O *front-end* de uma aplicação web desempenha um papel fundamental na sua usabilidade e interação com o usuário. Ele é responsável pela apresentação visual e pela

interface através da qual os usuários interagem com a aplicação. Utilizando tecnologias como HTML, CSS e JavaScript, o *front-end* traduz os dados e a lógica de negócios fornecidos pelo *back-end* em uma experiência de usuário envolvente e intuitiva. Ele também é responsável por garantir a responsividade da aplicação, adaptando-se a diferentes dispositivos e tamanhos de tela. Além disso, o *front-end* lida com validação de entrada de dados, gerenciamento de eventos e integração de recursos externos, como APIs e bibliotecas. Um *front-end* bem projetado é essencial para fornecer uma experiência de usuário de alta qualidade, garantindo a usabilidade, o desempenho e a acessibilidade da aplicação web.

Para a implementação do *front-end* da aplicação, foi escolhido o JavaScript como a linguagem principal. JavaScript é uma escolha popular e amplamente utilizada no desenvolvimento *front-end* devido à sua capacidade de prover ao *browser* recursos de comunicação assíncrona e interatividade (KUMAR; SINGH, 2016). Da mesma fora, JavaScript traz o suporte a diversas bibliotecas e *frameworks* que facilitam o desenvolvimento de uma aplicação. Com JavaScript, é possível criar uma experiência de usuário interativa e responsiva, tornando a aplicação mais envolvente para os usuários. Um *framework* permite uma programação mais estruturada, uniforme e simples (KALUŽA; TROSKOT; VUKELIĆ, 2018), pontos cruciais para a aplicação. Com isso em mente, foi escolhido um *framework* de JavaScript como a principal ferramenta de desenvolvimento. Além disso, a escolha de JavaScript para o *front-end* está em sintonia com o uso do ambiente de execução Node.js no *back-end*. Essa escolha permite a utilização da mesma linguagem em todo o ecossistema da aplicação, facilitando a comunicação e compartilhamento de código entre as camadas *front-end* e *back-end*.

4.2.1 React

O React é um *framework* JavaScript amplamente conhecido e utilizado por desenvolvedores por sua simplicidade, além de concisão e efetividade (RAWAT; MAHAJAN, 2020). Uma das principais razões para a escolha do React é a sua abordagem baseada em componentes. Com o React, é possível construir interfaces de usuário divididas em componentes independentes e reutilizáveis (AGGARWAL et al., 2018), o que facilita a manutenção e o desenvolvimento escalável da aplicação. Além disso, o React utiliza uma técnica chamada "Virtual DOM" que otimiza o processo de renderização, resultando em uma experiência de usuário mais rápida e responsiva (AGGARWAL et al., 2018). Ele também possui uma sintaxe declarativa e intuitiva, tornando o código mais legível e de fácil compreensão.

Com todas essas vantagens, a escolha do React como *framework* para o *front-end* da aplicação proporciona uma base sólida para o desenvolvimento de interfaces de usuário interativas, flexíveis e de alto desempenho.

4.3 DESENVOLVIMENTO

Para desenvolvimento da aplicação web com objetivo de prover suporte à tomada de decisão, o projeto foi inicialmente dividido em fases para seu desenvolvimento. A fase inicial do projeto foi dedicada ao planejamento e especificação de sua implementação, de forma em que foi realizado o levantamento dos requisitos básicos da aplicação e como seriam soluções para o objetivo proposto. A fase que se seguiu foi a de implementação dos requisitos básicos da aplicação. Após isso, o trabalho foi tomado pelo planejamento dos relatórios a serem expostos para o usuário para auxílio a tomada de decisão, bem como sua implementação no projeto. Após isso foram analisadas melhorias de projeto e demais funcionalidades que fossem de interesse a SPE.

No capítulo de cenários é possível visualizar diferentes ambientes de teste e fluxos de operação os quais demonstram os desenvolvimentos das diferentes etapas deste projeto.

4.3.1 Projeto e Especificação

De princípio, o requisito básico do sistema era a capacidade do usuário de enviar notas fiscais para serem armazenadas e futuramente estudadas. Ao analisar as notas fiscais, duas necessidades de projeto surgiram para esse requisito.

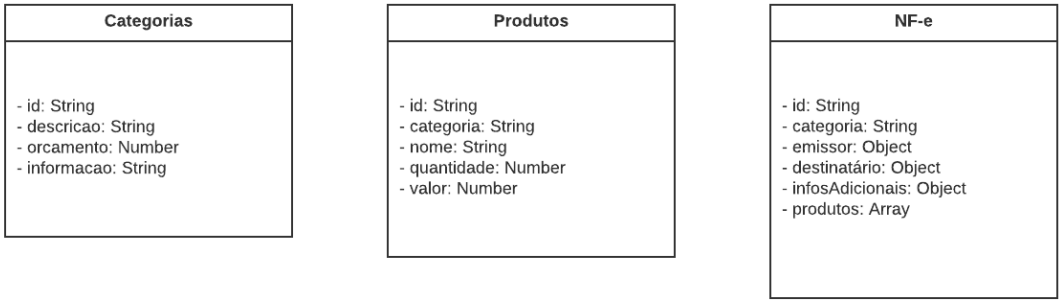
A primeira seria uma forma de catalogar as notas e as agrupar de acordo com a necessidade do usuário. Para isso, tomou-se a decisão de projeto de organizar as notas fiscais por categorias cadastráveis pelo usuário. Essas categorias serviriam para organizar as notas perante à qual área orçamentária de um projeto da construção civil elas seriam pertencentes. Dessa forma, notas fiscais referentes ao orçamento de materiais, por exemplo, seriam atribuídas a uma categoria criada para esse propósito. Da mesma forma, NF-es referentes a serviços hidráulicos poderiam ter uma categoria homônima a área. Com isso em mente, o usuário precisaria ter acesso a um CRUD (Create Read Update Delete - Criar Ler Atualizar e Remover) para as categorias. Para as necessidades do projeto, todas categorias criadas possuiriam uma descrição, um orçamento e um campo para informações adicionais.

A segunda necessidade se dava por uma nota fiscal possuir, em si, uma lista de produtos registrados na transação a qual representa. A categorização anterior atribuía uma área orçamentária a uma nota, entretanto uma NF-e pode conter em si diversos produtos. O que fazer quando nem todos produtos pertencentes à nota se encaixam na mesma categoria? Com isso, os produtos de uma nota foram armazenados como objetos individuais, independentes entre si, a fim de permitir o usuário manipular sua categoria. Inicialmente, um objeto produto possuiria como campos seu nome, valor unitário e quantidade, como

vistos na Figura 4.

Com essas necessidades sanadas, foi possível projetar o objeto da nota fiscal. Neste momento, as notas já poderiam ser categorizadas e organizadas, e informações pertinentes aos seus valores e produtos já haviam sido delegadas a um objeto secundário. Tendo isso em mente, informações como o emissor e o destinatário da nota, como visto respectivamente nas Figuras 2 e 3, bem como informações gerais da nota, na Figura 1, seriam atribuídas ao objeto da nota em si. Dessa forma uma nota carregaria uma lista de produtos, seria agrupada por uma categoria, e traria as demais informações de interesse ao projeto. A estrutura desses três objetos foi planejada como na Figura 12.

Figura 12 – Estrutura dos objetos principais do projeto.



Fonte: Autor.

4.3.2 Requisitos Básicos

Com os requisitos básicos da aplicação pautados na fase de desenvolvimento anterior, e o planejamento de sua estrutura básica já especificado, iniciou-se uma fase de implementação. Ela objetiva colocar em prática e em funcionamento as necessidades básicas da aplicação. Para auxílio no projeto e implementação do trabalho, foram obtidas notas fiscais eletrônicas pertencentes a uma SPE da construção civil, as quais foram disponibilizadas para o específico desenvolvimento deste projeto. Com elas foi possível ter consciência das movimentações reais e comuns em um projeto, bem como seus produtos e valores.

Para isso, inicialmente foi implementado no banco de dados os modelos para os objetos principais da aplicação. No caso do MongoDb, são criados *schemas* para os documentos que representariam esses objetos nas suas coleções. Na sequência, com o banco de dados capaz de armazenar os objetos nos padrões escolhidos, seguiu-se para a implementação no servidor Node.js. No servidor Node.js foram criadas as rotas pertinentes ao

CRUD de categorias, inserção e remoção de notas, bem como edição das categorias de um produto. O conceito de *routers* do Express, como previamente analisado, foi base para a quebra desses serviços individuais em tarefas menores. Um *router* seria responsável por lidar com os serviços de notas, enquanto um segundo trataria de categorias, e por fim um terceiro das operações referentes a produtos. Essa capacidade de isolar serviços permitiu a flexibilidade de alterar a implementação e realizar manutenção em um *router* sem afetar outro.

Nesse momento de implementação, é de importância o destaque de duas regras de negócios utilizadas durante essa etapa. A primeira envolve o envio das notas e seu armazenamento no banco de dados. As notas ao serem recebidas do *front-end* em seu formato em XML seriam mapeadas pelo servidor Node.js para o formato JSON. Dessa forma as notas em XML nunca chegariam de fato ao banco de dados, mas sim seus campos de interesse em um objeto JSON. Com o ecossistema da aplicação fundamentalmente baseado em JavaScript, isso evitaria a necessidade de mapeamentos futuros a esse objeto, dado que a informação seria facilmente manipulável por toda a aplicação. A segunda envolve a remoção de categorias do sistema. Como as notas fiscais dependiam das categorias para se relacionarem entre si, serem catalogadas e agrupadas, uma categoria só poderia ser removida caso não possuísse nenhuma nota fiscal vinculada a ela.

Por fim, ainda nessa fase inicial de implementação, foram criadas no *front-end* as páginas para o usuário interagir com o sistema, incluindo notas e cadastrando categorias. A arquitetura dos componentes do React foi de extrema importância para uma interface de usuário leve e modular. As páginas poderiam ter seu conteúdo quebrado em componentes isolados e reutilizáveis ao longo do desenvolvimento, como a tabela de notas fiscais que é reutilizada no dashboard. Isso evita duplicação de código e traz facilidade e simplicidade do desenvolvimento

4.3.3 Relatórios

Tendo a possibilidade do usuário interagir com o sistema, inserindo notas fiscais e as categorizando com base em sua preferência, foi possível planejar e executar o desenvolvimento dos relatórios. A proposta dos relatórios era trazer informativos que trouxessem apoio a tomada de decisão. Para essa aplicação, se tratando de SPEs da construção civil, relatórios acerca de despesas através da vida do projeto se mostraram de interesse e grande valia. Dessa forma, um administrador do projeto, bem como seus demais integrantes, teriam o controle os gastos dos projetos por categoria.

É de destaque a escolha de não realizar relatórios específicos por produtos, mas por áreas orçamentárias e as notas as quais estão relacionadas. Durante o planejamento da estrutura dos relatórios, foi encontrada uma quantidade elevada de produtos distintos

nas notas fiscais obtidas. Com essa alta quantidade de produtos distintos nas notas, seria inviável uma interface de usuário em que o mesmo selecionava dentre os produtos existentes o que fosse de seu interesse. Com isso optou-se por realizar os relatórios com base nas áreas orçamentárias.

Dessa forma, foram planejados relatórios que mostrassem as despesas da SPE ao longo de seu tempo de vida. Os relatórios usariam as datas das notas das áreas orçamentárias, bem como o valor de seus produtos, para trazer os gastos de projeto. Além disso, optou-se por não armazenar informações referentes aos relatórios no banco de dados. Ao armazenar os relatórios no banco de dados, a cada nota inserida, o relatório precisaria ser refeito e atualizado no banco de dados. Nessa mesma linha, ao atualizar a área orçamentária de um produto, ou remover uma nota, novamente o relatório precisaria ser refeito. Dessa forma, decidiu-se por deixar a geração do relatório em tempo de execução com a solicitação do usuário.

Foram planejados e implementados três relatórios distintos que mostram as despesas das áreas orçamentárias através do tempo. Os três relatórios são um diário, um mensal, e um anual. O trio inicial de relatórios funciona de maneira semelhante. Todos enviam sua requisição para o servidor levando duas datas que formam um intervalo. Para o relatório diário ambas são iguais, para o mensal as datas englobam o início e o fim de um mês, e de forma semelhante o relatório anual recebe as datas iniciais e finais de um ano.

Com isso em mente, para cada intervalo de data obtido, o servidor solicitava ao banco de dados todos os produtos das notas que possuíam data de emissão presente naquele intervalo. Agrupando os produtos por sua área orçamentária, foi possível descobrir e devolver para o usuário o valor total gasto por cada área orçamentária. Esse objeto de retorno continha os dados que formariam o relatório.

Há uma pequena, mas importante diferença, entre os relatórios diários e mensais quando comparados aos relatórios anuais. Como escolha de projeto foi tomada a decisão de, além dos dados que informam os gastos anuais pertencentes às áreas orçamentárias, trazer também os valores de orçamentos escolhidos para aquelas áreas. Os baixos valores diários e mensais movimentados quando comparados aos altos valores orçamentários, traziam problemas estéticos e dificuldade de leitura e interpretação dos gráficos, ao passo que os valores de interesse ficavam à sombra dos grandes valores orçamentários. Enquanto isso, ao comparar com os valores anuais, a construção do gráfico se mostrou esteticamente melhor e com melhor legibilidade.

4.3.4 Melhorias

A aplicação possuindo seus requisitos básicos de desenvolvimento, bem como a geração de relatórios para o apoio à tomada de decisão, iniciou-se uma fase de planejamento e implementação de melhorias. O objetivo da fase era inicialmente encontrar pontos nos desenvolvimentos anteriores os quais poderiam ser melhorados, bem como novos recursos para a aplicação.

A primeira melhoria seria a modelagem de outros gráficos e informativos importantes para a aplicação, bem como uma forma sucinta de visualização que provesse um panorama geral do estado financeiro da SPE. Com essa melhoria em mente, foi planejado um *dashboard* com métricas importantes para o usuário e outros gráficos informativos. Para seu desenvolvimento, no *front-end* foi criada uma página específica para o *dashboard*. A página continha métricas de despesa como, por exemplo, orçamento total e total de despesas registradas, como novos gráficos informativos. Em si também possuía uma versão resumida da tabela de notas fiscais. Um *router* específico foi criado no servidor para o gerenciamento das rotas da página, que continha consultas que percorriam várias coleções do MongoBD.

A segunda melhoria envolvia poder atribuir *tags* às notas que denotavam seu pagamento. Buscava-se uma forma de classificar as notas como pagas ou não, a fim de trazer informações mais precisas ao usuário do sistema a fim de informar-se das finanças do projeto. Com isso, foi implementada uma nova rota no sistema que permitia enviar arquivos ao sistema e atribuí-los às notas fiscais. O objetivo era utilizar arquivos OFX, que registram movimentações financeiras, para denotar notas já pagas. OFX é a sigla para Open Financial Exchange (Intercâmbio Financeiro Aberto), um *framework* relacionado a XML para a troca de informações financeiras entre instituiçõs e clientes via Internet (ZHAO; SANDAHL, 2001). Dessa forma, a rota recebia um arquivo do cliente e o índex da nota alvo, armazenava-o no sistema e criava o vínculo entre os diferentes documentos. Isso permitia o usuário, à necessidade, recuperar o arquivo alvo interagindo com a nota especificada. Além disso, os informativos de notas já pagas - ou seja, com arquivos atribuídos - eram de importância para a exposição de métricas financeiras ao usuário.

4.3.5 Implementações Futuras

Ao final da fase de implementação de melhorias, a implementação geral da aplicação estava concluída. Seus requisitos básicos previamente estavam modelados e concretos. O usuário era capaz de interagir com o sistema e obter informações acerca de suas movimentações financeiras com gráficos e informativos. Melhorias tinham sido levantadas e agregadas ao sistema de forma modular e reutilizável. Tendo isso como base, foi possível criar e executar cenários de uso de um usuário e planejar futuras melhorias e

implementações para o sistema.

Inicialmente como futura implementação planejada, surgiu a ideia de um sistema de autenticação. Um recurso de autenticação possibilitaria não só o acesso somente às credenciais cadastradas e autorizadas, bem como a modelagem de funções baseadas em permissões de usuário. Esse recurso traria a capacidade de limitar ações dos usuários baseados em seus cargos e permissões obtidas. Com isso, seria possível a criação de um cargo de leitor, onde um usuário com esse cargo seria capaz apenas de consumir informações da interface, sem interagir com os recursos desenvolvidos. Um segundo cargo possível, por exemplo, seria um cargo capaz de alimentar o sistema com informações. Ou seja, adicionar notas, editar categorias de produtos, atribuir pagamentos às notas. Esse cargo interagiria mais profundamente com o sistema e possuiria mais recursos à sua disposição. Por fim, caberia um cargo de administrador que, por sua vez, poderia manipular as permissões de outros usuários e controlar totalmente o sistema.

Com essa implementação, seria possível trazer maior confiabilidade ao sistema. É de destaque a importância dessa implementação futura planejada ser implementada antes de demais funcionalidades, ao passo em que ela gerenciaria o acesso às novas e antigas funcionalidades. Dessa forma evitaria assim o retrabalho em códigos antigos e facilidade no desenvolvimento.

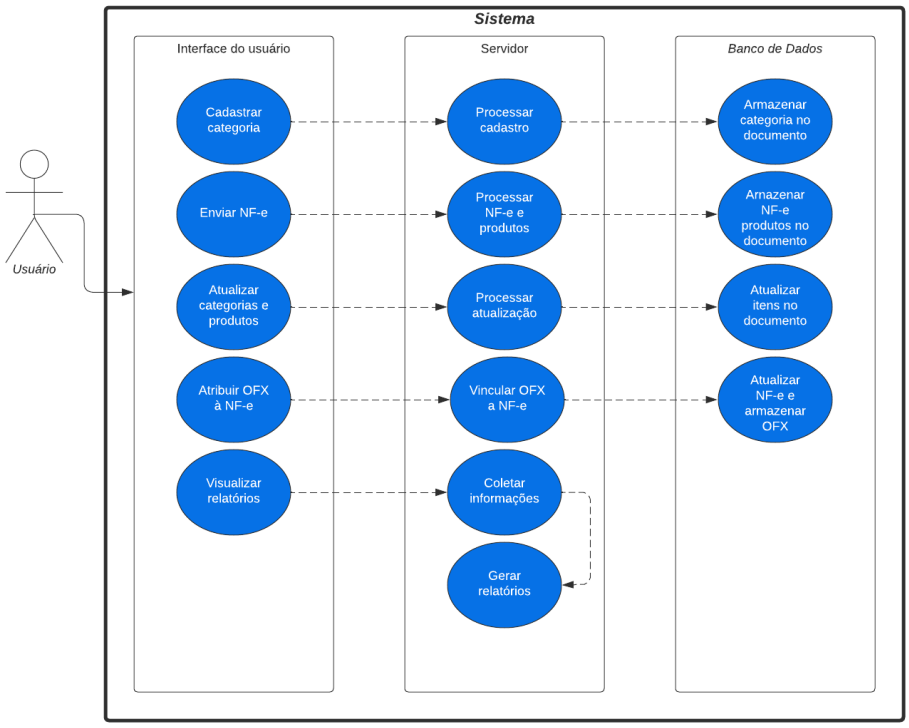
Outra implementação futura planejada seria a paginação das requisições das notas na ponta do servidor. Por mais que haja um alto nível de performance e desempenho das tecnologias utilizadas para a aplicação, é necessária a busca pela otimização do sistema. No estado atual do sistema, o usuário solicita as notas ao servidor e toda são repassadas de forma simultânea. Ao passo em que a aplicação cresce o volume de notas a acompanha, as respostas às requisições tendem a ficar mais lentas ao buscar todas as notas do sistema. Mesmo o sistema estando preparado e implementado para tratar a sobrecarga, é possível preveni-la.

Uma paginação na ponta do servidor ao usuário requisitar as notas traria para ele somente a informação que precisaria naquele momento. Dessa forma, o usuário só possuiria acesso às notas que estariam na página da tabela que estaria visualizando, por exemplo. Ao trocar de página, uma nova requisição seria feita para as notas da página seguinte e assim por diante. Com isso, uma sobrecarga ao sistema seria evitada e uma menor latência nas respostas seria obtida.

5 CENÁRIOS

A fim de demonstrar os resultados obtidos após o desenvolvimento e implementação do sistema projetado, foram criados cenários de execução do ponto de vista do usuário. Esses cenários trazem em maior detalhe, de forma visual, as etapas encontradas no capítulo de metodologia do projeto. Seu objetivo é demonstrar como o usuário comum da plataforma utilizaria de seus recursos para obter a proposta objetivada no trabalho. Com isso cada cenário traz suas funcionalidades e o capítulo finaliza com um cenário o qual demonstra os relatórios da plataforma. Assim, com esses cenários e resultados, é possível o obter o desejado apoio à tomada de decisão em uma SPE da construção civil. Na Figura 13, o diagrama de casos de uso identifica as principais operações que o usuário pode ralizar no sistema, mostrando sua sequência de execução pela plataforma.

Figura 13 – Diagrama de casos de uso da aplicação.



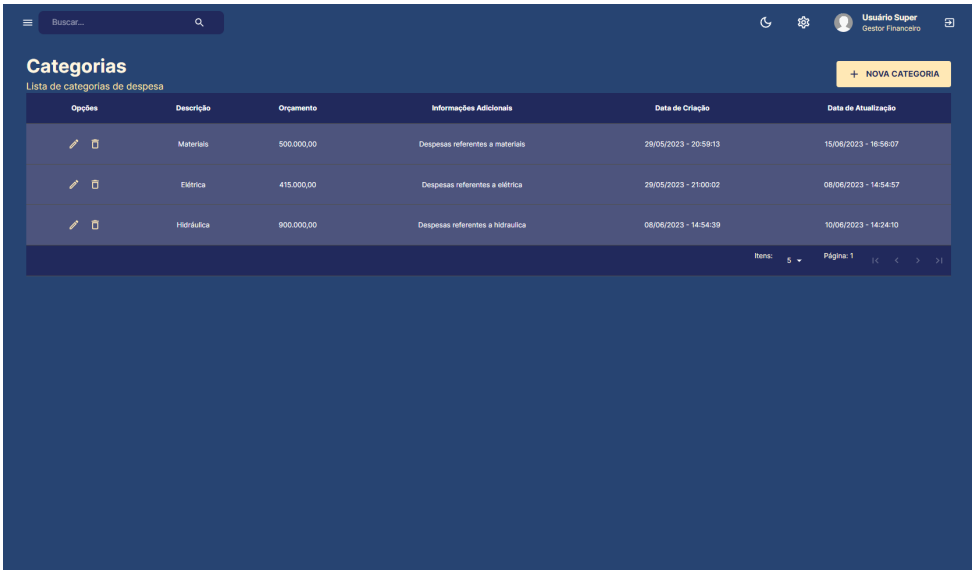
Fonte: Autor.

5.1 CENÁRIO 1 - CRIAÇÃO DE ÁREAS ORÇAMENTÁRIAS

O primeiro cenário de análise demonstra o funcionamento da primeira funcionalidade básica da aplicação. Nele é possível visualizar como funciona a interação do usuário

com a aplicação a fim de coletar, criar, editar e remover as áreas orçamentárias. A Figura 14 exibe de forma geral a página dedicada à funcionalidade. As áreas orçamentárias cadastradas pelo usuário têm seu destaque na tabela que preenche a maior parte da página. A criação de notas fiscais se dá pelo botão indicado a este fim. Enquanto isso, para edição e remoção de uma categoria específica, utiliza-se da coluna de opções com os botões para tal presentes na linha que representa aquela categoria.

Figura 14 – Página de áreas orçamentárias.



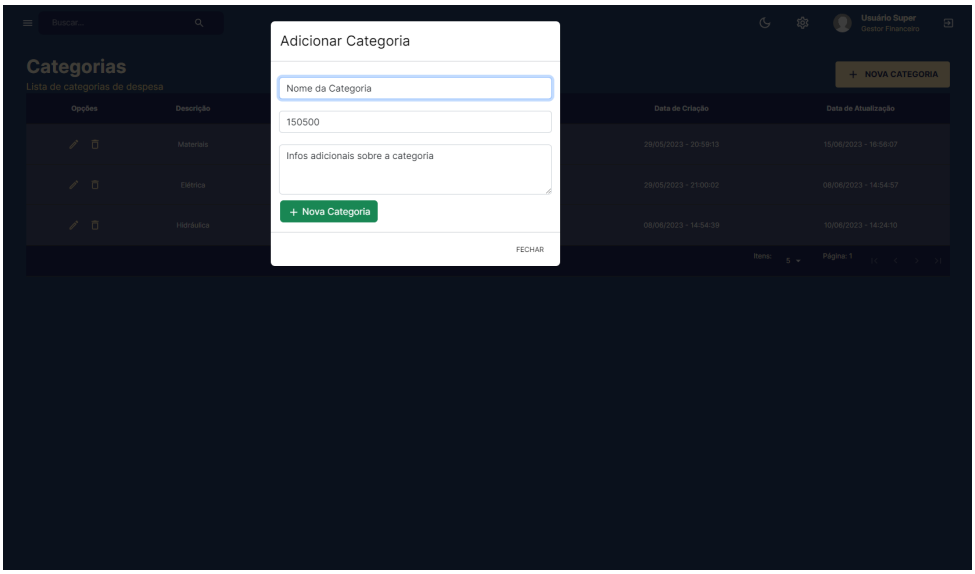
Fonte: Autor.

Interagindo com o botão de criação de áreas orçamentárias abre uma janela - vista na Figura 15 - sobre a aplicação, janela que contém o formulário para criação desse item. Dessa forma, o usuário insere as informações pertinentes ao cadastro daquela categoria. Seu nome, orçamento e informações gerais. Ao enviar, o objeto gerado pelo formulário e pela requisição ao chegar no servidor não precisa ser mapeado para ser armazenado no banco de dados. Com MongoDB já recebendo objetos JSON e com o formulário estando no mesmo modelo configurado anteriormente, a categoria é logo armazenada. Com o sucesso do cadastro, uma requisição é imediatamente feita pelo *front-end* para obter a lista atualizada de categorias com seu mais novo cadastro.

5.2 CENÁRIO 2 - INCLUSÃO DE UMA NF-E

O segundo cenário a ser analisado exibe outra funcionalidade básica da aplicação. Nesse cenário é demonstrado como o usuário insere as notas fiscais na aplicação. Dos requisitos básicos esse se destaca como o de maior importância, visto que sem as notas fiscais não existiriam informações para a geração dos relatórios. De forma como as áreas

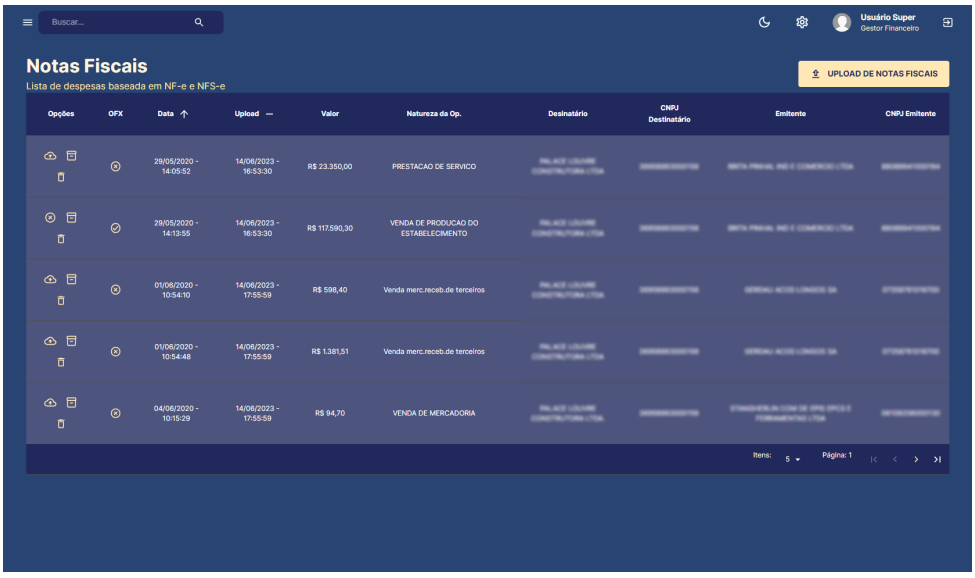
Figura 15 – Formulário de cadastro de áreas orçamentárias.



Fonte: Autor.

orçamentárias, foi criada uma tabela que contém as principais informações acerca dos arquivos enviados. Na Figura 16 há a exibição da página criada para a interação do usuário com as notas fiscais. Nessa mesma figura, a fim de resguardar informações das partes envolvidas nas reais transações financeiras, foram borrados os campos de destinatário e emitente bem como seus respectivos CNPJs. A forma pela qual o usuário insere notas fiscais no sistema é interagindo com o botão acima da tabela. E para a remoção de uma nota, de forma semelhante à tabela de categorias, o usuário interage com o botão de remoção na coluna de opções que está presente na linha pertinente àquela nota.

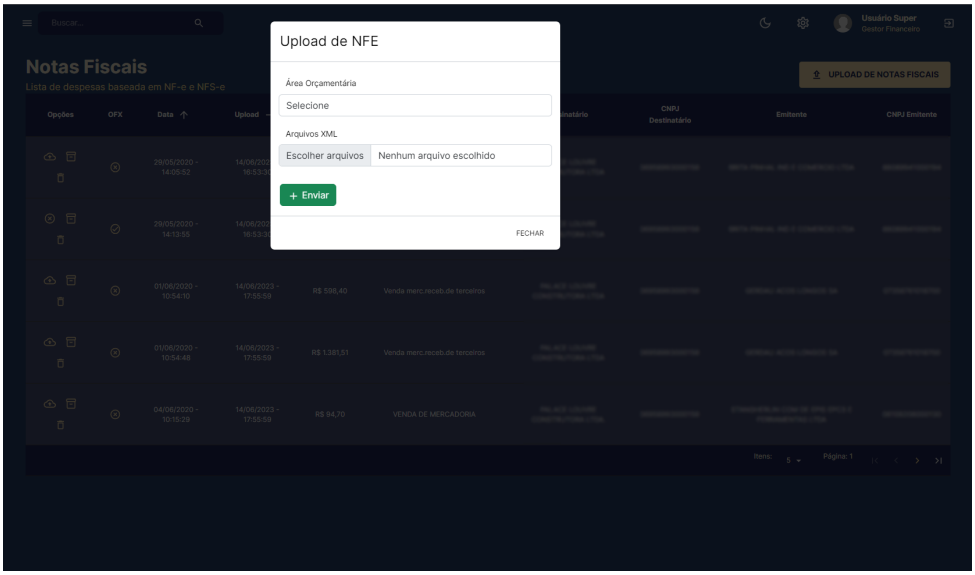
Figura 16 – Página de notas fiscais.



Fonte: Autor.

Desta interação surge o formulário em janela flutuante, visto na Figura 17, para o envio das notas ao sistema. A inserção de NF-e foi projetada para ser de fácil utilização e intuitiva ao usuário. Ao inserir uma nota, o usuário precisa escolher uma área orçamentária pertinente àquela nota afins de categorização como visto em seções anteriores. Ainda nisso, o usuário também insere um ou mais arquivos no formato XML (único formato aceito) para serem armazenados pelo sistema. O envio de múltiplas notas em uma única requisição permite ao usuário a rápia categorização das mesmas dentro das áreas orçamentárias selecionadas. De forma como foi analisado na Seção 4.3.2 desse texto, o arquivo de nota não é em si armazenado no banco de dados, mas sim suas informações em um formato JSON.

Figura 17 – Formulário de envio de NF-es.



Fonte: Autor.

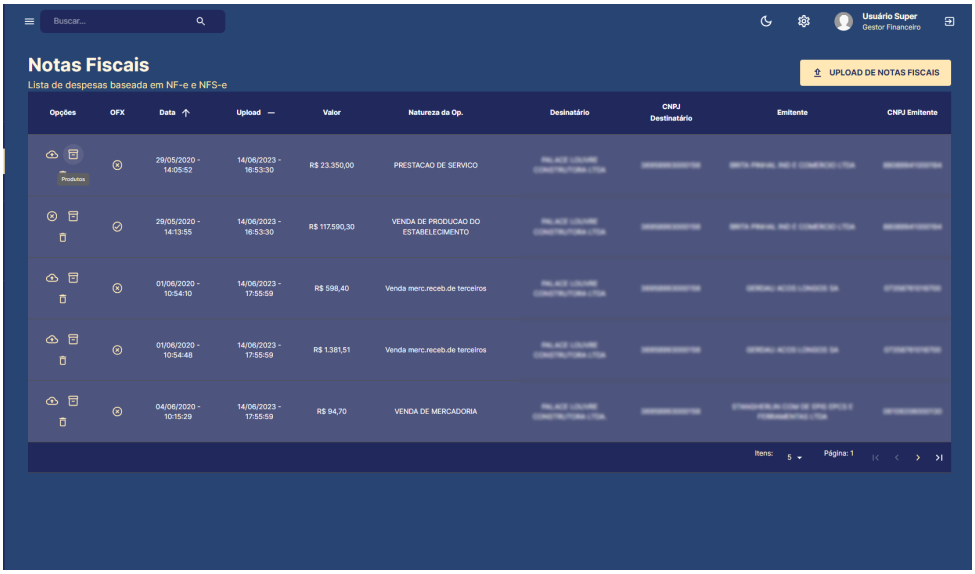
5.3 CENÁRIO 3 - EDIÇÃO DA ÁREA ORÇAMENTÁRIA DE UM PRODUTO

O terceiro cenário de análise demonstra a última das necessidades básicas da aplicação. Como visto no cenário anterior, é possível enviar uma ou mais notas para serem armazenadas no sistema e agrupadas conforme suas categorias. Também foi analisado nesse trabalho a capacidade de uma nota fiscal de possuir mais de um produto em seu registro. Esse cenário abrange a funcionalidade da edição da área orçamentária de um ou mais produtos de uma nota, requisito básico levantado no início do desenvolvimento.

Para isso, o usuário pode permanecer na mesma tela de notas fiscais vistas na Figura 16. Ao interagir com o ícone de produtos da nota, como indicado na Figura 18, o usuário tem acesso a uma nova janela suspensa. Essa janela provê informações acerca

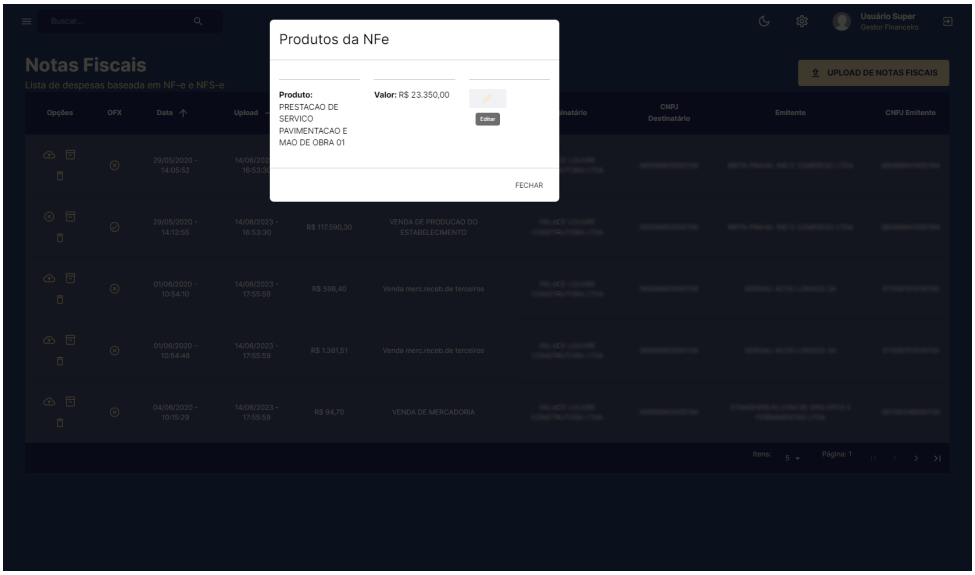
dos produtos daquela nota, como seu nome e valor total. Além disso, nessa janela, há um botão, visto na Figura 19 para interação do usuário para realizar a alteração da área orçamentária daquele produto. Isso por sua vez, abre uma segunda janela suspensa com a opção do usuário selecionar a área orçamentária desejada para aquele produto, como mostra a Figura 20. Com isso, o usuário pode individualmente alterar cada produto da nota à sua vontade e necessidade.

Figura 18 – Interação para visualização dos produtos de uma nota.



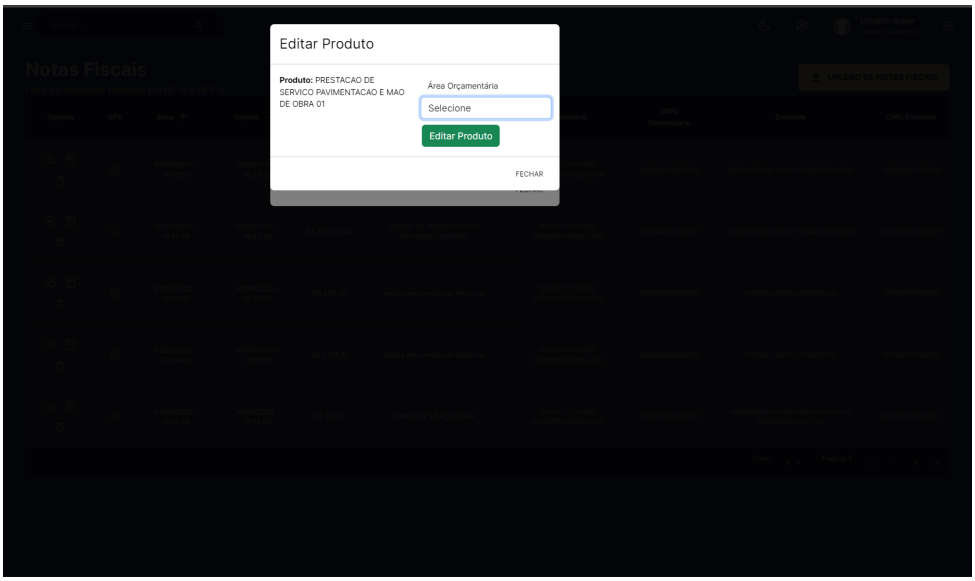
Fonte: Autor.

Figura 19 – Visualização dos produtos de uma nota para edição de sua área orçamentária.



Fonte: Autor.

Figura 20 – Formulário de edição de área orçamentária de um produto.



Fonte: Autor.

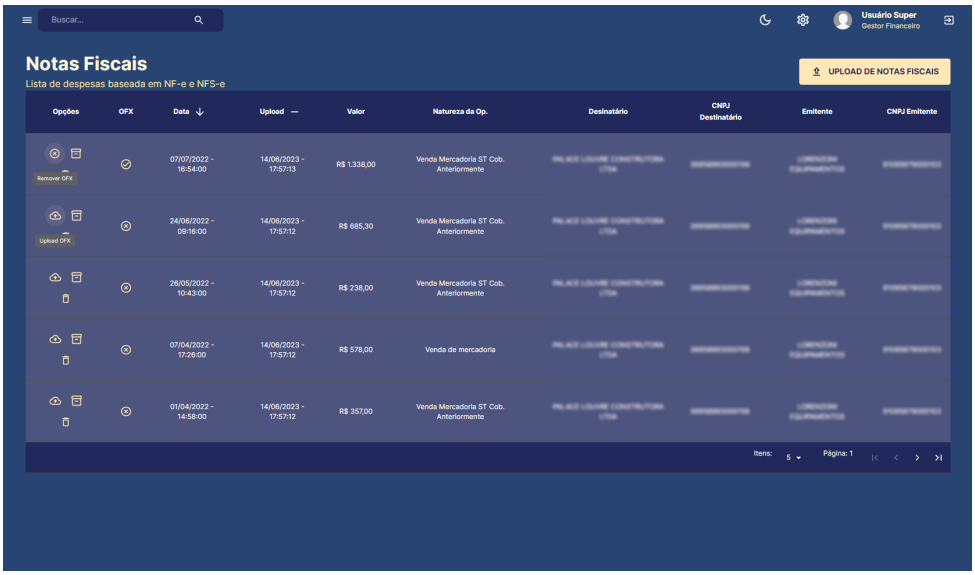
5.4 CENÁRIO 4 - ADIÇÃO DE UM OFX A UMA NF-E

Na sequência, é explorado o cenário que demonstra como o usuário pode atribuir um arquivo OFX ao pagamento de uma nota. Esse cenário exemplifica uma das primeiras melhorias desenvolvidas para o sistema. Com ela o usuário é capaz de enviar um arquivo ao servidor e criar um vínculo entre ele e uma nota fiscal. Dessa forma é possível classificar as notas em pagas e não pagas com base na existência ou não de arquivos vinculados a elas.

Para isso, o usuário pode se manter na tela dedicada para notas fiscais, mostrada na Figura 16. Os recursos refentes à manipulação de notas foram agrupados nessa página a fim de trazer maior agilidade ao usuário ao interagir com o sistema. Da mesma forma em que edita os produtos de uma nota, na coluna opções para a nota existe um botão em que o usuário atribui um arquivo à nota. Isso é melhor visto na Figura 21. Quando uma nota já possui um arquivo vinculado, a interface de usuário reconhece essa propriedade e altera a funcionalidade do botão. Essa alteração permite, com a interação do usuário, a remoção do arquivo vinculado à nota.

Com isso, para adicionar um arquivo a uma nota cabe ao usuário atribuir um único arquivo à janela flutuante exibida, como mostrado na Figura 22. Dessa forma o arquivo será redirecionado ao servidor onde será armazenado no formato BSON - Binary JSON, JSON Binário - no banco de dados. Isso mostra outra facilidade dada à escolha de linguagens e tecnologias do sistema. O arquivo precisa somente ser encaminhado ao banco onde será armazenado, não necessitando de mapeamentos ou compressões. Arquivos leves, como os utilizados na aplicação podem ser facilmente manuseados sem auxílio de bibliotecas e *frameworks* auxiliares.

Figura 21 – Forma de interação do usuário com o sistema de arquivos.



Fonte: Autor.

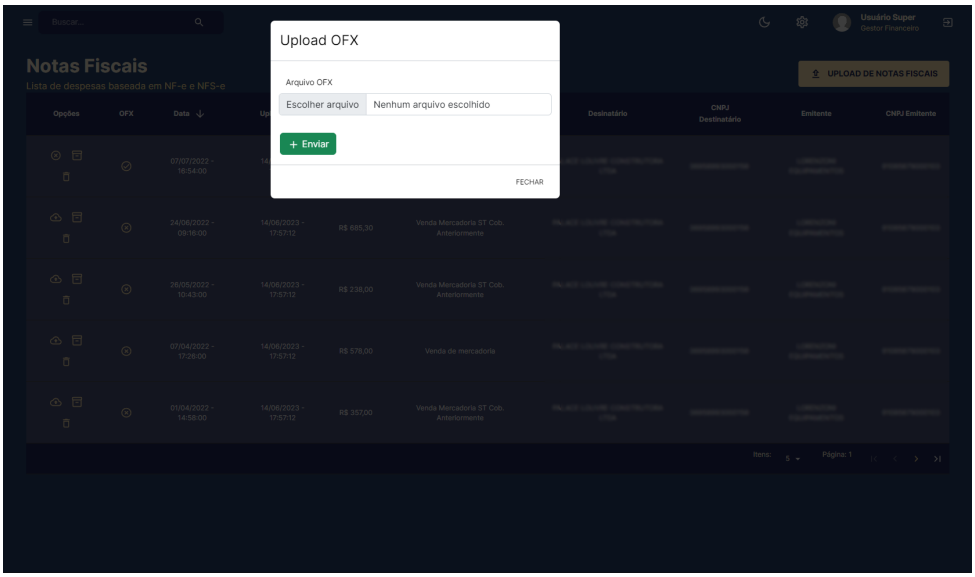
5.5 CENÁRIO 5 - VISUALIZAÇÃO DE RELATÓRIOS

Este último cenário demonstra os gráficos e informativos expostos ao usuário e com base nos parâmetros de sua escolha. Como descrito na Seção 4.3.3, inicialmente foram implementados três gráficos para o usuário que mostrassem suas despesas através do tempo. Ao usuário selecionar seus parâmetros de interesse, uma requisição é disparada para o servidor Node.js que coleta do banco de dados MongoDB as informações necessárias e as organiza de forma informativa para o usuário. Dessa forma é possível ter maior controle financeiro do projeto e trazer o apoio à tomada de decisão.

Os relatórios diário, mensal e anual podem ser vistos em maior detalhe respectivamente nos Gráficos 1, 2 e 3. Destaca-se aqui a presença dos orçamentos de cada área junto com suas despesas nos gráficos anuais. Dessa forma o usuário pode ter ainda mais controle e informativos acerca de seu projeto e realizar ajustes de orçamentos caso necessários.

Nesse cenário também é possível visualizar o *dashboard* desenvolvido ao usuário na fase de melhorias. O objetivo do *dashboard* é trazer de forma rápida e concisa informações-chave para o usuário. Dessa forma, vê-se na Figura 23 a forma na qual a implementação foi resultada. Existem quatro *cards* que trazem valores de importância ao usuário, bem como dois gráficos informativos de despesas e uma versão resumida da tabela de notas fiscais. Os *cards* trazem valores como o total orçamentado para o projeto e o total gasto, bem como quanto desse valor gasto está proporcional ao valor orçamentado. Além disso, há neles a informação do valor pago em comparação ao registrado no sistema, e também o valor de ISS retido. O ISS é uma tributação que deve ser paga por empresas e profissionais que prestam serviços e emitem nota fiscal. Na prática, ele cor-

Figura 22 – Inserção de arquivo vinculado a uma NF-e.



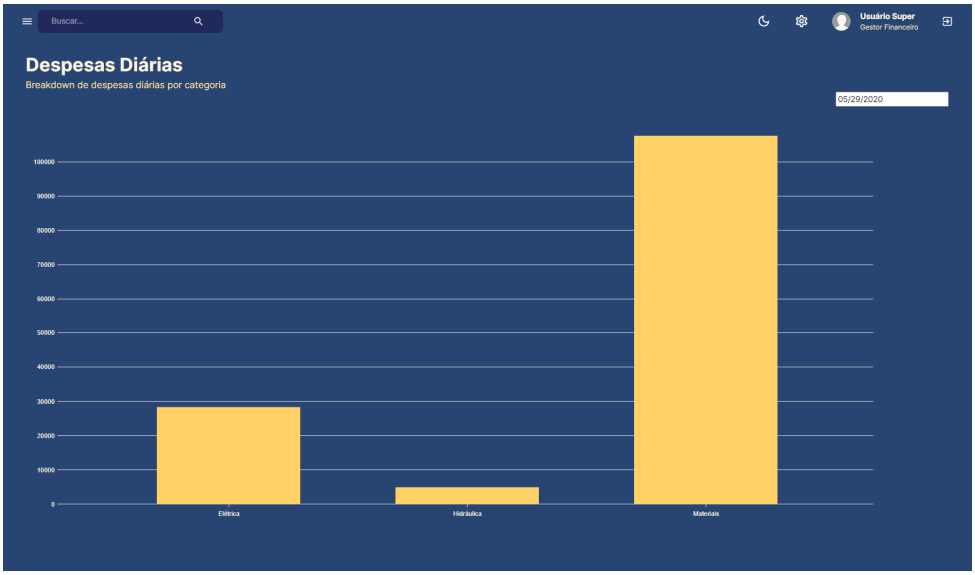
Fonte: Autor.

responde à obrigação que o tomador de serviço tem de reter o valor equivalente ao ISS que o prestador deve no momento de pagamento do serviço.

Figura 23 – *Dashboard* da aplicação.

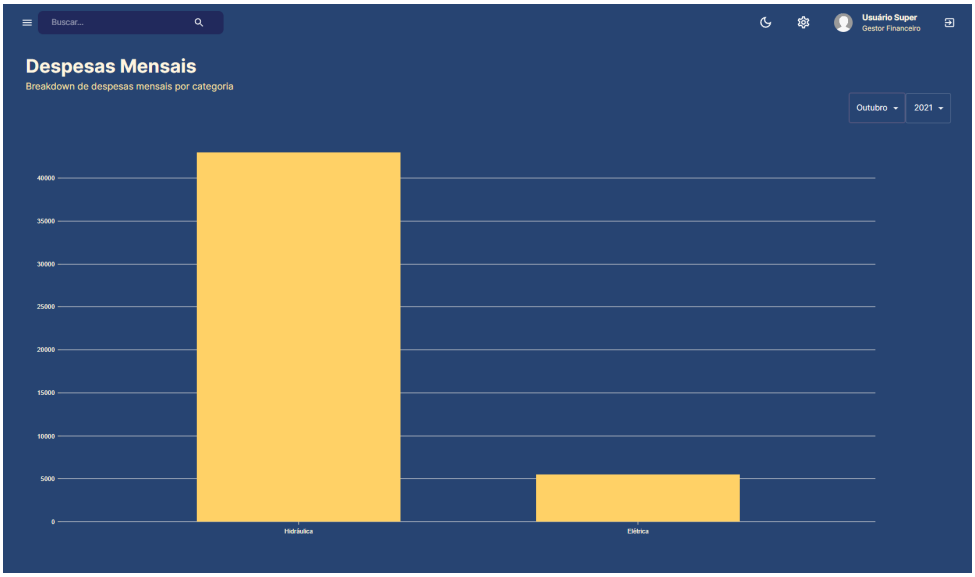
Fonte: Autor.

Gráfico 1 – Relatório diário de despesas de uma SPE.



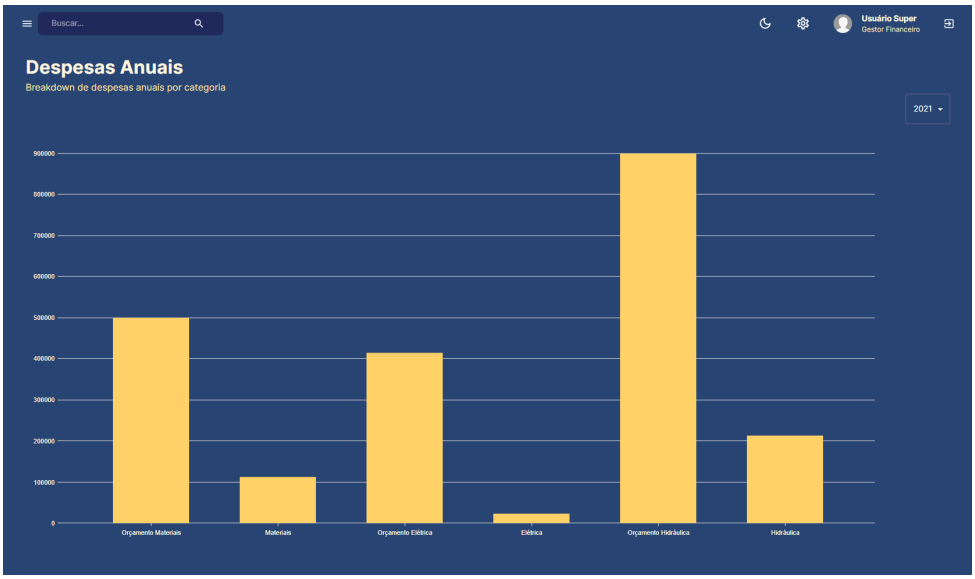
Fonte: Autor.

Gráfico 2 – Relatório mensal de despesas de uma SPE.



Fonte: Autor.

Gráfico 3 – Relatório anual de despesas de uma SPE.



Fonte: Autor.

6 CONCLUSÃO

Este trabalho apresentou a implementação da proposta de um sistema de apoio à decisão para SPEs na construção civil. Foi abordado o referencial teórico e bibliográfico que embasa o trabalho, bem como a metodologia para sua construção. Finalmente, foram realizados cenários que exibissem os resultados obtidos no desenvolvimento da aplicação e as interações do usuário com o sistema desenvolvido.

Inicialmente, o fato que pode ser analisado foi a escolha de tecnologias para a implementação da aplicação desenvolvida. O objetivo dessa etapa era de encontrar o conjunto de ferramentas mais pertinentes ao desenvolvimento, com isso trazendo maior desempenho e qualidade de vida à aplicação. Foi realizado o estudo das principais tecnologias existentes no cenário de desenvolvimento de aplicações web, se aprofundando em suas vantagens e desvantagens. Dessa forma, foram encontrados recursos para o desenvolvimento da aplicação que se destacavam pela sua alta performance e desempenho, de forma que juntas criavam um ecossistema inteiro composto por JavaScript.

A aplicação estando totalmente baseada em JavaScript trouxe apenas benefícios ao desenvolvimento. Ao iniciar uma lógica de desenvolvimento em uma ponta da aplicação, ela era capaz de ser continuada e estendida em qualquer outra parte da mesma aplicação sem necessidade de transformações de dados ou adaptações de lógica. Isso trouxe não só maior facilidade na implementação, mas coesão e solidez. A escolha de ferramentas para *front-end* se mostrou certeira e de extrema importância. Seu destaque se dá principalmente pela estrutura da biblioteca *React* em organizar seus itens em componentes reutilizáveis. Dessa forma foi possível criar componentes específicos para tabelas ou relatórios e utilizá-los com facilidade em outras partes do sistema. Isso evitou duplicação desnecessária de código e, com isso, dificuldades em futuras manutenções ao sistema.

De mesma forma, para o *back-end* da aplicação, o par de servidor Node.js e banco de dados MongoDB supriu todas as necessidades planejadas da aplicação. Em um sistema com interação frequente e em alta quantidade do usuário com o servidor, o sistema de *routers* para gerenciamento das requisições teve seu destaque. Foi possível dividir as rotas com base em suas funcionalidades e as isolando entre si. Ao dar manutenção ou desenvolver um novo serviço, os demais implementados não eram afetados. Além disso, a escalabilidade da aplicação se mostrou extremamente elevada por esse fator, dado que facilmente uma funcionalidade pode ser adicionada. Na mesma linha, o banco de dados escolhido foi de fácil agregação ao sistema e de simples manuseio. Os dados podiam ser diretamente recebidos do *front-end* e inseridos no banco sem necessidade de tratamento prévio. Isso possibilitou trabalhar numa arquitetura de dados universal na aplicação.

O planejamento preemptivo, bem como a organização da implementação do projeto em fases sequenciais, garantiu a sua construção sólida e consistente. O tempo dedicado

ao levantamento de requisitos iniciais da aplicação foi essencial para separar esses de melhorias e funcionalidades não necessárias naquela fase. Também permitiu a construção planejada e bem arquitetada desses requisitos básicos e seus vínculos, vide que sua implementação fora prototipada e planejada desde o princípio. Além disso, a modularidade da plataforma se destaca quando comparada a outras soluções encontradas. A facilidade na qual é possível inserir novos módulos, bem como sua construção e implementação focadas no maior desempenho possível, agregam valor à implementação desenvolvida.

Como objetivo, o desenvolvimento da aplicação buscava trazer apoio à tomada de decisão para SPEs da construção civil. Esse apoio viria na forma de gráficos e relatórios que trouxessem métricas e informações das movimentações financeiras realizadas durante sua existência. Foram desenvolvidos cenários que, utilizando as funcionalidades implementadas, simulam interações do usuário e como este pode utilizar o sistema. Esses cenários percorrem desde funcionalidades básicas da aplicação como suas melhorias. O destaque dentre esses cenários é para o cenário 5, na Seção 5.5, que ilustra como o usuário pode visualizar as métricas geradas pela aplicação.

Essas métricas, vindo na forma de gráficos, tabelas e *cards*, trazem informações valiosas ao usuário. Com tais métricas o usuário do sistema pode analisar suas despesas através do tempo de vida do projeto. De mesma forma, o usuário tem a possibilidade de analisar e controlar tais despesas por categorias cadastradas por si. Com isso, é possível afirmar que o sistema cumpre seu objetivo de trazer o apoio à tomada de decisão. A coletânea de informações que traz serve como respaldo para um administrador de uma SPE ao tomar uma decisão financeira

No que tange trabalhos futuros, a Seção 4.3.5 adentra no tópico de implementações futuras do projeto, trazendo possíveis novas funcionalidades e recursos. É de certeza que, como um sistema de autenticação e paginação de dados já existem, há espaço para o planejamento e desenvolvimento de outras novas funcionalidades ao sistema. Além disso, um importante desenvolvimento futuro para a aplicação é hospedá-la em uma plataforma. Dessa forma, com a aplicação hospedada e combinada com um sistema de autenticação, seria possível testar ainda mais seu desempenho com o acesso de múltiplos usuários realizando múltiplas requisições. Da forma na qual foi construído, o sistema existe facilmente capaz de agregar a si novas funcionalidades.

Conforme o tempo de vida de uma SPE se estende e mais usuários utilizam o sistema, mais necessidades de novas métricas e relatórios surgem. A construção modular da aplicação traz a capacidade supracitada de agregar a si novas funcionalidades e com a adição delas ainda garantir a integridade dos serviços previamente desenvolvidos. Campos básicos como preços e quantidades, bem como mais complexos como impostos e identificadores, foram utilizados ao longo desse trabalho para geração de métricas de interesse ao usuário. É importante destacar como ainda existem múltiplos campos na estrutura da NF-e e NFS-e que podem também trazer o apoio buscado com o desenvolvimento da aplicação.

Com a busca por novos valores de interesse para análise, bem como o desenvolvimento de novas funcionalidades ao sistema surge também a necessidade de aprofundar ainda mais o estudo na estrutura das notas e sua organização e em planejamentos futuros para a plataforma.

REFERÊNCIAS

AGGARWAL, S. et al. Modern web-development using ReactJs. **International Journal of Recent Research Aspects**, v. 5, n. 1, p. 133–137, 2018.

ALVES, F. R.; FREITAS, L. M. d. Análise da administração de obras por sociedade de propósito específico. **Engenharia Civil-Pedra Branca**, 2020.

BANGARE, S. et al. Using Node.Js to build high speed and scalable backend database server. **International Journal of Research in Advent Technology**, v. 4, p. 19, 2016.

BATISTA, A. K. P.; MACHADO, L. de S. Planejamento tributário em empresa da construção civil: o caso das sociedades de propósito específico (SPEs). **ConTexto-Contabilidade em Texto**, v. 21, n. 49, p. 57–72, 2021.

BOICEA, A.; RADULESCU, F.; AGAPIN, L. I. MongoDB vs Oracle–database comparison. In: IEEE. **2012 Third International Conference on Emerging Intelligent Data and Web Technologies**. Bucharest, Romania, 2012. p. 330–335.

BRANCO, L. d. A. Nota fiscal eletrônica e SPED: aspectos práticos e implicações tributárias, 2008. **Jus Navegandi. Disponível em: <http://jus.com.br/revista/texto/10920/nota-fiscal-eletronica-e-sped-aspectos-praticos-e-implicacoes-tributarias>. Acesso em 14/06/2023**, v. 25, 2012.

BRAY, T. **The JavaScript Object Notation (JSON) Data Interchange Format**. RFC Editor, 2014. RFC 7159. (Request for Comments, 7159). Disponível em: <https://www.rfc-editor-.org/info/rfc7159>.

BROWN, E. **Web development with node and express: leveraging the JavaScript stack**. Sebastopol, CA: O'Reilly Media, 2019.

CONSORTIUM, W. W. W. et al. Extensible markup language (XML) 1.1. World Wide Web Consortium, 2006.

FIELDING, R. T. **Architectural styles and the design of network-based software architectures**. California: University of California, Irvine, 2000.

FRĂTEAN, T. Bazele de date NoSQL–o analiză comparativă. **To Day Software Magazine**, n. 10.

GERON, C. M. S. et al. SPED–sistema público de escrituração digital: percepção dos contribuintes em relação os impactos de sua adoção. **Revista de Educação e Pesquisa em Contabilidade (REPeC)**, v. 5, n. 2, p. 44–67, 2011.

GYŐRÖDI, C. et al. A comparative study: MongoDB vs. MySQL. In: IEEE. **2015 13th International Conference on Engineering of Modern Electric Systems (EMES)**. Oradea, Romania, 2015. p. 1–6.

HAHN, E. **Express in Action: Writing, building, and testing Node.Js applications**. Shelter Island, NY: Simon and Schuster, 2016.

JOSE, B.; ABRAHAM, S. Performance analysis of NoSQL and relational databases with MongoDB and MySQL. **Materials today: PROCEEDINGS**, Elsevier, v. 24, p. 2036–2043, 2020.

KALUŽA, M.; TROSKOT, K.; VUKELIĆ, B. Comparison of front-end frameworks for web applications development. **Zbornik Veleučilišta u Rijeci**, Veleučilište u Rijeci, v. 6, n. 1, p. 261–282, 2018.

KENDALL, S. C. et al. **A note on distributed computing**. Mountain View, CA: Sun Microsystems, Inc., 1994.

KHAN, S.; MANE, V. SQL support over MongoDB using metadata. **International Journal of Scientific and Research Publications**, v. 3, n. 10, p. 1–5, 2013.

KUMAR, A.; SINGH, R. K. Comparative analysis of AngularJs and ReactJs. **International Journal of Latest Trends in Engineering and Technology**, v. 7, n. 4, p. 225–227, 2016.

LEI, K.; MA, Y.; TAN, Z. Performance comparison and evaluation of web development technologies in PHP, Python, and Node.Js. In: IEEE. **2014 IEEE 17th International Conference on Computational Science and Engineering**. Washington, DC, 2014. p. 661–668.

MATIAS, A. B. et al. Finanças corporativas de longo prazo: criação de valor com sustentabilidade financeira. **São Paulo: Atlas**, v. 2, 2007.

NURSEITOV, N. et al. Comparison of JSON and XML data interchange formats: a case study. **Caine**, Citeseer, v. 9, p. 157–162, 2009.

OPENJS FOUNDATION. **Introduction to Node.Js**. 2011. https://nodejs.dev/en/learn/introduction-to-nodejs/. Acessado em: 30/05/2023.

____. **Express 4.18.1**. 2022. https://expressjs.com/. Acessado em: 30/05/2023.

PEZOA, F. et al. Foundations of JSON schema. In: **Proceedings of the 25th international conference on World Wide Web**. Montréal, Québec, Canada: International World Wide Web Conferences Steering Committee, 2016. p. 263–273.

RAWAT, P.; MAHAJAN, A. N. ReactJS: A modern web development framework. **International Journal of Innovative Science and Research Technology**, v. 5, n. 11, p. 698–702, 2020.

REDDY, M. **API Design for C++**. Burlington, MA: Elsevier, 2011.

RUSCHEL, M. E.; FREZZA, R.; UTZIG, M. J. S. O impacto do SPED na contabilidade desafios e perspectivas do profissional contábil. **Revista Catarinense da Ciência Contábil**, v. 10, n. 29, p. 09–26, 2011.

WANG, G.; TANG, J. The NoSQL principles and basic application of cassandra model. In: IEEE. **2012 International Conference on Computer Science and Service System**. Washington, DC, 2012. p. 1332–1335.

WEI-PING, Z.; MING-XIN, L.; HUAN, C. Using MongoDB to implement textbook management system instead of MySQL. In: IEEE. **2011 IEEE 3rd International Conference on Communication Software and Networks**. Xi'an, China, 2011. p. 303–305.

ZHAO, Y.; SANDAHL, K. Xml-based frameworks for internet commerce. **Enterprise Information Systems II**, Springer, p. 197–202, 2001.

NUP: 23081.088473/2023-78 **Prioridade:** Normal

Homologação de ata de defesa de TCC e estágio de graduação

125.322 - Bancas examinadoras de TCC: indicação e atuação

COMPONENTE

Ordem	Descrição	Nome do arquivo
7	Trabalho de conclusão de curso (TCC) (125.32)	tcc.pdf

Assinaturas

19/07/2023 13:34:19

GABRIEL HADDAD VIEIRA (Aluno de Graduação - Aluno Regular)
07.09.05.02.0.0 - Ciência da Computação - Bacharelado - 13881

19/07/2023 15:14:11

GIOVANI RUBERT LIBRELOTTO (PROFESSOR DO MAGISTÉRIO SUPERIOR (Ativo))
07.66.00.00.0.0 - DEPARTAMENTO DE LINGUAGENS E SISTEMAS DE COMPUTAÇÃO - DLSC