Associação Propagadora Esdeva
Centro Universitário Academia – UniAcademia
Curso de Engenharia de Software
Trabalho de Conclusão de Curso – Artigo

Desenvolvimento de um sistema para análise de dados a partir da plataforma de gestão de projetos Bitrix24, utilizando React.js e Node.js

Davi Queiroz Barreira Pena[1]
Centro Universitário Academia, Juiz de Fora, MG

Marcos Alexandre Miguel[2]
Centro Universitário Academia, Juiz de Fora, MG

Linha de pesquisa: Engenharia de Software

Resumo

Este trabalho descreve a criação de um sistema de análise de dados integrado com a API de gerenciamento de tarefas do Bitrix24. Esse sistema gera métricas e gráficos detalhados, oferecendo visualizações que podem ajudar em tomadas de decisão ao longo do desenvolvimento de software. Além disso, disponibiliza a possibilidade de adicionar filtros, alguns como filtros de membros, grupos e tags, juntamente com o filtro principal de intervalo de datas. A aplicação foi feita em React.js, Node.js e PostgreSQL, sua implementação é discutida em detalhes, destacando os desafios enfrentados e os benefícios proporcionados. O projeto exemplifica a aplicação de princípios de engenharia de software para aprimorar a eficiência operacional e a tomada de decisões baseadas em dados em um ambiente real de colaboração que utiliza o Bitrix24 como plataforma de gerenciamento de tarefas.

Palavras-chave: Gestão de projetos, gestão de tarefas, API, análise de dados, Dashboard, Bitrix24, desenvolvimento de software, Node.js, React.js, PostgreSQL.

Abstract:

This work describes the development of a data analysis system integrated with the Bitrix24 task management API. This system generates detailed analysis and graphs, providing insights that can help with decision-making throughout software development. Additionally, it provides the possibility of adding filters, some such as member, group, and tag filters, along with the main date range filter. The implementation is discussed in detail, highlighting the encountered challenges and the benefits delivered. This project serves as an illustration of the application of

[1] Discente do Curso de Engenharia de Software do Centro Universitário Academia - UniAcademia. E-mail: dvbarreira@hotmail.com.
[2] Docente do Curso de Engenhara de Software do Centro Universitário Academia - UniAcademia. Orientador.

software engineering principles to enhance operational efficiency and data-driven decision-making within a real collaborative environment utilizing Bitrix24 as its task management platform.

1. INTRODUÇÃO

A eficácia na gestão de tarefas e a análise de dados são itens essenciais para o sucesso de projetos e operações empresariais, e são consequentemente melhorados de acordo com o surgimento e aprimoramento das tecnologias mais recentes. Além de sua função operacional, existem vantagens no uso de aplicativos que auxiliam o gerenciamento de projetos como melhoria na produtividade, comunicação, integração, simulação e precisão, e neste contexto, a análise de dados e a gestão de tarefas desempenham um papel essencial na construção do sucesso organizacional, obtenção de vantagens competitivas e melhor gerenciamento do projeto (CANDIDO, 2021).

O trabalho desenvolvido tem como objetivo relatar a criação de um sistema de análise de dados que se integra com a API[3] de gerenciamento de tarefas da plataforma Bitrix24, adotada como núcleo das operações em gestão de projetos e tarefas por mais de doze milhões de organizações (BITRIX24, 2023). Uma vez que o trabalho de integração, formatação e arquitetura do sistema já foram feitos, este projeto se destaca por disponibilizar a estrutura para que os usuários possam usar como base, de forma escalável, de acordo com as suas necessidades. Além disso, também fornece a geração de métricas informativas e a apresentação de gráficos esclarecedores, proporcionando um entendimento melhor sobre as tarefas associadas aos projetos. Adicionalmente, ele proporciona um conjunto de filtros que flexibiliza a experiência do usuário, disponibilizando opções como filtragem por intervalo de datas, membros, grupos (projetos) e *tags*, mostrar apenas dados de tarefas de *hotfix*, mostrar apenas dados de tarefas de alta prioridade e mostrar apenas dados de membros selecionados.

O que torna este projeto viável é o suprimento de informações que não estão completas ou presentes na plataforma original do Bitrix24, assim gerando um *dashboard* intuitivo e de fácil navegação, visando otimizar a eficiência operacional e impulsionar a tomada de decisões baseadas em dados em um ambiente colaborativo no qual o Bitrix24 protagoniza a plataforma de gerenciamento de tarefas.

[3] API. Disponível em: <https://aws.amazon.com/pt/what-is/api/>. Acesso em: 05 novembro 2023.

A implementação foi concretizada por meio de tecnologias atuais, conta com um *back-end* REST[4] em Node.js[5], *front-end* em React.js[6], base de dados em PostgreSQL, e todo o código está versionado no GitHub[7], disponível publicamente, com exceção do arquivo ".env", tanto do *back-end*, que contém as chaves privadas de integração com o Bitrix24 da empresa que disponibilizou os dados, quanto do *front-end*, que dependerá do ambiente onde o projeto será rodado. Para o funcionamento deste projeto, estes ".env" deverão ser criados e adicionadas neles as propriedades para a integração do seu próprio projeto no Bitrix24. Foram também utilizadas bibliotecas auxiliares tanto no *back-end* quanto no *front-end*, entre elas estão em destaque ApexCharts[8] e o *framework* Express.js[9], tanto as bibliotecas quanto a configuração dos arquivos ".env", serão discutidas mais à frente em detalhes. Neste estudo, a implementação é analisada detalhadamente, evidenciando os desafios superados e os benefícios derivados desse sistema.

Aplicações Públicas[10], adicionam novas funções e características à conta do Bitrix24. É possível tanto usá-las como uma solução para quem deseja desenvolver uma aplicação, que segundo a área do desenvolvedor[11] envolveria bastante burocracia até sua publicação no mercado interno do Bitrix24, quanto para quem deseja uma solução rápida e menos flexível, não necessitando da criação de um sistema fora da plataforma. Assim existe uma lista de aplicações já criadas no Bitrix24, mas nenhuma que fosse adaptável e implementasse todas as funções presentes neste sistema, além de possuírem flexibilidade e a possível escalabilidade para a adição de novas funções.

Com base em dados reais de uma empresa, foi possível concluir este projeto, e para isso, foi assinado um termo de confidencialidade anexo neste trabalho, respeitando integralmente a privacidade dos dados disponibilizados. Para o funcionamento do sistema de acordo com as credenciais do Bitrix24 do usuário, serão necessárias configurações adicionais cujos detalhes serão discutidos no tópico 3.2.4.

---

[4] REST. Disponível em: <https://redhat.com/en/topics/api/what-is-a-rest-api/>. Acesso em: 08 dezembro 2023.
[5] Node.js. Disponível em: <https://nodejs.org/>. Acesso em: 05 novembro 2023.
[6] React.js. Disponível em: <https://react.dev/>. Acesso em: 05 novembro 2023.
[7] GitHub. Disponível em: <https://github.com/>. Acesso em: 05 novembro 2023.
[8] Apex Charts. Disponível em: <https://apexcharts.com/>. Acesso em: 05 novembro 2023.
[9] Express.js. Disponível em: <https://expressjs.com/>. Acesso em: 05 novembro 2023.
[10] Bitrix24, aplicações públicas. Disponível em: <https://bitrix24.com/apps/>. Acesso em: 17 novembro 2023.
[11] Bitrix24, desenvolver aplicação pública. Disponível em: <https://training.bitrix24.com/support/training/course/?COURSE_ID=169&CHAPTER_ID=020024&LESSON_PATH=13643.20024/>. Acesso em: 17 novembro 2023.

2. REVISÃO BIBLIOGRÁFICA

2.1. Gestão de Projetos

Para alcançar uma gestão de projetos eficiente, é preciso entender a empresa que está em pauta, pois cada uma tem a sua realidade única e, portanto, diferentes ferramentas disponíveis, vantagens e desvantagens para um processo de sucesso, já que “o alcance da excelência em gerenciamento de projetos não é possível sem um processo repetitivo que possa ser utilizado em cada projeto. Esse processo repetitivo é a metodologia de gerenciamento de projetos” (XAVIER, 2012).

Para CANDIDO (2012), a história tem sido marcada pela busca contínua por dominar recursos disponíveis e transformá-los em produtos que visam proporcionar conforto, satisfação e, em última instância, crescimento socioeconômico. Isto evoluiu ao longo do tempo, tornando-se um elemento de competição nos mercados globais e de soberania em escala global. Diante desse novo contexto, as organizações têm buscado soluções que as conduzam a vantagens competitivas e liderança nos mercados em que atuam, da forma mais eficiente possível, que leva ao gerenciamento de projetos, que “cuja base é o fortalecimento da equipe, tem se mostrado uma ótima opção para conduzir empreendimentos não recorrentes e com prazos determinados de duração”. Diante desta necessidade de adaptação que as organizações buscam para competir no mercado, surge a relevância de também explorar abordagens de gerenciamento de tarefas para otimizar os times, e, portanto, a entrega de um produto de qualidade com eficiência e segurança.

2.2. Kanban e Gerenciamento de Tarefas

Segundo SILVA, F. Alan e NETO (2012), originado no Japão em 1997 com o intuito de aprimorar a produção de automóveis, o método Kanban introduziu um sistema inovador no qual a produção é ajustada conforme a demanda, marcando uma mudança significativa na abordagem tradicional da época. A disseminação desse método ganhou destaque em 2007, quando Rick Garber e David J. Anderson compartilharam suas descobertas, que desde então, equipes de desenvolvimento de *software* têm adotado o Kanban como uma estratégia eficaz para alcançar resultados eficientes. O principal ponto desta abordagem reside na visualização e monitoramento do fluxo de trabalho, proporcionando uma compreensão clara das tarefas em andamento.

O termo “Kanban”, em japonês, traduz-se como “sinal visual”, uma referência direta à prática de utilizar sinais visuais para indicar o progresso e a demanda nas etapas do processo. Uma de suas características marcantes é a capacidade de evidenciar problemas existentes no processo, permitindo que as equipes

identifiquem gargalos, ineficiências e oportunidades de melhoria de forma rápida e eficaz. Entre as diversas metodologias de desenvolvimento de *software*, o Kanban destaca-se por sua abordagem menos prescritiva, incentivando as equipes a adotarem práticas flexíveis que se alinhem melhor às suas necessidades específicas, promovendo uma cultura de melhoria contínua e adaptação ágil (SILVA, F. Alan e NETO, 2012).

2.3. React.js e Node.js para Desenvolvimento de Software

O React.js permite construir interfaces de usuário a partir de peças individuais chamadas componentes, que podem ser combinados em telas, páginas e aplicativos inteiros, ele é uma ferramenta versátil que proporciona uma experiência uniforme, seja trabalhando individualmente ou colaborando com milhares de outros desenvolvedores. Ele foi projetado com a finalidade de permitir a perfeita integração de componentes criados por diversas pessoas, equipes e organizações independentes e para desenvolver com React.js, escreve-se componentes que combinam código e marcação, onde os componentes são essencialmente funções JavaScript (React.js, 2023).

Segundo CHEN (2019) é praticamente inquestionável que em breve haverá uma crescente demanda por esta estrutura e suas capacidades. A natureza do React.js como uma biblioteca que efetivamente apoia as empresas na realização de seus objetivos consolida sua posição no mercado, assegurando, assim, sua relevância contínua nos anos vindouros.

O Node.js adota uma abordagem distinta ao utilizar um *loop* de eventos como parte integrante de sua estrutura em tempo de execução, em contraste com outros sistemas que tipicamente exigem uma chamada bloqueante para iniciar esse *loop*, nele não é necessário efetuar esta chamada inicial para o *loop* de eventos. A plataforma simplesmente entra no *loop* de eventos após a execução do *script* de entrada e o deixa quando não há mais *call-backs* a serem processados. Esse comportamento é similar ao do JavaScript em navegadores, onde o funcionamento do *loop* de eventos permanece em segundo plano para o usuário, adicionalmente, o Node.js confere ao protocolo HTTP um *status* de alta prioridade, sendo desenhado com ênfase em *streaming* e baixa latência. Esta característica o torna particularmente adequado como base para soluções de engenharia de *software* (Node.js, 2023).

2.4. Engenharia de Software

"O método de engenharia observa as soluções existentes, sugere as soluções mais adequadas, desenvolve, mede e analisa, e repete até que nenhuma melhoria adicional seja possível. Isto é uma abordagem orientada à melhoria evolutiva que assume a existência de algum modelo do processo ou produto de *software* e modifica este modelo com propósito de melhorar os objetos do estudo" (TRAVASSOS, GUROV e AMARAL, 2002).

A importância da engenharia de *software*, conforme destacada por LESSA (2009), desempenha um papel fundamental atualmente, proporcionando estrutura e metodologia para o desenvolvimento de *software* de alta qualidade. É a capacidade de organizar e gerenciar processos de desenvolvimento de *software* de forma eficiente, permitindo a criação de aplicativos confiáveis, seguros e escaláveis. Sua origem remonta à década de 1950, quando o rápido crescimento da indústria de *software* levou à necessidade de abordagens mais sistemáticas e consistentes para o desenvolvimento de programas. Desde então, essa disciplina evoluiu, adotando melhores práticas e técnicas avançadas de gerenciamento de projetos, tornando-se um pilar fundamental no aprimoramento de metodologias de desenvolvimento de *software* das organizações.

2.5. Revisão Sistemática

A revisão buscou identificar soluções existentes para integração de dados e criação de *dashboards* na plataforma Bitrix24, com foco em gerenciamento de tarefas. Entretanto, a busca abrangente realizada no Google, na plataforma original do Bitrix24 e no Google Acadêmico não resultou na descoberta de artigos específicos que abordassem diretamente a integração com o Bitrix24. Embora alguns artigos relacionados a diferentes tópicos pertinentes ao gerenciamento de projetos e desenvolvimento com React.js tenham sido encontrados, nenhum deles abordava sobre a temática da integração com o Bitrix24. A escassez de informações específicas limitou a extração de dados diretamente relacionados ao escopo do trabalho, impossibilitando, assim, a avaliação da qualidade metodológica de artigos específicos referentes à integração com o Bitrix24. Esta ausência de literatura especializada realça a relevância do presente trabalho ao preencher esta falta de informação identificada na pesquisa existente, proporcionando uma solução importante para integração e análise de dados no contexto do Bitrix24. Destaca-se que essa contribuição é interessante tanto para organizações que buscam aprimorar o gerenciamento de projetos quanto para a comunidade acadêmica, que, até o momento, carece de referências diretas sobre trabalhos similares nessa área específica.

3. Metodologia de Desenvolvimento

Este projeto surgiu da necessidade de desenvolver uma aplicação dedicada à visualização de métricas extraídas dos dados retornados da API de gerenciamento de tarefas da plataforma Bitrix24, visando simplificar a análise de dados relacionados à gestão de projetos e tarefas. A implementação técnica compreende o uso de Node.js com Express.js para o desenvolvimento do *back-end*, enquanto o *front-end* é construído em React.js. Para a gestão eficiente do banco de dados, optou-se pelo PostgreSQL como Sistema de Gerenciamento de Banco de Dados. Esta escolha estratégica de ferramentas reflete a busca por uma arquitetura sólida, rápida, escalável e eficiente para atender às demandas específicas deste projeto.

3.1. Ferramentas de Desenvolvimento

As decisões relativas às ferramentas desempenham um papel essencial no sucesso do desenvolvimento, impactando diretamente a eficiência, custo e escalabilidade da aplicação. Dependendo das necessidades da organização, é imperativo realizar uma cuidadosa avaliação e seleção das ferramentas mais adequadas para compor o sistema. (SEBRAE, 2013).

3.1.1. Back-end

Desenvolvido em Node.js, como já discutido anteriormente no ponto 2.3, ele é bem utilizado principalmente para aplicações de pequeno e médio porte, que o tempo de execução JavaScript no lado do servidor é aproveitado para criar aplicações altamente escaláveis e eficientes em termos de recursos, permitindo a criação de aplicações de servidor e API robustas e de alto desempenho (Node.js, 2023), além de poder utilizar as diversas bibliotecas de JavaScript para auxiliar no desenvolvimento. Em conjunto, foi utilizado o *framework* Express.js para melhor organização e facilidade no tratamento de rotas e solicitações. Juntamente a estas escolhas, foram utilizadas bibliotecas adicionais, algumas principais são:

- JWT[12]: é uma biblioteca vastamente utilizada para autenticar e autorizar solicitações, proporcionando segurança e integridade aos dados transmitidos, é utilizada também no sistema para manter o usuário logado por um tempo determinado.

[12] JWT. Disponível em: <https://https://jwt.io//>. Acesso em: 17 novembro 2023.

- node-postgres[13]: essencial para interagir com bancos de dados PostgreSQL, proporcionando uma camada de abstração que simplifica consultas e operações no banco de dados.
- Nodemon[14]: é usada para reiniciar automaticamente o servidor durante o desenvolvimento sempre que o código-fonte é modificado.

As escolhas estratégicas de tecnologias, incluindo Node.js com o *framework* Express.js, e as bibliotecas suplementares em consideração, constituem a base necessária para alcançar os objetivos do servidor, e não apenas visam atingir os propósitos fundamentais do servidor, mas também desempenham um papel crítico na viabilização da integração fluida com o *front-end* e a infraestrutura de banco de dados.

3.1.2. Front-end

Elaborado em React.js, biblioteca comentada mais detalhadamente anteriormente no ponto 2.3, proporciona a este projeto uma sinergia de linguagem com o *back-end*. Utilizando funções JavaScript para a construção de componentes (React.js, 2023), a escolha de uma linguagem unificada para o desenvolvimento tanto do *back-end* quanto do *front-end* oferece a vantagem de permitir que o mesmo desenvolvedor contribua em ambas as áreas, promovendo uma sinergia eficiente e facilitando a colaboração ao longo do ciclo de desenvolvimento. Algumas das principais bibliotecas adicionadas ao *front-end* do projeto são:

- MUI[15] e Material Icons[16]: são um conjunto completo de componentes de interface de usuário predefinidos e ícones, simplificando o *design* e a criação de aplicativos atraentes e consistentes.
- Redux[17]: para gerenciar o estado da aplicação de maneira centralizada, simplificando o fluxo de dados e aprimorando a escalabilidade e a manutenção do aplicativo.
- React Apex Charts[18]: é utilizado um *wrapper* do Apex Charts para o React.js, que já vem pronto para ser integrado na aplicação. É a biblioteca de gráficos do sistema, disponibilizando uma visualização elegante e interativa.

---

[13] node-postgres. Disponível em: <https://node-postgres.com/>. Acesso em: 17 novembro 2023.
[14] Nodemon. Disponível em: <https://nodemon.io/>. Acesso em: 17 novembro 2023.
[15] MUI. Disponível em: <https://mui.com/>. Acesso em: 05 novembro 2023.
[16] Material Icons. Disponível em: <https://mui.com/material-ui/material-icons/>. Acesso em: 17 novembro 2023.
[17] Redux. Disponível em: <https://redux.js.org/>. Acesso em: 17 novembro 2023.
[18] React Apex Charts. Disponível em: <https://apexcharts.com/docs/react-charts/>. Acesso em: 05 novembro 2023.

- ESLint[19]: No desenvolvimento de código em JavaScript, o ESLint é implementado para manter um código limpo, consistente e livre de erros, aprimorando a qualidade e a legibilidade do código-fonte.
- Prettier[20]: utilizado em conjunto com o ESLint para automatizar a formatação do código, garantindo uma apresentação uniforme e configurável, que aprimora a padronização e a legibilidade do código-fonte. É necessário configurar o ESLint para funcionar corretamente com o Prettier.

Discorrendo sobre essas bibliotecas adicionais, torna-se evidente o impacto significativo que exercem na expansão das capacidades do projeto, desempenhando um papel crucial ao aprimorar a eficiência do desenvolvimento, oferecendo benefícios como a disponibilidade de componentes reutilizáveis, formatação automática do código, compartilhamento global de estados, entre outros. A melhor escolha das ferramentas contribui de maneira direta para o alcance bem-sucedido dos objetivos previamente estabelecidos, solidificando a importância estratégica dessas escolhas no contexto do desenvolvimento como um todo do projeto, englobando o *back-end*, o *front-end* e o sistema de gerenciamento de banco de dados que mais se adeque às suas especificações.

3.1.3. Sistema de Gerenciamento de Banco de Dados (SGBD)

O PostgreSQL é um poderoso sistema de banco de dados relacional de código aberto, com mais de 35 anos de desenvolvimento. Possui uma ampla gama de recursos e reputação consolidada de ser confiável, eficiente e compatível com diversas ferramentas e bibliotecas. É uma escolha sólida para sistemas, destacando-se pela robustez e estabilidade que garantem a integridade dos dados, tornando-o ideal para sistemas críticos. Além disso, oferece um conjunto abrangente de recursos, suportando tipos de dados personalizados, procedimentos armazenados, gatilhos e funções que permitem modelar dados e lógica complexos. Em resumo, o PostgreSQL é uma escolha poderosa para compor arquiteturas de projetos de *software* devido à sua confiabilidade, robustez de recursos e desempenho (POSTGRESQL 2023).

3.2. Arquitetura do Projeto

No processo operacional do *back-end*, uma requisição é direcionada à API do Bitrix24, incorporando filtros de data inicial e final de criação. Esta consulta

[19] ESLint. Disponível em: <https://eslint.org/>. Acesso em: 17 novembro 2023
[20] Prettier. Disponível em: <https://prettier.io/>. Acesso em: 17 novembro 2023

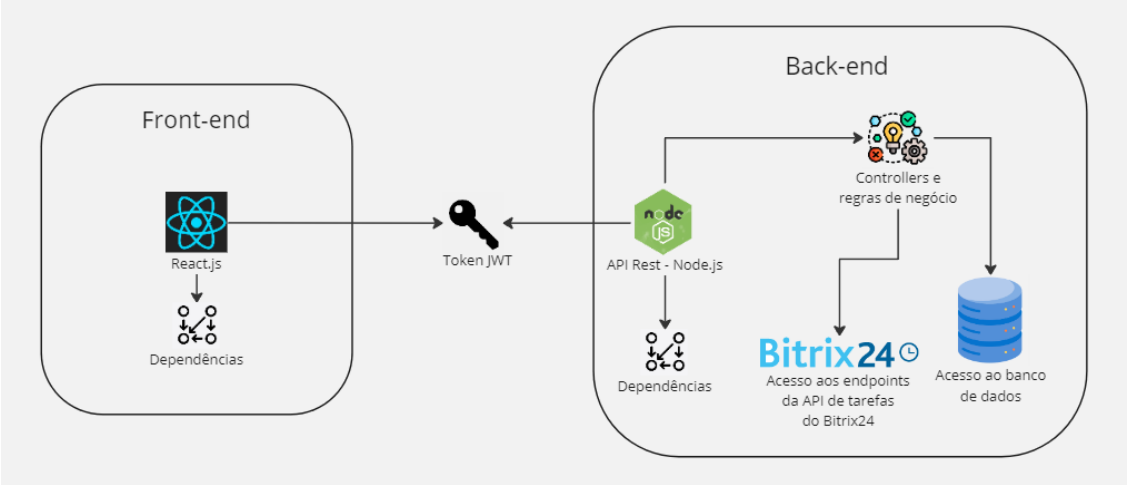
dinâmica proporciona de forma *online* todas as tarefas relacionadas, as quais, antes de apresentadas ao *front-end*, passam por um dos principais e mais complexos processos do projeto, que é a formatação destas tarefas para melhor adequação aos filtros e métricas do sistema. Estas tarefas ao chegarem efetivamente no *front-end,* são submetidas a filtros adicionais com o propósito de otimizar tanto o desempenho quanto a experiência do usuário. Essa estratégia visa diminuir a significativa latência associada à requisição à API do Bitrix24, resultando em uma navegação mais fluida e agradável para o usuário.

O código fonte do projeto está disponível na plataforma GitHub, no seguinte link: https://github.com/daviqz/TCC.

3.2.1. Diagrama da Arquitetura

A arquitetura parte do princípio de que a interação entre a interface do usuário (*front-end*) e a camada lógica do sistema (*back-end*) ocorre por meio de requisições HTTP, nas quais são utilizados *tokens* JWT para autenticação. Essa abordagem de autenticação proporciona segurança, pois cada requisição enviada ao servidor inclui um *token* que o servidor pode validar para verificar a autenticidade do usuário (HASURA, 2020).

Figura 1 – Arquitetura do sistema



Fonte: Elaboração Própria.

No fluxo operacional da arquitetura do sistema, conforme ilustrado na Figura 1, o *front-end* inicia o processo enviando uma requisição ao *back-end*, acompanhada do *token* JWT. O *back-end*, por sua vez, valida essa requisição, interrompendo o processo em caso de erro ou permitindo a continuidade em caso de sucesso, que, uma vez superada essa etapa, a operação direciona-se para a próxima etapa. Ao atingir a camada lógica, o sistema integra a API do Bitrix24 para

a requisição das tarefas disponíveis, que, retornando de forma *online* ao sistema, passam por uma formatação para melhor manipulação no *front-end*. Ao longo do processo descrito, o banco de dados da aplicação é acessado para buscar as chaves de acesso do Bitrix24, necessárias em todas as requisições aos *endpoints* da API do Bitrix24.

3.2.2. Front-end

O projeto em React.js conta com alguns arquivos principais de configuração, como o “jsconfig.json”, que é para a configuração do editor de texto, principalmente para importar globalmente um arquivo sem precisar ficar navegando pelas pastas, o “.eslintrc.json”, que é a configuração do ESLint, “.prettierrc.json”, onde estão as configurações do Prettier.

A pasta principal é a “src”, onde estão, por exemplo, os códigos dos componentes, rotas e configuração do Axios e do Redux. Dentro desta pasta existem mais divisões para melhor organização:

- *components*: parte dos componentes que são mais independentes da tela ou do componente que fazem parte.
- *pages*: são os componentes de página do sistema, por sua vez, podem ou não possuir componentes próprios que estão em sinergia direta com a página.
- *routes*: configuração e verificação das rotas do *front-end*, as páginas navegáveis são importadas e transformadas em rotas de navegação.
- *service*: configuração do Axios para ser utilizado com o *back-end*.
- *storage*: configuração do Redux e do local-storage.
- *utils*: possui algumas funções de suporte na aplicação.

Com a estrutura e a navegação definidas no *front-end*, o *back-end* estabelece uma conexão essencial para criar uma experiência de desenvolvimento completa e coesa. Essa integração fluida entre as camadas permite otimizar não apenas a funcionalidade e a eficiência do sistema, mas também promover uma abordagem eficaz ao processo de desenvolvimento.

3.2.3. Back-end

Desenvolvido em arquitetura REST, também chamada RESTful, é um estilo de design para sistemas distribuídos que utiliza solicitações HTTP para acessar e manipular recursos por meio de URIs, seguindo princípios como a utilização de métodos HTTP e a ausência de estado nas comunicações. Neste contexto, os

dados são representados em formato JSON, e a comunicação é sem estado, permitindo que o *front-end* acesse os recursos por meio de *endpoints* da API para realizar operações (RED HAT, 2023).

A estrutura do *back-end* foi organizada em pastas principais cada uma contendo os respectivos arquivos relacionados aos seus nomes, para melhor navegabilidade. Esta estrutura se constrói dentro de uma pasta denominada “src”, onde contém as seguintes pastas:

- *resource*: é comum utilizar este nome em APIs REST, também conhecido como controller, gerencia o fluxo das solicitações e respostas, levando diretamente à um service.
- *service*: é a camada de tratamento dos dados retornados da integration (dados diretos do Bitrix24) ou do repository (dados diretos do PostgreSQL da aplicação).
- *repository*: é a camada de acesso direto com a base de dados do sistema, é o fim do fluxo de acesso quando buscado algum dado próprio da aplicação, retornando dados para o service que a chamou, para assim tratar os dados se necessário.
- *integration*: tem toda a lógica de integração com a API de gerenciamento de tarefas do Bitrix24. Esta é uma camada final de fluxo, faz chamadas por meio do Axios na API do Bitrix24, assim, retornando os dados necessários. Esta integração, requer alteração no arquivo “.env” do sistema para o seu funcionamento.
- *utils*: concentra algumas funções reutilizáveis do sistema.

A abordagem de segmentar os arquivos de acordo com suas funções específicas simplifica a manutenção e promove a escalabilidade do código, facilitando a adaptação de novos desenvolvedores e futuras demandas. É essencial também considerar as configurações do sistema para garantir que tudo funcione corretamente.

3.2.4. Configurações do Sistema

Para o sistema funcionar corretamente, o repositório do GitHub do projeto disponibilizado no seguinte link: https://github.com/daviqz/TCC, dispõe do arquivo README.md, que explica detalhadamente como fazer as configurações necessárias da aplicação. Em suma, o arquivo chamado “example.env” contém o nome das chaves necessárias para o sistema rodar, estas deverão ser configuradas de acordo com as chaves geradas pelo próprio usuário, incluindo por exemplo, no *back-end*, as chaves da API do Bitrix24. Se o *host* for local, basta realizar a

configuração desses arquivos e renomeá-los para ".env". No caso de *hosts* em nuvem, será necessário preencher as chaves necessárias conforme permitido e configurado pelo ambiente específico. Esta configuração existe tanto no *back-end* no *front-end*.

- No ".env" do *back-end* existem algumas chaves de configuração:
  - *DATABASE_USER*, *DATABASE_HOST*, *DATABASE_NAME*, *DATABASE_PASSWORD*, para a conexão com a base de dados.
  - *JWT_SECRET_KEY*, necessário para a verificação do *token* JWT.
  - *BITRIX_CLIENT_ID*, *BITRIX_CLIENT_SECRET*, são as de conexão com a API do Bitrix24, disponibilizadas e criadas dentro da própria plataforma Bitrix24 pelo usuário que tem acesso aos projetos da organização, e a *BITRIX_URL* é a URL do site do Bitrix24 que está sendo integrado, como padrão na "example.env" é a URL do site brasileiro "bitrix24.com.br", caso for utilizado pela organização o "bitrix24.com", esta chave deverá ser alterada.
  - *BASE_FRONT_URL*, a URL de acesso do *front-end*, para o redirecionamento ao autenticar no Bitrix24 no *login* do sistema.
- No ".env" do *front-end* existe apenas uma chave, a *REACT_APP_API_URL*, este é o endereço que o *back-end* disponibiliza para a conexão do *front-end*.

Como configuração opcional, é possível ajustar as regras do ESLint e Prettier, respectivamente nos arquivos ".eslintrc.json" e ".prettierrc." O ".eslintrc.json" estabelece regras de codificação na propriedade "*rules*", enquanto o ".prettierrc" configura as regras de indentação para a formatação do texto.

Ao solidificar as configurações, parte-se do lado técnico para explorar a essência da proposta. Agora, com uma base já descrita, adentra-se na solução mais detalhada, onde a teoria vira prática e as configurações se tornam funcionalidades visíveis e utilizáveis.

4. Solução Proposta

A solução visa aprimorar a gestão de projetos e tarefas no Bitrix24 através da disponibilização da estrutura de um sistema de análise de dados integrado à API de gerenciamento de tarefas do Bitrix24, que seja escalável e que já implemente algumas métricas e gráficos, proporcionando um *dashboard* que pode ser relevante para ajudar no gerenciamento de projetos de organizações, buscando completar informações que muitas vezes estão ausentes nas soluções existentes.

4.1. Dashboard

O projeto propõe visualizações por meio de um *dashboard*, centralizando as métricas e gráficos gerados para oferecer uma visão do desempenho do projeto. O objetivo primordial é otimizar a eficiência operacional e facilitar a tomada de decisões fundamentadas com base nos dados analisados. Além disso, um ponto importante desta nossa solução é a flexibilidade proporcionada pelos conjuntos de filtros. Os usuários podem personalizar a visualização de dados de acordo com suas necessidades específicas, incluindo opções como intervalo de datas, membros da equipe, grupos (projetos) e *tags*. Adicionalmente, é possível filtrar tarefas como *hotfix*, exibir apenas dados de membros selecionados e apresentar exclusivamente informações de alta prioridade.

4.2. Geração de Métricas e Gráficos Informativos

Como ponto central do *dashboard*, permitem uma análise direta das tarefas associadas a projetos. Ao extrair dados da API do Bitrix24, o sistema transforma essas informações em métricas e gráficos que podem ser relevantes em tomadas de decisão, oferecendo uma visão do andamento dos projetos e do desempenho e *status* das tarefas. Além disso, essa abordagem visual facilita a compreensão das informações, tornando a análise mais acessível para os usuários.

4.3. Escalabilidade

A solução proposta foi criada pensando em adaptabilidade e escalabilidade, implementando as funções necessárias para uma análise possivelmente relevante das tarefas em projetos. Diferentemente das aplicações públicas já disponíveis no Bitrix24, o sistema oferece flexibilidade para a adição de novas funções, sem perder a capacidade de se ajustar às necessidades específicas de cada organização. Em essência, a conclusão bem-sucedida da integração e formatação dos dados estabelece as bases para as fases subsequentes, onde as mudanças planejadas podem ser incorporadas, aproveitando eficientemente o trabalho realizado anteriormente e contribuindo para o progresso contínuo do projeto.

4.4. Informações Ausentes em Outras Aplicações

O sistema busca preencher de informações que podem estar ausentes ou incompletas na plataforma original do Bitrix24, assim como em integrações criadas por terceiros. Ao fazer isso, pretende-se disponibilizar uma ferramenta que não apenas seja complementar, mas que possa substituir ou também aprimora significativamente a utilidade das aplicações já existentes. Essa

abordagem visa suprir deficiências e elevar o cenário geral de gestão de projetos, proporcionando uma solução integrada e aperfeiçoada para atender às demandas específicas das organizações.

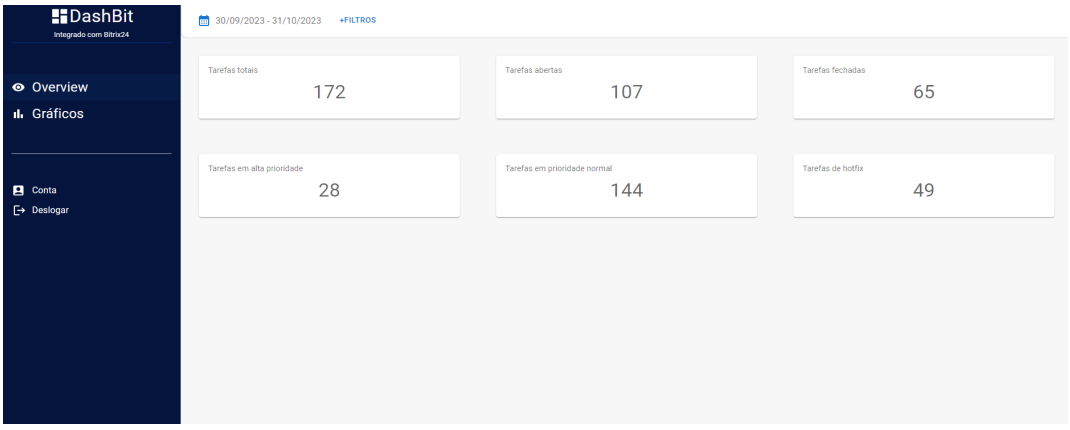
5. Utilização do Sistema

A implementação deste sistema de *dashboard*, oferece uma solução aberta que aborda de forma eficaz os desafios enfrentados pelas empresas ao monitorar e otimizar o desempenho de suas tarefas e projetos no Bitrix24. Ao utilizar dados reais fornecidos por uma empresa, o sistema mostra que pode fornecer informações importantes que capacitam a tomada de decisões mais precisas e estratégicas. Além disso, ao manter o código aberto, permite mudanças de acordo com as necessidades específicas de cada organização, resultando em um sistema que evolui constantemente, promovendo a inovação e a eficácia de outras organizações que possam se beneficiar desta solução.

5.1. Tela Inicial e Layout do Sistema

A ideia da primeira tela (Figura 2) o *login*, é levar uma experiência o usuário que seja visualmente agradável e que gere valor à primeira vista, neste caso, é a tela de *Overview*. O *layout* foi desenvolvido para facilitar a navegação, permitindo que o usuário localize facilmente a área destinada aos filtros do sistema.

Figura 2 – Primeira tela após o *login*



Fonte: Elaboração Própria.

Como mostra a Figura 2, a primeira tela exibida no sistema é a de *Overview*. Na parte superior, encontram-se os filtros de data e filtros adicionais**Erro! Fonte de referência não encontrada.**. À esquerda, o menu de navegação do sistema é mostrado, com as opções *Overview*, Gráficos e Deslogar.

5.2. Overview

A ideia principal desta parte (Figura *3*) é apresentar métricas que independem de filtros para proporcionar valor e garantir uma compreensão fácil para o usuário não se perder com visualizações mais complexas.

Figura 3 – Componentes na tela de *Overview*

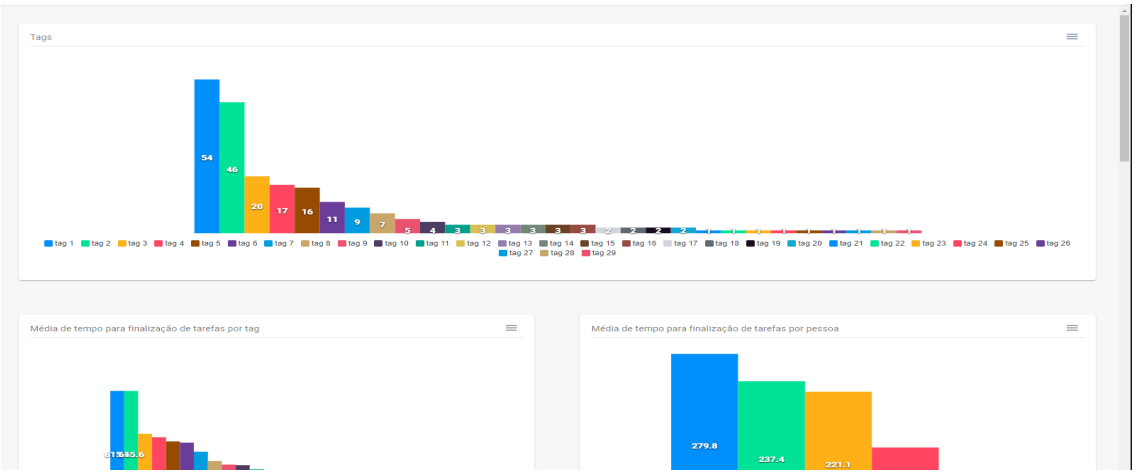
Fonte: Elaboração Própria.

A tela descrita na Figura 3 exibe métricas gerais do sistema como quantidade total de tarefas, tarefas abertas, tarefas fechadas, tarefas em alta prioridade, tarefas em prioridade normal e tarefas de *hotfix*.

5.3. Gráficos

A seção de gráficos (Figura *4)* do sistema oferece uma análise visual de dados das tarefas. Em resumo, os gráficos destacam a distribuição de tarefas por *tags* e fornecem métricas de tempo médio para conclusão, identificam os envolvidos nas tarefas e facilitam a análise de interações na equipe.

Figura 4 – *Layout* da tela de gráficos, mostrando principalmente o gráfico de *tags*



Fonte: Elaboração Própria.

Nesta área descrita na Figura *4*, além dos gráficos chamados "*Tags*", "Média de tempo para a conclusão de tarefas por *tag*" e "Média de tempo para a finalização

de tarefas por pessoa” observados na figura acima, se encontram também nesta área, análises gráficas específicas para participantes, fechadores, criadores, observadores e responsáveis pelas tarefas.

Para uma visão mais prática e detalhada do projeto, segue um vídeo que discorre sobre alguns pontos principais e as funcionalidades do sistema desenvolvido: https://youtu.be/ndD2YQro4cg.

6. Resultados

A implementação da solução proposta tem o potencial de gerar uma série de impactos positivos e significativos benefícios para a gestão de projetos e tarefas, podendo abranger outros pontos de acordo com as necessidades particulares de cada organização. Dentre os pontos gerais, alguns mais notáveis são:

6.1. Tomada de Decisões

A disponibilidade de métricas informativas e gráficos esclarecedores pode permitir uma tomada de decisões mais informada e rápida. Oferecendo aos gestores uma visão do desempenho de projetos específicos e do status de tarefas, capacitando-os a agir prontamente diante de desafios ou oportunidades emergentes.

6.2. Eficiência Operacional

Ao extrair informações mais detalhadas e potencialmente relevantes através do conjunto de filtros, a solução se propõe a otimizar a eficiência operacional, podendo capacitar as equipes para concentrarem seus esforços nas áreas críticas, resultando em melhorias substanciais na produtividade.

6.3. Informação Online

O *dashboard* proporciona uma visão *online* do estado atual dos projetos, essa representação pode facilitar a identificação de padrões, tendências e áreas que exigem atenção imediata, isso sem passar por qualquer processamento ou fila, qualquer mudança nos quadros de tarefas no Bitrix24, será refletida no sistema ao recarregar os dados manualmente. A disponibilidade de informação desse tipo também tem a capacidade de dar a oportunidade para as os gestores das equipes reagirem proativamente às mudanças nas demandas do projeto.

6.4. Preenchimento de Lacunas Informativas

A capacidade do sistema de suprir informações ausentes em sistemas de terceiros ou na plataforma original do Bitrix24, foi o motivo da sua concepção. As lacunas de dados, anteriormente uma limitação, agora podem ser preenchidas, tanto pelas métricas já disponíveis, quanto pela escalabilidade que o sistema oferece caso o usuário queira adicionar alguma funcionalidade nova, ou mudar alguma existente. Podendo levar uma visão mais abrangente e detalhada das operações e contribuir diretamente para uma análise mais fundamentada.

6.5. Adaptação

A flexibilidade da solução possui capacidade de se adaptar às necessidades específicas dos usuários. Além disso, permite adaptações futuras e a adição de novas funcionalidades, contribuindo para uma ferramenta mais versátil ao longo do tempo, como o exemplo do filtro de *hotfix*, que foi adicionado depois por uma demanda específica e está presente apenas nos dados disponibilizados pela organização em questão.

7. Considerações Finais

Ao longo deste trabalho, demonstramos o desenvolvimento e a implementação de um sistema de *dashboard*, integrado com a API de gerenciamento de tarefas do Bitrix24. Para validar sua eficácia, o sistema foi alimentado e submetido a testes utilizando dados reais fornecidos por uma empresa parceira, que futuramente irá utilizar e aprimorar este sistema de acordo com a sua demanda interna, para isto, foi estabelecido um acordo de confidencialidade entre as partes.

A solução tratada no aplicativo desenvolvido implementa recursos que não estão disponíveis ou são limitados, tanto nos aplicativos públicos da plataforma, quanto em outros sistemas integrados com esta API. Este sistema também disponibiliza fácil adição ou alteração de recursos, uma vez que os dados já estão sendo tratados por poucas funções específicas no *back-end*, tornando mais organizado e evitando espalhar complexidade pelo código.

O sistema de *dashboard* foi planejado e construído utilizando tecnologias atuais, amplamente reconhecidas na indústria. A escolha do React.js e Node.js, juntamente com suas bibliotecas correspondentes, representa um investimento estratégico em inovação e eficiência. O React.js, conhecido por sua reatividade e facilidade de desenvolvimento de interfaces de usuário dinâmicas, proporciona uma

experiência de usuário ágil e atraente. O Node.js, por sua vez, oferece uma base sólida para o lado do servidor, garantindo um desempenho robusto e escalabilidade. Além disso, as bibliotecas associadas a estas tecnologias trazem funcionalidades adicionais e uma vasta comunidade de desenvolvedores, garantindo que o sistema esteja preparado para atender às demandas em constante evolução da empresa e do mercado. Estas escolhas refletem uma solução de análise de métricas de tarefas que seja eficiente, ágil e visualmente agradável.

Para os próximos passos do projeto, há diversas oportunidades interessantes a serem exploradas. Entre as abordagens que podem ser consideradas, algumas se destacam como: aprimorar a extração de métricas, permitindo não só o rastreamento temporal, mas também a exportação de dados em formato CSV para análises mais detalhadas. Além disso, a melhoria na apresentação visual das informações, com a introdução de tipos de gráficos variados (como barras, linhas e *donuts*), proporcionaria maior flexibilidade aos usuários. Adicionalmente, a capacidade de personalizar campos diretamente no Bitrix24 adaptaria o sistema de maneira mais precisa às necessidades específicas da empresa. Tornar o sistema um aplicativo público integrado à plataforma Bitrix24 o deixaria mais acessível. A integração com outras plataformas de gerenciamento de tarefas ampliaria sua utilidade, atendendo às diversas demandas. Essas estratégias representam uma expansão significativa, transformando o sistema em uma ferramenta mais poderosa e versátil para análise de métricas de tarefas.

Por fim, desafios e situações inesperadas podem surgir ao longo do processo, e esta ferramenta demonstra a importância da implementação de uma análise de dados para o gerenciamento de projetos, disponibilizando métricas valiosas que convergem em informações cruciais para a visualização de pontos positivos e melhorias nos times, facilitando a identificação de possíveis obstáculos e oportunidades de aprimoramento. Com as métricas e informações apresentadas neste *dashboard*, os usuários do Bitrix24 podem tomar decisões mais informadas e estratégicas, contribuindo para a melhoria contínua da gestão do projeto, um gerenciamento mais eficaz das tarefas e um desempenho geral mais otimizado da equipe.

Referências:

BITRIX24. Disponível em: https://bitrix24.com.br. Acesso em 28 nov. 2023.

CANDIDO, Roberto et al. Gerenciamento de projetos. Curitiba: Aymará Educação (2012). Disponível em: http://repositorio.utfpr.edu.br/jspui/handle/1/2061. Acesso em 28 out. 2023.

CARDOSO, Rubem Alves. Uma análise do Kanban em cenários de desenvolvimento distribuído de software. Curitiba: Aymará Educação (2015). Disponível em: http://repositorio.utfpr.edu.br/jspui/handle/1/7423. Acesso em 28 out. 2023.

CHEN, Songtao; THADURI, Upendar Rao; BALLAMUDI, Venkata Koteswara Rao Front-End Development in React: An Overview. Engineering International. (2019). Disponível em: https://www.researchgate.net/profile/Venkata-Ballamudi/publication/374154236_Front-End_Development_in_React_An_Overview/links/6510d94f61f18040c220eb13/Front-End-Development-in-React-An-Overview.pdf. Acesso em 28 out. 2023.

HASURA. JWT Authentication: A Practical Guide. Disponível em:

https://hasura.io/blog/best-practices-of-using-jwt-with-graphql. Acesso em: 28 nov.

2023.

LESSA, Rafael Orivaldo; LESSA JUNIOR, Edson Orivaldo. Modelos de processos de engenharia de software. (2009) Disponível em: https://ead.uepg.br/apl/sigma/assets/editais_antigos/PS0056E0077.pdf.

POSTGRESQL. Disponível em: https://postgresql.org. Acesso em: 27 nov. 2023.

REDHAT. O que é uma API REST. (2013). Disponível em: https://www.redhat.com/pt-br/topics/api/what-is-a-rest-api. Acesso em 17 nov. 2023.

SEBRAE. É importante escolher as tecnologias adequadas para a sua empresa. (2013). Disponível em: https://sebrae.com.br/sites/PortalSebrae/artigos/e-importante-escolher-as-tecnologias-adequadas-para-a-sua-empresa,0831ebb38b5f2410VgnVCM100000b272010aRCRD. Acesso em 17 nov. 2023.

SILVA, Diogo Vinícius de S., F. Alan de O., NETO, Pedro Santos. Os benefícios do uso de Kanban na gerência de projetos de manutenção de software. Anais do VIII Simpósio Brasileiro de Sistemas de Informação. SBC, (2012). 120 p. Disponível em: https://sol.sbc.org.br/index.php/sbsi/article/view/14454/14300. Acesso em 28 out. 2023.

TRAVASSOS, Guilherme Horta; GUROV, Dmytro; AMARAL, E. A. G. G. Introdução à engenharia de software experimental. (2002). Disponível em: https://www.pesc.coppe.ufrj.br/uploadfile/es59002.pdf. Acesso em 28 out. 2023.

XAVIER, Carlos Magno et al. Metodologia de gerenciamento de projetos. Rio de Janeiro: Brasport (2005). Disponível em: http://g2b.com.br/downloads/07_metodologia_gerenciamento_de_projetos_carlos_magno_da_silva_xavier_2012.pdf. Acesso em 28 out. 2023.