UNIVERSIDADE FEDERAL DE CAMPINA GRANDE
CENTRO DE ENGENHARIA ELÉTRICA E INFORMÁTICA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

FELIPE OLIVEIRA DA SILVA

LSI ANALYTICS: SIMPLIFICANDO O DESENVOLVIMENTO DE APLICAÇÕES WEB DE BUSINESS INTELLIGENCE

CAMPINA GRANDE - PB

2024

FELIPE OLIVEIRA DA SILVA

LSI ANALYTICS: SIMPLIFICANDO O DESENVOLVIMENTO DE APLICAÇÕES WEB DE BUSINESS INTELLIGENCE

Trabalho de Conclusão Curso apresentado ao Curso Bacharelado em Ciência da Computação do Centro de Engenharia Elétrica e Informática da Universidade Federal de Campina Grande, como requisito parcial para obtenção do título de Bacharel em Ciência da Computação.

BANCA EXAMINADORA:

Cláudio de Souza Baptista

Orientador – UASC/CEEI/UFCG

Carlos Eduardo Santos Pires

Examinador – UASC/CEEI/UFCG

Francisco Vilar Brasileiro

Professor da Disciplina TCC – UASC/CEEI/UFCG

Trabalho aprovado em: 15 de Maio de 2024.

CAMPINA GRANDE - PB

RESUMO

Este trabalho apresenta o LSI Analytics, uma plataforma de Business Intelligence (BI) projetada para permitir aos usuários a análise e visualização eficiente de dados a partir de diversas fontes. A plataforma oferece uma abordagem intuitiva e flexível para a criação de gráficos, facilitando a tomada de decisões informadas pelos usuários. O estudo aborda os desafios enfrentados no desenvolvimento da plataforma, incluindo a garantia de renderização dinâmica e manipulação eficiente de conjuntos de dados. Como perspectivas futuras, são propostas melhorias adicionais na plataforma, visando ampliar suas capacidades de análise de dados e visualização. Este estudo contribui para o avanço no campo do Business Intelligence, fornecendo uma solução acessível e eficaz para a análise de dados.

LSI ANALYTICS: SIMPLIFICANDO O DESENVOLVIMENTO DE APLICAÇÕES WEB DE BUSINESS INTELLIGENCE

ABSTRACT

This work presents LSI Analytics, a Business Intelligence (BI) platform designed to allow users to efficiently analyze and visualize data from various sources. The platform offers an intuitive and flexible approach to creating charts, making it easier for users to make informed decisions. The study addresses challenges faced in developing the platform, including ensuring dynamic rendering and efficient manipulation of datasets. As future perspectives, additional improvements to the platform are proposed, aiming to expand its data analysis and visualization capabilities. This study contributes to advancement in the field of Business Intelligence by providing an affordable and effective solution for data analysis.

LSI ANALYTICS: SIMPLIFICANDO O DESENVOLVIMENTO DE APLICAÇÕES WEB DE BUSINESS INTELLIGENCE

Felipe Oliveira da Silva
Unidade Acadêmica de Sistemas e Computação - UASC
Universidade Federal de Campina Grande - UFCG
Campina Grande, Paraíba, Brasil
felipe.oliveira.silva@ccc.ufcg.edu.br

Cláudio de Souza Baptista
Unidade Acadêmica de Sistemas e Computação - UASC
Universidade Federal de Campina Grande - UFCG
Campina Grande, Paraíba, Brasil
baptista@computacao.ufcg.edu.br

RESUMO

Este trabalho apresenta o LSI Analytics, uma plataforma de Business Intelligence (BI) projetada para permitir aos usuários a análise e visualização eficiente de dados a partir de diversas fontes. A plataforma oferece uma abordagem intuitiva e flexível para a criação de gráficos, facilitando a tomada de decisões informadas pelos usuários. O estudo aborda os desafios enfrentados no desenvolvimento da plataforma, incluindo a garantia de renderização dinâmica e manipulação eficiente de conjuntos de dados. Como perspectivas futuras, são propostas melhorias adicionais na plataforma, visando ampliar suas capacidades de análise de dados e visualização. Este estudo contribui para o avanço no campo do Business Intelligence, fornecendo uma solução acessível e eficaz para a análise de dados.

Palavras-chave: Analytics, Open-source, Visualização Gráfica, BI

Repositório: https://github.com/felipe1496/lsi-analytics

1. INTRODUÇÃO

Em um mundo cada vez mais regido por sistemas de informação, a análise de dados se coloca na linha de frente para que empresas consigam tomar decisões estratégicas e serem bem sucedidas [1]. Tendo isso em vista, existem inúmeras soluções, pagas e gratuitas, para auxiliar nesse processo, simplificando as decisões com base em dados e a criação de insights. Algumas delas são, por exemplo, plataformas de Business Intelligence como *Metabase*[1], *Power BI*[2], *AWS Quick Sight*[3], *Apache Superset*[4], e *Tableau*[5]. No entanto, o uso de aplicações web de BI ainda pode ser um processo desafiador, exigindo conhecimentos técnicos específicos. Embora haja um grande número de plataformas, é ligeiramente difícil encontrar um ambiente de fácil uso, onde seja necessário nada mais que um navegador web, uma chave de acesso a uma base de dados, além da habilidade para ter insights gráficos.

É nesse contexto que surge a plataforma LSI Analytics, uma ferramenta que visa simplificar e democratizar o acesso à análise de dados, tornando-a acessível para profissionais desde gestores de negócios até analistas de mercado e cientistas de dados. A plataforma possui o objetivo de facilitar a vida do usuário de forma que, além de criar visualizações customizadas,

também consiga ter um *user experience*[6] que instigue-o a utilizar e recomendar a aplicação.

Este trabalho está organizado da seguinte forma: na primeira seção da solução proposta, abordamos os principais conceitos da plataforma e suas funcionalidades. Em seguida, descrevemos detalhadamente como foi feito o frontend, backend e banco de dados. Depois são incluídos os procedimentos de coleta e análise de dados para a avaliação da plataforma. Na quarta seção é apresentado a conclusão que foi possível atingir com a criação da plataforma.

2. SOLUÇÃO PROPOSTA

LSI Analytics é uma ferramenta de código aberto e gratuita que fornece interface gráfica para a produção de visualizações gráficas, numéricas e textuais de forma total ou filtrada. Na plataforma, é possível introduzir uma fonte de dados e, logo em seguida, começar a criar visualizações de forma intuitiva. A criação é dividida em 5 passos principais: inserção das configurações básicas de tipo da visualização, nome e atualização dos dados; escolha da fonte de dados; produção da query SQL que vai dar origem ao objeto de banco; criação dos estilos do gráfico; e por fim, o posicionamento no dashboard.

Dentre as principais funcionalidades do LSI Analytics destacam-se: criação de uma fonte de dados; criação de visualizações gráficas, tais como gráfico de pizza, gráfico de barra, gráfico de linha; geração de *KPIs*[7]; criação de filtros interativos associados às demais visualizações; e customização dos dashboards criados por meio do posicionamento e dimensionamento dos elementos no painel.

Na Figura 1, é apresentada a interface do usuário do LSI Analytics, demonstrando um conjunto de elementos que incluem um gráfico de barras, um gráfico de linha, uma visualização numérica e um filtro aplicável às visualizações gráficas mencionadas anteriormente. Além disso, observa-se a presença de um botão com uma engrenagem, o qual engloba uma série de

[1] https://www.metabase.com
[2] https://powerbi.microsoft.com
[3] https://aws.amazon.com/pt/pm/quicksight
[4] https://aws.amazon.com/pt/pm/quicksight
[5] https://www.tableau.com
[6] https://rockcontent.com/br/blog/user-experience/
[7] https://www.rdstation.com/blog/marketing/kpis/

ações relacionadas ao gráfico, tais como exclusão, edição e criação de novas visualizações.

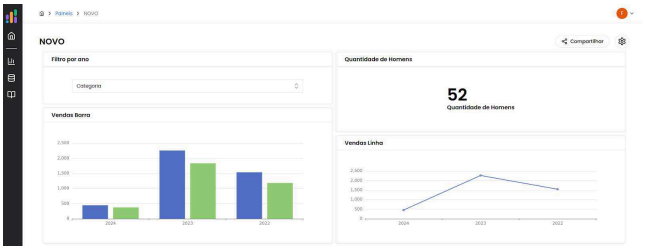

Figura 1: *Dashboard* contendo um Card de valor fixo da quantidade de homens em uma tabela de usuários, gráfico de barras e de linha das vendas de uma empresa, e um filtro de seleção por ano que se aplica ao gráfico de barra das vendas.

A Figura 2 compreende o painel administrativo do dashboard, onde o usuário pode fazer todo o gerenciamento da sua criação gráfica. Nesse painel é possível criar novas visualizações, editar o posicionamento das visualizações já existentes ou das que estão em processo de criação, e, por fim, alterar nome e descrição dos elementos. Com o objetivo de simplificar a utilização, a plataforma conta com um botão único de salvamento. Sendo assim, ao detectar mudanças em relação ao painel armazenado na base dados, a opção de salvar fica disponível para todo o painel de uma vez só, não sendo necessário clicar para salvar a cada alteração.

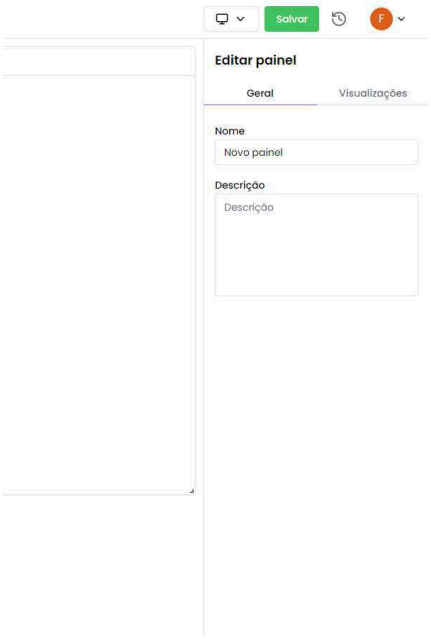

Figura 2: À direita, painel administrativo do *dashboard*.

Ao iniciar o processo de criação de uma visualização no painel de edição, conforme exemplificado na Figura 3, o usuário é conduzido por um fluxo sequencial. O primeiro estágio requer a inserção de informações iniciais, tais como nome da visualização, tipo desejado e método de atualização do conteúdo, conforme ilustrado na Figura 4. Posteriormente, o usuário é direcionado a selecionar uma das fontes de dados previamente cadastradas, conforme evidenciado na Figura 5. Em seguida, o usuário elabora a sua consulta ao banco de dados por meio de um script SQL, o qual será executado na fonte de dados selecionada, conforme delineado na Figura 6. Após essa etapa, o usuário procede à especificação das colunas a serem utilizadas como categorias e valores na visualização, e então posiciona o gráfico dentro da área designada por meio de uma interface de arrastar e soltar, conforme exemplificado na Figura 7 e 8. Por fim, o usuário conclui o processo de criação clicando no botão de salvar, o que resulta na persistência da visualização.

Figura 3: Menu de visualizações do *dashboard*.

Figura 4: Formulário de configuração inicial da nova visualização.

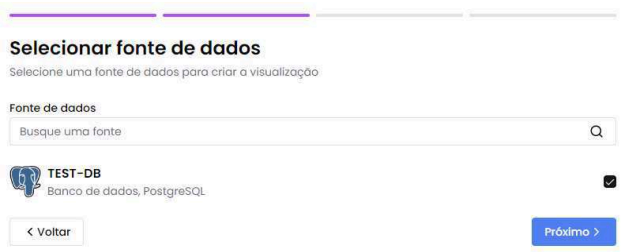

Figura 5: Seleção de fonte de dados.

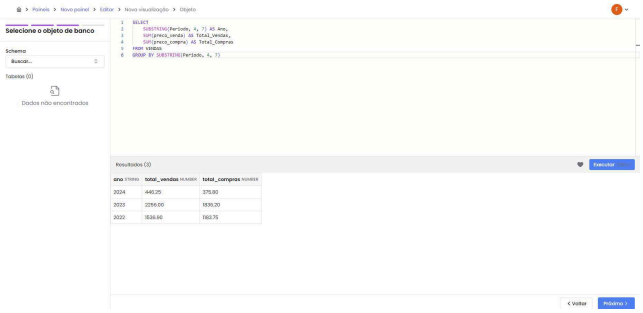

Figura 6: Editor de SQL na fonte de dados do usuário.

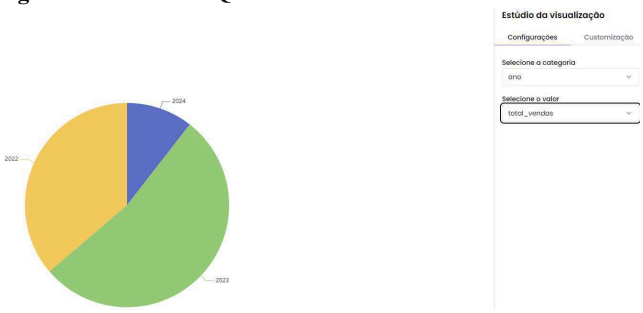

Figura 7: Estúdio de criação do gráfico.

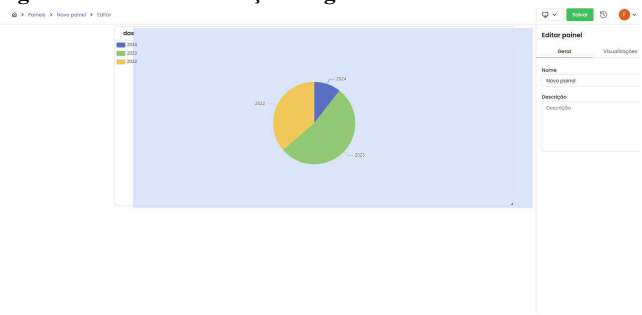

Figura 8: Grade de posicionamento com arrasta e solta do *dashboard*.

A estrutura do LSI Analytics tem como base uma arquitetura REST[8], dividido em frontend e backend juntamente com um banco de dados, mas também consumindo fontes de dados adicionadas por usuários da aplicação. A seguir são detalhados os aspectos de arquitetura e tecnologias utilizadas.

[8] https://pt.wikipedia.org/wiki/REST

2.1 Frontend

O módulo de interface de usuário (frontend) do sistema é implementado utilizando a biblioteca React, uma escolha popular dentro do espectro de desenvolvimento de interfaces ricas em aplicações web [3]. Esta biblioteca facilita o gerenciamento de estados da aplicação por meio de ganchos (hooks), especificamente através do uso do *useState*[9], além de adotar o padrão *Provider*[10] para a distribuição de estados entre componentes. O React emprega uma abordagem de renderização que se apoia em uma estrutura hierárquica de componentes, um paradigma amplamente adotado pelas ferramentas contemporâneas destinadas ao desenvolvimento de frontends.

Outro grande diferencial que incentivou o uso da biblioteca React foi a sua implementação de renderização e atualização de interfaces do Virtual DOM. Este mecanismo minimiza a quantidade de manipulações diretas no DOM real, otimizando o desempenho e garantindo a atualização da interface de usuário de forma ágil e com o mínimo de interrupções percebidas pelo usuário [2].

No contexto da arquitetura do LSI Analytics, a interação com o estado interno da aplicação e o preenchimento de dados na interface são feitos através de comunicação com o servidor backend. Essa interação é intermediada pela biblioteca *axios*[11], uma solução amplamente adotada para a realização de solicitações HTTP de forma assíncrona. Através de eventos gerados pelas interações do usuário com a interface, solicitações são enviadas para a API do sistema backend, cujas respostas fornecem os dados necessários para atualizar os estados mantidos no frontend. Esse fluxo de dados não apenas facilita a separação de responsabilidades entre a lógica de apresentação e a lógica de negócios, mas também promove uma arquitetura escalável e manutenível, na qual a interface do usuário permanece consistentemente sincronizada com as informações mais recentes disponibilizadas pelo sistema. Esse fluxo é evidenciado pela Figura 9.

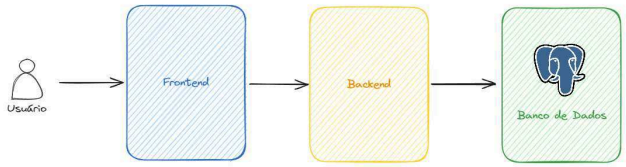

Figura 9: Estrutura da comunicação dos subsistemas que compõem o LSI Analytics.

2.1.1 Arquitetura dos componentes gráficos

O pilar da interface do usuário da plataforma se ergue principalmente sobre os gráficos gerados por meio da biblioteca Apache *ECharts*[12]. Este recurso é essencial no processo de visualização de dados, facilitando a interpretação e análise de informações complexas de maneira intuitiva. A integração entre os dados provenientes das consultas SQL específicas a cada gráfico e a interface de usuário é feita de forma que simplifique o

[9] https://react.dev/reference/react/useState
[10] https://www.patterns.dev/vanilla/provider-pattern/
[11] https://axios-http.com/ptbr/docs/intro
[12] https://echarts.apache.org/en/index.html

processo de transformação dos dados para a interface aceita pelos componentes de gráfico do Echarts.

Essa integração é orquestrada por uma classe especializada, desenvolvida seguindo o padrão de design *Adapter*[13], sua implementação está ilustrada na Figura 10. O padrão Adapter é essencial para reconciliar interfaces incompatíveis, permitindo que sistemas independentes comuniquem-se sem a necessidade de alterar diretamente seu código-fonte [4]. No contexto em questão, a classe adaptadora desempenha um papel crucial ao converter os resultados das consultas SQL em um formato compatível com os requisitos de entrada dos componentes gráficos fornecidos pelo ECharts. Esse processo de adaptação assegura que os dados possam ser eficientemente incorporados e visualizados dentro da interface de usuário.

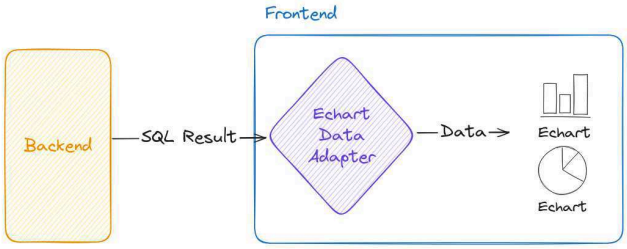

Figura 10: Arquitetura do fluxo de transformação de dados resultado sql para dados gráficos.

Através da integração desses mecanismos, a plataforma é capaz de apresentar dados dinâmicos, aprimorando significativamente a capacidade do usuário de compreender tendências e padrões derivados dos dados. A partir dessa utilização conjunta obteve-se uma abordagem modular e flexível do design de sistemas, proporcionando a possibilidade de uma extensão facilitada para adição de novas bibliotecas de gráficos no futuro.

2.1.2 Usabilidade da aplicação

Durante a fase de desenvolvimento de aplicações web utilizando a biblioteca React, um dos aspectos mais significativos está na capacidade de criar componentes que podem ser reutilizados dentro da mesma aplicação, bem como por outras aplicações através da criação de um pacote NPM[14] para os componentes [5]. Nesse contexto, a seleção de uma biblioteca de componentes de código aberto se torna essencial pela busca de aproveitar o trabalho de comunidades que dedicaram atenção a diversos elementos, como acessibilidade e experiência de desenvolvimento.

Dito isso, foi essencial a escolha de bibliotecas NPM relacionadas a interface. Dessa forma, é inegável que a experiência do usuário foi uma das preocupações principais no desenvolvimento do LSI Analytics. Assegurar a acessibilidade e usabilidade para todos os usuários foi um critério mandatório no desenvolvimento da ferramenta.

Assim, a escolha do auxiliar de construção de biblioteca de componentes ShadCN[15] para a implementação da aplicação foi muito importante, visto que esta biblioteca oferece componentes esteticamente agradáveis, alinhados à estética minimalista que tem ganhado popularidade [6]. Além disso, destacam-se os recursos de acessibilidade, os quais são embasados em outro pacote de componentes acessíveis, o RadixUI[16].

O RadixUI é conhecido por sua abordagem de baixo nível, focada em proporcionar aos desenvolvedores um conjunto de componentes não estilizados e totalmente acessíveis, o que significa que oferece a estrutura e a lógica de comportamento dos componentes sem impor estilos específicos. Isso permite uma grande flexibilidade e liberdade na personalização visual, possibilitando que a plataforma de geração de gráficos mantenha uma identidade visual única e adaptada às necessidades específicas dos usuários finais [7].

Além disso, a utilização do ShadCN como uma camada sobre o RadixUI permite tirar vantagem de suas funcionalidades avançadas de estilização e composição de componentes, facilitando a criação de interfaces ricas e interativas com menos código. Isso não apenas acelera o processo de desenvolvimento, mas também resulta em uma base de código mais limpa, fácil de manter e de escalar. A integração dessas tecnologias significa que a aplicação pode oferecer uma experiência de usuário responsiva e acessível, crucial para ferramentas de visualização de dados, onde a clareza e a precisão da apresentação são fundamentais [8].

Finalmente, a escolha dessas tecnologias reflete uma tendência contemporânea no desenvolvimento de aplicações web, onde a modularidade [9], a personalização e a acessibilidade [10] são requisitos essenciais. Ao adotar o ShadCN e o RadixUI, o projeto está alinhado com práticas de desenvolvimento web modernas, demonstrando um compromisso com a qualidade, a inovação e a experiência do usuário. Isso não apenas beneficia os usuários finais, mas também estabelece um fundamento técnico para o futuro crescimento e evolução da plataforma.

2.2 Backend

Para o desenvolvimento do backend, optou-se pela adoção da plataforma NodeJS[17], amplamente reconhecida pela sua notável capacidade de facilitar a criação de aplicativos de alta performance, sendo uma das plataformas mais populares do ano de 2023 [11]. Esse web framework utiliza um padrão baseado em divisão de responsabilidades como demonstrado na Figura 11.

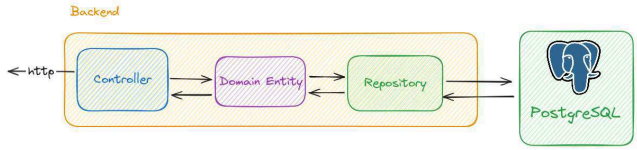

Figura 11: Arquitetura do fluxo do backend.

Dessa forma, a fim de promover a reutilização de código e simplificar o processo de configuração inicial de um serviço backend, além da criação de endpoints, optou-se deliberadamente

[13] https://refactoring.guru/design-patterns/adapter
[14] https://www.npmjs.com/
[15] https://ui.shadcn.com/
[16] https://www.radix-ui.com/
[17] https://nodejs.org/en

pela seleção de um framework que opera conjuntamente com Node.js. Neste contexto, a escolha foi o NestJS[18], um framework caracterizado por sua alta performance e por oferecer uma experiência de desenvolvimento boa.

O NestJS destaca-se por sua habilidade em estabelecer uma interface unificada para a integração com diversos outros frameworks construídos sobre o Node.js. Para este projeto específico, foi adotado o ExpressJS, um framework web para construção de APIs REST, sendo este mínimo e flexível, conhecido por fornecer um conjunto robusto de funcionalidades essenciais para a implementação deste padrão. Apesar disso, conforme dito, o NestJS tem a capacidade de criar uma interface única sobre outros frameworks, isso significa que, do ponto de vista do desenvolvedor, a programação é realizada como se o código fosse diretamente destinado ao ExpressJS ou a qualquer outro framework selecionado para operar nos bastidores [12].

Essa abordagem, adotada pelo NestJS, não apenas facilita o desenvolvimento e a manutenção de aplicações ao abstrair complexidades específicas de diversos frameworks, mas também potencializa a modularidade e a escalabilidade das aplicações. Ao proporcionar uma camada de abstração unificada, o NestJS permite que desenvolvedores concentrem-se na lógica de negócios e na criação de funcionalidades, sem a necessidade de aprofundar-se excessivamente nos detalhes técnicos de implementação de baixo nível. Esta metodologia contribui significativamente para a eficiência do desenvolvimento e para a qualidade do produto final, reforçando a escolha do NestJS como uma solução estratégica para o desenvolvimento de aplicações sobre a plataforma Node.js.

2.2.1 Injeção de dependências

A injeção de dependência é um padrão de design fundamental no desenvolvimento de software, crucial para alcançar a inversão de controle entre classes e suas dependências. O NestJS incorpora esse padrão de maneira elegante e eficaz, trazendo múltiplos benefícios para o desenvolvimento de aplicações, especialmente no que tange à realização de testes.

```
@Injectable()
export class PrismaPanelsRepository implements PanelsRepository {
  constructor(private prisma: PrismaService) {}

  public async create(props: CreatePanelProps) {
    const panel = await this.prisma.panel.create({
      data: {
```

Figura 12: Exemplo de classe injetável no NestJS.

Como podemos ver na Figura 12, é possível fazer com que uma determinada classe seja injetável, a partir desse momento é possível utilizá-la a partir do construtor de outras classe, mas antes, devemos dizer qual classe abstrata deve corresponder à instância da classe injetável. Para isso, utiliza-se o código descrito na Figura 13.

```
@Module({
  controllers: [PanelsController],
  providers: [
    {
      provide: PanelsRepository,
      useClass: PrismaPanelsRepository,
    },
```

Figura 13: Exemplo do uso de conexão entre uso de uma classe abstrata para uma classe implementação.

Dessa forma, sempre que a classe PanelsRepository for utilizada, por padrão será utilizada uma instância da classe PrismaPanelsRepository. A utilização desse Repository está descrito na Figura 14.

```
@Controller('/panels')
export class PanelsController {
  constructor(
    private panelsRepository: PanelsRepository,
    private prisma: PrismaService,
    private datafontRepository: DataFontsRepository,
    private viewsRepository: ViewsRepository,
  ) {}

  @Post()
  @UseGuards(AuthGuard)
  @HttpCode(HttpStatus.CREATED)
  public async create(
    @Req() request: Request,
    @Body() createPanelDto: CreatePanelDto,
  ) {
    const userId = request.userId;
    const panel = await this.panelsRepository.create({
      ...createPanelDto,
      userId,
    });

    return PanelsMapper.toHTTP(panel);
  }
```

Figura 14: Utilização prática da classe injetável em um Controller.

2.2.2 Banco de dados

É fato que na maioria das aplicações web se faz necessário a persistência dos dados através de uma base. No LSI Analytics existem inúmeras informações relacionadas à persistência que precisam ser guardadas em uma base para melhorar a experiência dos usuários. Dito isso, para conceber essa funcionalidade utilizou-se o banco de dados PostgreSQL[19], um banco de dados escolhido principalmente por ser um dos principais SGBDs relacionais que utilizam a filosofia ACID (Atomicidade, Consistência, Isolamento e Durabilidade) [13] gratuitos.

2.2.3 Adaptador para do banco de dados para o código

Ferramentas de Mapeamento Objeto-Relacional (ORM, do inglês Object–Relational Mapping) desempenham papel essencial ao facilitar a interação entre o código da API e o banco de dados. Tais soluções permitem estabelecer conexões, realizar consultas e efetuar a transformação das informações armazenadas no banco de dados em objetos de domínio específicos. Essa capacidade de transformação é crucial, pois viabiliza que a API administre as

[18] https://nestjs.com/

[19] https://www.postgresql.org/

regras de negócio de maneira mais eficiente e alinhada às necessidades específicas do sistema [14].

Neste cenário, destaca-se o Prisma[20], um dos ORMs mais populares e eficientes atualmente disponíveis. Esse ORM é notável por sua capacidade de converter as informações do banco de dados em objetos que podem ser utilizados pelo NodeJS de maneira excepcional. Além disso, o Prisma oferece a criação de tipagem para esses objetos, o que enriquece significativamente a experiência do desenvolvedor, tornando o processo de codificação para a API mais agradável e produtivo.

O Prisma provê agilidade e produtividade na construção da API, graças à sua abordagem singular para definir os objetos do banco de dados, sem a necessidade direta de escrever scripts SQL para a criação, atualização e remoção de tabelas. A partir dessa biblioteca é escrito um model com uma sintaxe única, e a partir disso o prisma gerencia todos os scripts de migração SQL para o versionamento do banco de dados conforme mostrado na Figura 15.

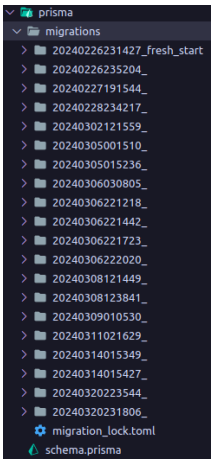

Figura 15: pasta contendo as migrações criadas e gerenciadas pelo prisma.

2.3 Deployment da Aplicação

Com o objetivo de disponibilizar a aplicação para o público foram utilizadas algumas ferramentas de deploy em cloud. As plataformas utilizadas foram cuidadosamente escolhidas com planos gratuitos e tendo em vista a alta disponibilidade.

2.3.1 Frontend - Vercel

A implantação do frontend do LSI Analytics foi realizada por meio da Vercel, uma das plataformas mais reconhecidas para o desenvolvimento de aplicações frontend baseadas em NodeJS. Essencialmente, a Vercel opera utilizando os serviços da AWS, contudo, oferece uma interface significativamente mais intuitiva e uma experiência de usuário superior àquela proporcionada pela Amazon.

O procedimento de deployment na Vercel é simples e direto. Inicialmente, seleciona-se o repositório Git destinado à implantação, como ilustrado na Figura 16. Posteriormente, procede-se ao registro das configurações, que incluem o nome do projeto, o framework frontend empregado e a pasta raiz. Com a simples ação de clicar no botão “Deploy”, ilustrado pela Figura 17, a aplicação é lançada e torna-se acessível online em questão de segundos. Esse processo, além de ser simplificado, é ágil, facilitando consideravelmente a publicação de aplicações frontend.

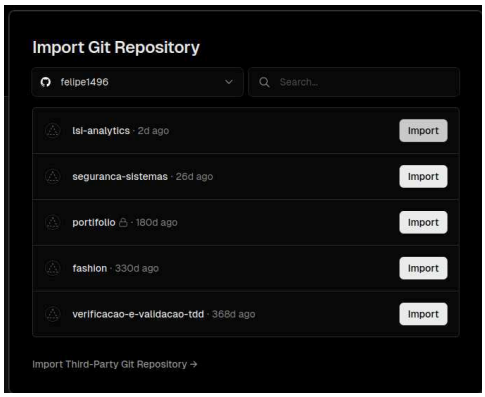

Figura 16: Seleção de repositório para deploy na Vercel.

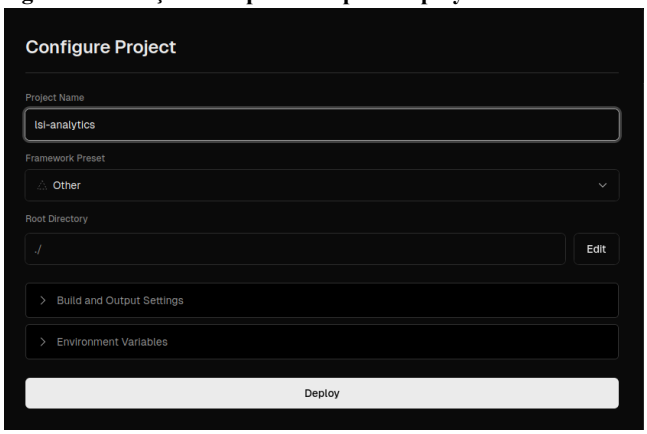

Figura 17: Painel de configurações pré deploy na Vercel.

2.3.2 Backend - Render

Para a implantação do backend, optou-se pela utilização do Render[21], uma plataforma robusta e eficiente, ideal para aplicações backend que exigem alta disponibilidade e desempenho. Render se destaca por sua facilidade de uso e pela eficiência no gerenciamento de infraestruturas complexas, o que permite aos desenvolvedores concentrarem-se mais no desenvolvimento do código do que na gestão do ambiente operacional.

O processo de deployment no Render é intuitivo e eficaz. Inicia-se com a seleção do repositório Git que contém o código fonte do backend, como demonstrado na Figura 18, seguido pela configuração das variáveis de ambiente e especificações técnicas necessárias para o correto funcionamento da aplicação. Tais etapas são fundamentais para garantir que o ambiente de produção esteja alinhado com as necessidades específicas do projeto.

[20] https://www.prisma.io/

[21] https://dashboard.render.com/

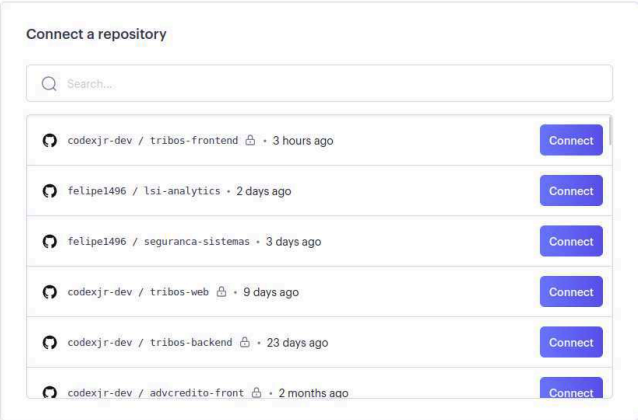

Figura 18: Seleção de repositório de deployment no Render

2.3.3 Banco de dados - Supabase

Foi utilizado para o banco de dados a plataforma Supabase[22], uma plataforma moderna e eficaz que simplifica o uso de PostgreSQL para desenvolvedores. O processo de configuração inicia-se com a criação de uma nova instância de banco de dados. Isso envolve definir configurações iniciais, como a versão do PostgreSQL, a configuração de região e o tamanho da instância, adequando-se assim às exigências específicas do projeto, como ilustrado pela Figura 19. Além disso, a plataforma permite a importação fácil de dados existentes e a configuração de regras de segurança para proteger as informações sensíveis.

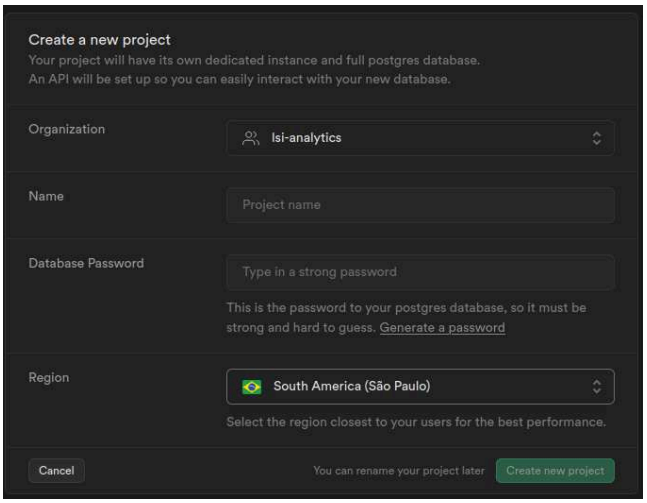

Figura 19: Configuração da instância do banco no Supabase

Após a configuração inicial, o Supabase oferece uma série de recursos automatizados, como backups regulares e monitoramento de desempenho, que garantem a integridade e a disponibilidade dos dados. Com um simples clique no botão "Launch", o banco de dados é ativado e pronto para ser integrado ao backend da aplicação.

3. AVALIAÇÃO DA FERRAMENTA

Para avaliar e analisar a eficácia do LSI Analytics, um estudo foi conduzido envolvendo 15 participantes, os quais responderam a um questionário desenvolvido para esse fim. O questionário foi adaptado a partir do *Post-Study System Usability Questionnaire*[23] (PSSUQ), versão 3, com modificações para garantir a relevância das perguntas em relação ao contexto específico do LSI Analytics.

O propósito principal desse questionário é coletar métricas sobre a satisfação dos usuários com a ferramenta. Ele consiste em oito afirmações, nas quais os participantes são solicitados a classificar suas experiências ao utilizar o LSI Analytics. As afirmações incluídas no questionário foram as seguintes:

(1) Eu possuo conhecimentos de BI (Business Intelligence).

(2) O sistema foi simples de utilizar.

(3) O sistema possui todas as funcionalidades e capacidades que eu esperava.

(4) Eu fui capaz de completar as tarefas e cenários rapidamente utilizando o sistema.

(5) O sistema me deu mensagens de erro que me disseram claramente como resolver os problemas.

(6) A informação (como ajuda online, mensagens em tela e outras documentações) provida com o sistema foi clara.

(7) A organização da informação nas telas foi clara.

(8) Você recomendaria o uso do sistema a outras pessoas.

Cada uma das afirmações foi avaliada em uma escala de 1 a 5, na qual o valor 1 indica discordância total e o valor 5 indica concordância total. Os participantes foram solicitados a realizar duas atividades simples: criar um gráfico de torta e posicioná-lo no painel e criar um gráfico de linha e posicioná-lo no painel. Após essa fase inicial, os usuários foram livres para utilizar a ferramenta sem supervisão.

Analisando a Figura 20, que apresenta os resultados da Pergunta 1, observa-se que um pouco mais de 50% dos entrevistados afirmaram possuir conhecimento em técnicas de Business Intelligence (BI), indicando um equilíbrio satisfatório entre os usuários familiarizados com outras ferramentas e acerca do cerne da plataforma.

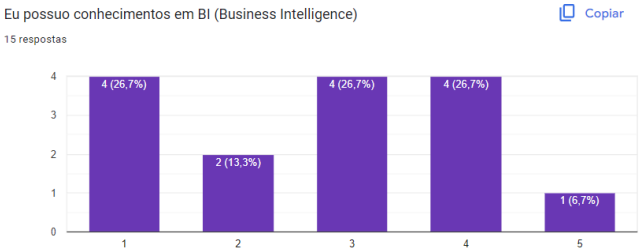

Figura 20: Avaliações da Pergunta "Eu possuo conhecimentos em BI (Business Intelligence)"

[22] https://supabase.com/

[23] https://uiuxtrend.com/pssuq-post-study-system-usability-questionnaire/

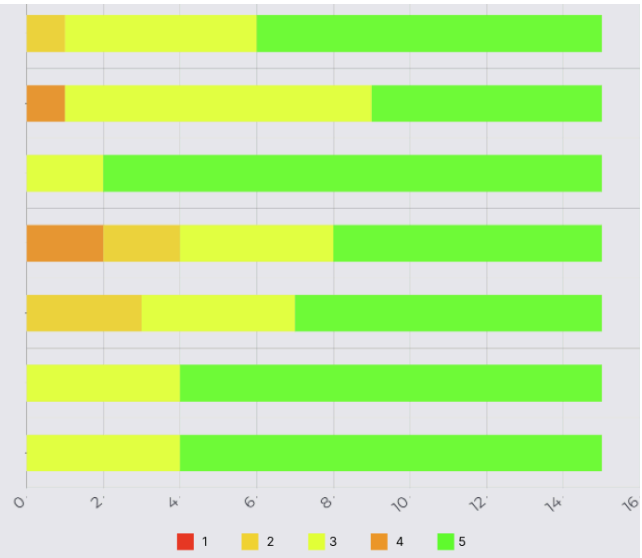

Figura 21: Respostas do Questionário de Avaliação do LSI Analytics

Os demais resultados da avaliação podem ser visualizados no gráfico presente na Figura 21, onde o eixo vertical representa as perguntas na ordem em que elas foram feitas como evidenciado no início da sessão de avaliação e o eixo horizontal representa a pontuação de 1 a 5 dada pelos usuários. Cerca de 85% das avaliações de cada afirmação tiveram pontuação igual à 5, o que representa um ótimo resultado.

4. CONCLUSÃO

No contexto do LSI Analytics, uma plataforma de Business Intelligence (BI), destaca-se a crescente necessidade por ferramentas que habilitam a análise de dados e a geração de gráficos a partir de fontes de dados. Este estudo apresenta uma solução que simplifica o processo de análise de dados, permitindo aos usuários criar visualizações gráficas de forma intuitiva e eficiente.

A abordagem proposta é baseada em um ambiente de código aberto, oferecendo aos usuários a flexibilidade necessária para explorar e visualizar dados de diferentes fontes. Além disso, a plataforma utiliza uma interface amigável.

Um dos desafios enfrentados foi garantir a renderização dinâmica e a manipulação eficiente de grandes conjuntos de dados, garantindo que os usuários pudessem interagir de forma fluida com as visualizações geradas. Para superar esse obstáculo, foram adotadas técnicas avançadas de otimização de desempenho e visualização de dados.

Como perspectivas futuras, planeja-se expandir ainda mais as capacidades da plataforma, incluindo recursos avançados de análise de dados, integração com mais fontes de dados e suporte à visualizações interativas. Isso permitirá aos usuários explorar e entender melhor seus dados, tomando decisões mais informadas e estratégicas.

REFERENCIAS

[1] ROMERO, Carlos Andrés Taver. Business Intelligence: Business Evolution after Industry 4.0. Digital Revolution in Sustainable Business Models and Finance Management. 2021. Disponível em: https://www.mdpi.com/2071-1050/13/18/10026.

[2] O'rinboev, A. 2023. ANALYZING THE EFFICIENCY AND PERFORMANCE OPTIMIZATION TECHNIQUES OF REACT.JS IN MODERN WEB DEVELOPMENT. Innovative research in the modern world: theory and practice. 2, 24 (Sep. 2023), 54–57.

[3] Stateofjs. Frameworks de Front-end. 2023. Disponível em: https://2022.stateofjs.com/pt-BR/libraries/front-end-frameworks/

[4] Alexander Shvets. 2023. Adapter.Disponível em: https://refactoring.guru/design-patterns/adapter

[5] Meta. 2024. Componentes e Props. Disponível em: https://pt-br.legacy.reactjs.org/docs/components-and-props.html

[6] Ulrik Soderstrom, Lovisa Carlsson, Thomas Mejtoft. 2019. Comparing Millennials View on Minimalism And Maximalism in Web Design. Disponível em: https://dl.acm.org/doi/abs/10.1145/3335082.3335104

[7] RadixUI. 2024. Primitives. Disponível Em: https://www.radix-ui.com/primitives/docs/overview/introduction

[8] shadcn. 2024. Build your component library. Disponível em: https://ui.shadcn.com/

[9] Viacheslav Petrenko. 2024. Understanding Contemporary Web Application Architecture: Key Components, Best Practices, and Beyond. Disponível em: https://litslink.com/blog/web-application-architecture

[10] Nádia Fernandes, Daniel Costa, Carlos Duarte, Luís Carriço. 2012. Disponível em: https://www.sciencedirect.com/science/article/pii/S1877050912007661

[11] Martin K. 2023 . Node.js for Backend Web Development in 2023. Disponível em: https://dzone.com/articles/nodejs-for-backend-web-development

[12] NestJS. 2024. Documentation. Disponível em: https://docs.nestjs.com/

[13] Nishtha Jatana, Sahil Puri, Mehak Ahuja, Ishita Kathuria, Dishant Gosain. 2012. Disponível em: https://www.ijert.org/research/a-survey-and-comparison-of-relational-and-non-relational-database-IJERTV1IS6024.pdf

[14] E. F. Codd. 2007. Relational database: a practical foundation for productivity. Disponível em: https://dl.acm.org/doi/abs/10.1145/1283920.1283937