
UNIVERSIDADE FEDERAL DE PERNAMBUCO
CENTRO DE INFORMÁTICA
GRADUAÇÃO EM ENGENHARIA DA COMPUTAÇÃO

# Uma ferramenta para visualização de dados referentes à evasão no âmbito dos cursos de graduação do Centro de Informática - UFPE

Trabalho de Graduação

Aluno: Alexandre Victor de Araújo Oliveira
Orientador: Márcio Cornélio Lopes
Área: Engenharia de Software

RECIFE
2022

Universidade Federal de Pernambuco

Centro de Informática

Alexandre Victor de Araújo Oliveira

# Uma ferramenta para visualização de dados referentes à evasão no âmbito dos cursos de graduação do Centro de Informática - UFPE

*Trabalho de Conclusão de Curso apresentado no curso de Bacharelado em Engenharia da Computação do Centro de Informática da Universidade Federal de Pernambuco como requisito parcial para obtenção do grau de Bacharel em Engenharia da Computação.*

Orientador: Márcio Cornélio Lopes

Recife
2022

*Este trabalho é dedicado aos meus Pais, Professores, Família e Amigos que sempre estiveram comigo e fizeram o melhor que podiam para me ajudar nos momentos em que mais precisei.*

## AGRADECIMENTOS

Agradeço primeiramente a Deus, por me edificar e fortalecer ao longo da minha jornada, seu amor me glorifica todos os dias.

Agradeço também ao Centro de Informática da Universidade Federal de Pernambuco (CIn-UFPE), por ser uma segunda casa desde 2014 e que possibilitou minha formação como engenheiro da computação.

Aos meus pais, Conceição Figueiredo de Araújo Oliveira e Alexandre Jorge Mendes de Oliveira pelo apoio de sempre, e por todo amor, carinho e a base necessária para concluir essa jornada. Quem eu sou hoje é resultado da educação fornecida por ambos, aos quais serei eternamente grato e sempre amarei.

À minha irmã Amanda, por sempre estar ao meu lado nos momentos de dificuldade e sempre me amar apesar das desavenças de irmãos.

À minha namorada Cecília pela força e compreensão durante nossa caminhada, assim como o amor que nunca esperei receber. Nunca me tornaria metade do homem que sou hoje sem estar ao seu lado.

À Maria José que me criou desde a infância e será para sempre o símbolo de uma segunda mãe para mim.

Aos meus avós, Jobson e Lúcia que tanto me deram amor e apoio em toda minha vida.

Aos meus padrinhos, Flávio e Glória, assim como meu tio Ricardo, pelos conselhos de vida e carinho.

Aos meus tutores de tecnologia, Hélio Meira Lins e Ricardo Mariz, com os quais sempre pude contar para agregar conhecimento e responder a dúvidas pertinentes na minha área de atuação.

Ao meu orientador, Professor Márcio Lopes Cornélio, com quem tive a oportunidade de agregar bastante conhecimento e atingir novos objetivos através de sua bondade e criatividade, sempre disposto a ajudar o próximo. Sua dedicação e atenção foram fundamentais para a realização desse trabalho.

Por fim, a toda minha família e amigos que estiveram ao meu lado quando mais precisei. Todos foram fundamentais para que eu pudesse trilhar meu caminho com sabedoria e persistência. Por esse motivo almejo todos os dias retribuir da mesma forma.

## RESUMO

O sistema educacional brasileiro apresenta índices de evasão que chamam atenção, especificamente nos cursos de graduação do ensino superior, em que há o emprego de grande parte dos recursos públicos. Esse fenômeno pode ser evidenciado a partir de dados, que por sua vez possuem utilidade na elaboração de políticas públicas ou privadas de melhorias na gestão acadêmica. Todavia, esses dados geralmente são expostos em planilhas e sua funcionalidade fica limitada, razão pela qual esse trabalho propõe a construção de uma ferramenta que requisita arquivos no formato *.csv* para a seguir disponibilizar as informações vigentes em gráficos, garantindo uma melhor visualização das informações. A metodologia de desenvolvimento adotada envolve a utilização do *Next.js* para cumprir o papel geral da construção do software, em conjunto com algumas bibliotecas e ferramentas da linguagem de programação *JavaScript* e com o emprego de uma função encarregada de calcular, no presente trabalho, a evasão no âmbito dos três cursos de graduação do Centro de Informática (CIn) da UFPE. Esse procedimento possibilitou a entrega de gráficos contendo dados da taxa de evasão dos referidos cursos por período de ingresso-período de evasão, demonstrando quais são os intervalos mais sensíveis à ocorrência do evento estudado.



**Palavras-chave:** Evasão; graduação; cálculo; software; *Next.js*.

## ABSTRACT

The Brazilian educational system has dropout rates that draw attention, specifically in higher education undergraduate courses, where a large part of public resources is used. This phenomenon can be evidenced from data, which in turn are useful in the elaboration of public or private policies for improvements in academic management. However, this data is usually exposed in spreadsheets and its functionality is limited, which is why this work proposes the construction of a tool that requests files in *.csv* format to then make the current information available in graphics, ensuring a better visualization of the information. The development methodology adopted involves the use of *Next.js* to fulfill the general role of building the software, together with some libraries and tools of the *JavaScript* programming language and with the use of a function in charge of calculating, in the present work, the dropout in the scope of the three graduation courses of the Centro de Informática (CIn) of the UFPE. This procedure enabled the delivery of graphs containing data on the dropout rate of the referred courses by admission period-dropout period, demonstrating which are the most sensitive intervals to the occurrence of the event studied.



**Keywords**: Evasion; graduation; calculation; software; *Next.js*.

**LISTA DE FIGURAS**

**SUMÁRIO**

# 1.INTRODUÇÃO

Milhares de alunos ingressam nas universidades públicas brasileiras todos os anos, porém uma parcela desses universitários não chegam a concluir o curso de graduação. Segundo uma projeção feita pelo Semesp, a taxa de evasão de universidades privadas no Brasil aumentou durante o período da pandemia [LÜDER, 2022], apontando uma possível relação entre esse evento e uma acentuação das causas que levam à evasão.

A educação é direito de todos e dever do Estado, conforme dispõe o art. 205 da Constituição Federal [BRASIL, 1988], porém nem sempre a gestão da verba destinada a essa política pública é distribuída e executada da maneira mais eficiente. Dados fornecidos pelo portal da transparência do governo brasileiro referente ao ano de 2021 apontam que os investimentos feitos em ensino superior no Brasil superam a educação básica [CONTROLADORIA-GERAL DA UNIÃO], o que vai de encontro à própria obrigatoriedade da universalização do ensino básico, conforme prevê a Lei de Diretrizes e Bases da Educação (Lei 9.394/96), objetivo que ainda não se efetivou de forma satisfatória.

Fatores como dificuldades financeiras, necessidade de conciliação com o trabalho, maternidade e responsabilidade com filhos menores, escolha do curso em idade precoce, distância entre o domicílio e a universidade, falta de identificação com o curso, dificuldades com notas, entre outros, são hipóteses de motivos que podem levar os indivíduos a abandonarem a tão sonhada vaga na universidade.

Além disso, cabe pontuar que outras despesas indiretas são fundamentais para a consecução das atividades-fim de ensino pesquisa e extensão nas universidades, como auxílios-alimentação, bolsas de fomento à pesquisa científica, auxílio para o custeio de residência estudantil, bolsa-xérox, auxílio-transporte, entre outro, trazendo maiores desafios para a efetivação de uma política pública educacional de qualidade.

Nesse sentido, Rocha e outros destacam, através de estudo qualitativo, como sendo o perfil de alunos do curso de engenharia da Universidade Federal do Ceará com tendência à evasão:

> estado civil solteiro, gênero feminino, idade entre 26 e 30 anos, natural do Ceará, que mora sozinho, cujos pais possuem escolaridade até o ensino fundamental e renda familiar entre R$ 2.000,00 e R$ 4.000,00, são alunos provenientes de mudança de

> curso, com deficiência de aprendizado no ensino médio, não satisfeitos com a ação de nivelamento, não cotistas e beneficiários de políticas de assistência [ROCHA; LIMA; ANDRIOLA, 2020].

Como se vê, a temática da evasão universitária demanda uma investigação ampla acerca de suas prováveis causas, a fim de que esse fenômeno seja combatido a partir de um diagnóstico preciso, de modo que políticas públicas e administrativas sejam adotadas para combater suas causas e garantir uma melhoria nas taxas de permanência dos alunos, prestigiando o emprego de verbas públicas e possibilitando uma melhoria nos parâmetros educacionais do Brasil.

As abordagens sobre evasão são das mais diversas e podem ser empregadas em um contexto geral da educação ou até mesmo em níveis específicos, como educação básica, superior, curso a distância ou turmas específicas, assim como por categoria de setor de investimento, ou seja, rede pública ou privada.

Nesse trabalho, o recorte proposto será o cálculo de evasão no âmbito dos cursos de graduação do Centro de Informática da Universidade Federal de Pernambuco, cuja funcionalidade será fornecer visualização de dados que possam servir para investigação a posteriori sobre causas.

O problema de pesquisa a ser respondido é: a ferramenta desenvolvida e proposta nesse trabalho mostra-se capaz de possibilitar uma visualização dos dados a partir dos cálculos de evasão?

Para alcançar o objetivo proposto, desenvolvemos uma ferramenta para facilitar a visualização dos dados da evasão no âmbito dos cursos de graduação do Centro de Informática. O método adotado envolverá o emprego das técnicas de Engenharia de Software, de forma que os profissionais que trabalhem com esses dados possam alcançar um diagnóstico mais detalhado e preciso das problemáticas que permeiam a evasão no campo estudado.

O software desenvolvido foi utilizado para a obtenção do cálculo das taxas de evasão dos cursos de graduação do Centro de Informática, fornecendo dados para subsidiar estudos

posteriores que investiguem as motivações que desencadeiam essas evasões, de extrema importância para que seja possível gerar propostas com vistas à mitigação da evasão.

## 2.BREVE REVISÃO DA LITERATURA SOBRE O TEMA DA EVASÃO

Nessa seção serão analisadas diferentes perspectivas em torno da evasão a fim de referenciá-las no contexto da geração de gráficos para análise dos dados. A importância de contextualizar o tema na literatura reside no fato de que a ferramenta precisa de dados da realidade para os cálculos. Assim, antes de construir a ferramenta é necessário conceituar evasão e as possíveis situações que a ocasionam.

O conceito de evasão pode variar a depender da abordagem que se propõe a problematizá-la. A definição aqui adotada coaduna-se com a proposta por Mussliner e outros, considerando sua ocorrência "quando de maneira formal ou informal o aluno abandona seus estudos, mesmo que o faça para ingressar em outro curso de sua preferência [MUSSLINER; MUSSLINER; MEZA; RODRÍGUEZ, 2021]".

Pesquisadores que buscaram possíveis causas sugerem que os anos iniciais da jornada acadêmica superior são considerados o período mais importante para retenção ou não dos alunos. Os cursos com maior índice de reprovação nos primeiros anos coincidem com aqueles com maiores índices de evasão [BRAGA; PEIXOTO; BOGUTCHI, 2003], apontando que os períodos iniciais dos cursos merecem uma atenção maior dos gestores acadêmicos. Para a pesquisadora Melina Klitzke, que também confirma esse dado, "o primeiro ano do estudante é crucial para a permanência dele. Se ele passa dessa fase, a taxa de evasão cai muito. Por isso, é importante desenvolver tutorias e monitorias que ajudem a sanar dificuldades que venham do ensino médio [MENEZES; MAIA, 2022]".

Em dissertação de mestrado na área de ciências econômicas defendida no ano de 2019, Chagas aponta em seus achados que "a motivação para a desistência muitas vezes pode estar relacionada a motivos não acadêmicos e que normalmente ocorrem no começo do curso, tal como, a evasão voluntária [CHAGAS, 2019]", bem como que é possível inferir em relação às áreas de conhecimento que cursos de exatas têm maiores chances de evasão em relação aos de humanas e ciências da vida, além do efeito das reprovações no agravamento das chances de

evasão, razão que leva o autor a sugerir "a importância de políticas endereçadas a questões acadêmicas para alunos recém ingressados", entre outras propostas relevantes.

Em suas investigações, o autor chama a atenção para a importância de "explorar dados para prever tendências, descobrir padrões e extrair inteligência de dados de um determinado negócio, em outras palavras, transformar dados em informação [CHAGAS, 2019]", isso porque ele verifica que muitas vezes a gestão da universidade não consegue empregar meios para reverter a situação que ocasiona a evasão, a qual acaba sendo expressada pelo discente de forma tardia, impedindo a sua evitação. Por essa razão, cabe realizar um esforço científico no sentido de fornecer elementos que possam vir a contribuir com uma maior previsibilidade dessas causas.

Por sua vez, Mussliner e outros [MUSSLINER; MUSSLINER; MEZA; RODRÍGUEZ, 2021] concluem pela importância da atuação de uma equipe multidisciplinar na instituição de ensino superior (IES), integrada por psicólogo, pedagogo, assistente social, técnico em assuntos educacionais e assistente em administração, bem como pela condução de estudos e pesquisas sobre o tema no local de lotação da referida equipe, ante a constatação de que a evasão é causada por "múltiplos fatores de diferentes esferas [MUSSLINER; MUSSLINER; MEZA; RODRÍGUEZ, 2021]".

Ademais, ainda que não seja o objeto de pesquisa empregado na ferramenta a ser desenvolvida no presente trabalho, verifica-se interessante estudo de caso dos MOOCs (Massive Open Online Courses) [RÕÕM; LEPP; LUIK, 2021], os quais são cursos extensivos de qualidade com grande capacidade de fornecer conhecimento através da entrega de conteúdo pela Internet. As autoras do citado estudo também relatam que a baixa taxa de conclusão é um desafio significativo para os organizadores do MOOC, com taxas de abandono que chegam a cerca de 90% tendo como principal período de evasão o início de todos os cursos estudados.

Voltando ao objeto central da aplicação, para Silva Filho e outros, em artigo escrito no ano de 2007 sobre a evasão no ensino superior brasileiro, as áreas de Serviços e de Ciências, Matemática e Computação tiveram as mais altas taxas de evasão anual média, sendo maior nas IES privadas, cuja taxa média no período estudado correspondeu a 26%, contra 12% das IES públicas [FILHO; MONTEJUNAS; HIPÓLITO; LOBO, 2007].

Esse quadro fático parece se manter com inexpressivas alterações, conforme expõe relatório de pesquisa realizado em 2016 pela Universidade Federal de Pernambuco com dados do Sistema Integrado de Gestão Acadêmica (SIGA) e do Censo da Educação Superior (INEP), que apontou ser a maior taxa de evasão a da área básica de ingresso (ABI-Engenharia), com 10,5% de evadidos em relação aos vinculados, e que a análise da evasão por tempo de permanência no curso, em semestres, "revelam que 37,6% evadem antes do terceiro semestre após o ingresso, ou seja, no primeiro ano [PROPLAN-UFPE, 2016]".

Mais especificamente no Centro de Informática (CIn), as taxas de evasão apontadas pelo relatório da UFPE ficaram abaixo de 9%, sendo 8,56% no curso de Ciência da Computação, 8,28% no curso de Sistema da Informação e 7,51% no curso de Engenharia da Computação [PROPLAN-UFPE, 2016].

Por isso, o estudo e análise dos dados é de extrema importância para a fundamentação desse trabalho. O INEP tem divulgado publicamente, de forma regular, dados referentes aos matriculados, ingressantes e egressos do ensino superior, porém o Brasil ainda possui poucos dados oficiais sobre a evasão.

Portanto, é a partir desse contexto de precariedade na sistematização de dados sobre evasão que a criação de uma ferramenta possibilitará aos pesquisadores a continuidade na elaboração de hipóteses de pesquisa para formulação de possíveis soluções para o problema.

## 3.LINGUAGENS, BIBLIOTECAS E FERRAMENTAS

Para o desenvolvimento do projeto foi utilizado o *Next.js*, um *framework* do *React* que fornece blocos de construção para criar aplicativos Web. Quanto a *framework*, queremos dizer que o *Next.js* lida com as ferramentas e configurações necessárias para o *React* e fornece estrutura, recursos e otimizações adicionais para a aplicação. Sua abordagem híbrida e única consegue unir desenvolvimento *front-end*, *back-end* e infraestrutura em uma única estrutura.

Já o *React* é uma biblioteca *front-end* que possui como um de seus objetivos prover facilidade de conexão entre diferentes partes de uma página e utiliza o que chamamos de *componentes* para reaproveitamento de código e padronização de interface. Desta maneira o

*React* mostrou ser uma tecnologia bastante flexível para solucionar problemas e construir interfaces reutilizáveis, visto que podemos utilizar cada um dos *componentes* criados para serem manipulados de maneira distinta na aplicação. O *React* utiliza *JavaScript* como linguagem de programação ou até mesmo o *TypeScript*, que nada mais é que um *superset* do *JavaScript*, fornecendo recursos de uma linguagem de programação fortemente tipada.

Um dos benefícios para o desenvolvedor é que através do uso do *inteliSense* de um ambiente de desenvolvimento integrado (IDE), é possível analisar erros de tipos a fim de evitar a entrega da aplicação com falhas estruturais.

Algumas bibliotecas foram cruciais para a construção da aplicação, como o *Chakra UI*, *Multer*, *next-connect*, *NextAuth.js* e o *Fauna*, e serão melhor abordadas no próximo capítulo.

As subseções a seguir expõem de maneira explicativa as linguagens, ferramentas e plataformas utilizadas no desenvolvimento da aplicação.

## **3.1** *Next.js* & *React*

### **3.1.1** Contexto Histórico

Em 2011, o *React* foi criado pelo time do Facebook e surgiu com o objetivo de otimizar a atualização e sincronização de atividades simultâneas no feed de notícias da rede social. Essas atividades, que foram chamadas de estados, se tornaram mais simples com o *React*, o qual, por demonstrar grande eficiência, foi incorporado à interface de outras redes sociais do grupo, como o Instagram. O início de sua popularização se deu em 2013, quando teve seu código aberto à comunidade.

Com o decorrer do tempo, o *Next.js* foi lançado pela primeira vez como um projeto de código aberto no GitHub em 2016, e o seu autor original é Guillermo Rauch, atual CEO da Versel, empresa responsável pelo *framework*.

### **3.1.2** Características Gerais

O uso do *Next.js* como *framework* de desenvolvimento Web construído sobre o *Node.js* tem como principais vantagens habilitar sobre o *React* funcionalidades como *server-side rendering* (renderização do lado do servidor), *static site generator* (geração de páginas estáticas) e *file-system routing* (roteamento através de arquivos do sistema). Com esse conjunto previamente citado, desenvolvedores agora usufruem de mais recursos de maneira concentrada e agilidade na entrega das aplicações desenvolvidas através do uso desta ferramenta.

Vale salientar que nem todas as aplicações devem focar apenas no *Next.js* como única ferramenta de desenvolvimento. Em alguns dos casos, as regras de negócio são vastas e apenas o uso do *Next.js* resultará em um projeto fadado ao fracasso. Logo cabe não somente ao desenvolvedor, mas também ao projetista, saber distinguir quando somente utilizar o *Next.js* e quando utilizar um servidor à parte que atenderá todas as necessidades levantadas previamente.

#### **3.1.2.1** *Node.js*

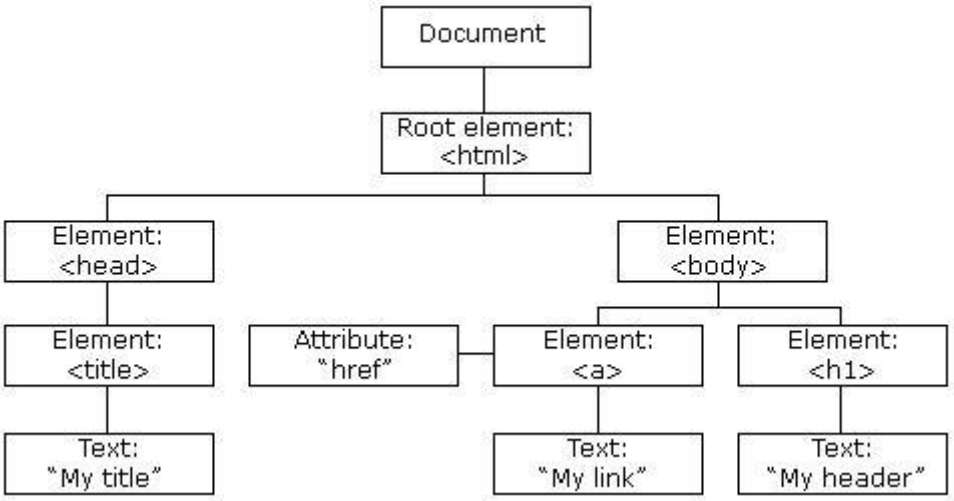
O Node.js é uma plataforma assíncrona com paradigma de I/O não bloqueante, orientado a eventos, multiplataforma que roda em cima do *Engine V8* e executa código *JavaScript* fora de um navegador da Web. A ideia por trás do *Node.js* é permitir com que desenvolvedores utilizem o *JavaScript* para a escrita de ferramentas de linhas de comando, scripts do lado do servidor e criação de aplicações de rede escaláveis.

Essa plataforma entrega aos seus usuários facilidade através da não preocupação de ter o processo travado, pois não existem travas, exceto quando o I/O é executado usando métodos síncronos da biblioteca padrão da ferramenta. Portanto, sem o bloqueio, obtém-se a possibilidade de criação de servidores escaláveis sem usar *threading*, através do uso de um modelo simplificado de programação orientada a eventos que usa *callbacks* para sinalizar a conclusão de uma tarefa. A cada conexão, o retorno de chamada é acionado, porém se não existir trabalho a ser realizado, o *Node.js* ficará inativo.

#### **3.1.2.2** *React*

É uma das bibliotecas do *JavaScript* mais populares e é utilizada para construção de uma interface de usuário (UI). Sua usabilidade de fácil compreensão entrega uma eficiente experiência do usuário (UX), fornecendo ao desenvolvedor reusabilidade de código através da construção de *componentes*, facilidade na escrita, excelente responsividade, assim como eficácia no processo de atualização do DOM, Figura 1.

Figura 1: Ilustração da DOM (Document Object Model)




Fonte: GeekHunter [FREIRES, 2019]

O *React* permite a construção de um DOM virtual e o hospeda na memória, desta forma, sempre que ocorrer qualquer alteração no DOM real, o virtual é modificado. Esse aspecto é bastante interessante, pois a manipulação de um DOM virtual é mais ágil que a de um DOM real, e após a criação desta estrutura virtual é traduzido à tela real o mínimo de processos possíveis, obtendo-se velocidade na atualização.

A facilidade na escrita se dá pela cordialidade entre a escrita de códigos HTML e *JavaScript* em um único arquivo JSX, ou TSX, utilizando *TypeScript*.

Uma das características do *React* sobre uma determinada página é o particionamento em pequenos pedaços, os quais são chamados de *componentes*, sendo cada um deles isolados e independentes entre si. Sua utilização é uma das principais vantagens da biblioteca, uma vez que são reutilizáveis e podem ser reaproveitados em qualquer outra página.

Além disso, o *React* é uma SPA (Single Page Application), cuja funcionalidade está concentrada em uma única página. Isso significa que quando há uma mudança de página, *componentes* que são compartilhados entre as páginas permanecem estáticos, caso possuam esta propriedade naquele momento, e os *componentes* dinâmicos, novos ou inexistentes serão recarregados, inseridos ou removidos, respectivamente. Portanto, não há a necessidade de recarregar a página por completo, somente o que foi alterado, consequentemente esta propriedade entrega otimização na performance, reduzindo o conteúdo a ser carregado.

Além do que foi citado anteriormente, essa biblioteca entrega muitas outras funcionalidades ao desenvolvedor que optar pelo seu uso. Alguns dos variados exemplos são o uso de contexto, *hooks*, estados, dentre outros. É bastante recomendado ao leitor o uso do *React* para o fins citados, porém vale lembrar que é importante a leitura da documentação para um aprendizado mais profundo acerca da biblioteca.

#### **3.1.2.3** *Next.js*

Trata-se de *framework* criado e mantido pela Vercel, que possui como motivação prover diversas funcionalidades à arquitetura *client-side* (lado do cliente) do *React*, na qual uma vez que o conteúdo é carregado, precisa-se aguardar o pacote que contém todo o *JavaScript* carregar para que por fim seja possível determinar o que mostrar na página.

Portanto, desenvolvedores não apenas conseguem desfrutar das maravilhas do *React* como também possuem em mãos funcionalidades como sistema de rotas, renderização estática e híbrida de conteúdo, pre-fetching, suporte a TypeScript, pacotes de funcionalidades, além de diversos plugins. Como é um *framework* de código aberto, assim como o *React*, qualquer pessoa está livre para desfrutar de todo o poder que o *Next.js* fornece, visto possui todas essas facilidades pré-configuradas, se assimilando bastante ao comando ‘create-react-app’, onde o projeto é inicializado de maneira bastante rápida através de um terminal *Unix*.

Nesse trabalho iremos comentar sobre duas das diversas utilidades fornecidas, sendo elas *server-side rendering* e *file-system routing*, as quais o projeto obteve o máximo de sustentação.

**3.1.2.3.1** *Server-side Rendering*

A renderização das páginas na parte do servidor resolve uma das deficiências do *React*, já que nem sempre é eficiente carregar todo o conteúdo na parte do cliente. O *SSR (server-side rendering)* funciona da seguinte forma, ao acessar um site é enviada uma requisição para o servidor. Em seguida, o servidor manda uma requisição para uma *API REST* pedindo a página solicitada. Após o recebimento dos dados, o servidor devolve ao cliente a página HTML com seu conteúdo renderizado.

Um ótimo exemplo do seu benefício seria no contexto de *web crawlers*, robôs automatizados que fazem a pesquisa e extração de grande volume de dados em tempo real, buscarem o conteúdo de algum e-commerce. Sem a renderização desse conteúdo pelo lado do servidor, informações cruciais acabam não sendo carregadas a tempo na página, comprometendo a busca.

Portanto, com o uso do SSR, o tempo de carregamento da aplicação é reduzido, já que o esforço passa a ser do servidor, consequentemente consome-se menos recursos.

**3.1.2.3.2** *File-system Routing*

O *file-system routing* veio como facilitador do roteamento das páginas da aplicação. Com apenas o uso do *React* precisaríamos de uma biblioteca chamada *react-router-dom* para lidar com roteamento na aplicação, porém com o *Next.js* isso ocorre com o que chamamos de *file-system routing*, ou seja, quando um arquivo JSX ou TSX é adicionado em um diretório com o nome ‘pages’ da aplicação, o nome desse arquivo estará automaticamente disponível como uma rota, e a partir do momento que esta página for acessada, o conteúdo do retorno da função *default* do arquivo será exibido em tela.

É importante mencionar que para os arquivos servirem como rotas eles obrigatoriamente devem estar localizados no diretório mencionado.

Outra característica que trás bastante poder ao *Next.js* são as *API Routes*. Qualquer arquivo localizado dentro da pasta ‘pages/api’ será tratado como um *endpoint* da *API Node.js*, cujo *Next.js* possui. Como são *bundles* do lado do servidor, os do lado do cliente não sofrem

impacto no tamanho. Para que a rota API funcione, é necessário exportar a função do arquivo como *default*. Este recurso possibilita a criação de toda uma API com as *API Routes*.

### **3.2** *Chakra UI*

Essa biblioteca de *componentes* facilita a construção da UI fornecendo um conjunto de *componentes* simples, modulares e acessíveis para construir aplicações *React*. Sua fácil manutenibilidade deve-se à desnecessidade de um arquivo .css para aquele componente específico, ou seja, a estilização é feita dentro da própria *tag* HTML.

O *Chakra UI* oferece um tema padrão, em que atributos como fontes, tamanhos de tela, cores, além de outros são pré-definidos, e sua flexibilidade se deve ao fato da biblioteca permitir ao desenvolvedor a criação livre do seu próprio tema.

### **3.3** *Multer*

Atualmente a maioria das aplicações necessitam de algum método de upload de arquivos, e é com esse propósito que o *Multer* aparece como uma biblioteca do *JavaScript*. Ele é um *middleware* que intercepta o arquivo e o armazena em algum diretório escolhido pelo desenvolvedor. Com o seu uso também é possível fazer o upload de múltiplos arquivos.

### **3.4** *next-connect*

*Next-connect* é uma *router* e *middleware layer* para servidores HTTP *Next.js*, *Micro* ou *Node.js*. O uso desta biblioteca facilita o roteamento de métodos e a adição de *middleware* às aplicações.

### **3.5** *NextAuth.js*

A autenticação é uma etapa fundamental nas aplicações atualmente, na qual as credenciais do usuário são utilizadas para verificar o usuário que está tentando acessar a aplicação. Essa biblioteca possui a finalidade de trazer uma estrutura de autenticação flexível a fim de sincronizar com qualquer serviço *OAuth*.

O *NextAuth.js* possui suporte para bancos de dados populares, como é o caso do *MongoDB*, *PostgresSQL*, *MariaDB* e *MySQL*, porém também pode ser usado sem um banco de dados, caso seja essa a escolha do desenvolvedor ou de sua equipe. A não necessidade do uso de um banco pode acontecer devido à sincronização com serviços como *JWT* (JSON Web Token) e *OAuth*.

### **3.6** *Fauna*

O *Fauna* é um banco de dados que transforma o *DBMS* (Database Management System) em uma *API* de dados para aplicações *client-serverless* fornecendo todos os recursos necessários para operar o banco, cuja vantagem é oferecer recursos de bancos tradicionais, mas sem sacrificar o desempenho, escalabilidade e flexibilidade. Possui uma interface *GraphQL* nativa da Web, com suporte para lógica de negócios personalizada e integração com ecossistemas sem servidor, por isso *client-serverless*.

### **3.7** *ApexCharts*

É uma biblioteca de gráficos que possui o intuito de auxiliar os desenvolvedores de software a criarem visualizações simples e responsivas para páginas da Web. Possui diversos tipos de gráficos customizáveis em seu arsenal que podem ser combinados a fim de fornecer uma diferença clara dos dados.

## 4.DESENVOLVIMENTO DA APLICAÇÃO

Esta seção possui a finalidade de expor a maneira na qual a aplicação foi desenvolvida, de forma detalhada e concisa.

### **4.1** Metodologia

A metodologia na etapa de desenvolvimento da aplicação foi estruturada utilizando-se as ferramentas e bibliotecas contextualizadas no Capítulo 3 desse trabalho. Para estruturar a base do projeto foi utilizado o seguinte comando: “yarn create next-app --typescript”, dessa maneira toda estrutura que precisamos do *Next.js* foi entregue. Para o versionamento do

código foi utilizado o *GitHub*, com ele foi possível criar *branches* para separar as etapas que foram almejadas, ou até mesmo os *componentes* criados.

Posteriormente a motivação da obtenção de uma arquitetura limpa foi essencial para a segregação do código de cada página em *componentes* que possuem uma única responsabilidade, através do princípio oriundo do *SOLID*, o *Single Responsibility Principle* (SRP). Foi a partir desse conceito que foram criados *componentes* com finalidades diferentes para a construção do todo e com a possibilidade de serem reutilizados em diferentes páginas. É importante mencionar que todos os arquivos e pastas estão concentrados na pasta 'src'.

Quanto às regras de negócio da aplicação foram necessárias algumas medidas para também separar bem as responsabilidades e alocar os arquivos para os diretórios que melhor os convêm. Por exemplo, arquivos de configuração estão alocados na pasta 'config', já os arquivos de *componentes* estão nos diretórios cujos nomes fazem referência ao seu propósito, já esses diretórios dos *componentes* estão alocados na pasta 'components', e assim por diante.

O diretório 'pages' possui a responsabilidade de tratar do roteamento da aplicação, como foi explicitado na Seção 3.1.2.3.2 desse trabalho. É nele que obtemos o poder do *file-system routing*, e é utilizado não apenas para inserir as páginas da aplicação, como também para cuidar das rotas HTTP que foram necessárias para cuidar do upload dos arquivos, assim como do processo da criação das rotas de autenticação.

Por fim, para consolidar o propósito da aplicação, a criação de uma função chamada *shapeEvasionCSVLines* para o cálculo da evasão foi inserida no arquivo 'helpers' no diretório 'utils'. Seu uso ocorre somente quando um arquivo que possui as colunas necessárias para o cálculo for adicionado, caso contrário é recomendado o uso de uma planilha .csv no formato de uma matriz quadrada, visto que o gráfico dispõe os dados em um plano 2D. Essa dinâmica é possível através do uso de um dos *hooks* do *React*, o *useEffect*, e será explicada mais detalhadamente na Seção 4.5.7.

As próximas seções possuem o intuito de explicar detalhadamente o funcionamento dos códigos fundamentais para a entrega do trabalho apresentado.

## **4.2** Upload de arquivos

O *next-connect* foi extremamente importante para prover uma camada de *middleware* ao *Next.js*. Posteriormente foi possível com o *Multer* prover arquivos à aplicação e com o *axios* lidar com as requisições, pois é um cliente HTTP baseado em *Promises*.

Foram construídas duas rotas para lidar com os arquivos da aplicação. Uma para inserção do arquivo e outra para a listagem dos arquivos. A de listagem apenas devolve os arquivos do sistema, já a de inserção insere e devolve a listagem, que por sua vez é um array de objetos que contém o nome e as linhas do arquivo como *chave:valor*.

### **4.2.1** Arquivo de configuração do *Multer*

O arquivo de configuração do *Multer*, localizado no diretório 'config', é essencial para definir o nome do arquivo e local de armazenamento. Nessa aplicação o nome do arquivo é criado com o horário do upload concatenado através de um *underscore* com o nome original do arquivo, e o local de armazenamento atual é o diretório './public/uploads'. Código disponível na Figura 2.

Figura 2 - Arquivo de configuração do *Multer*

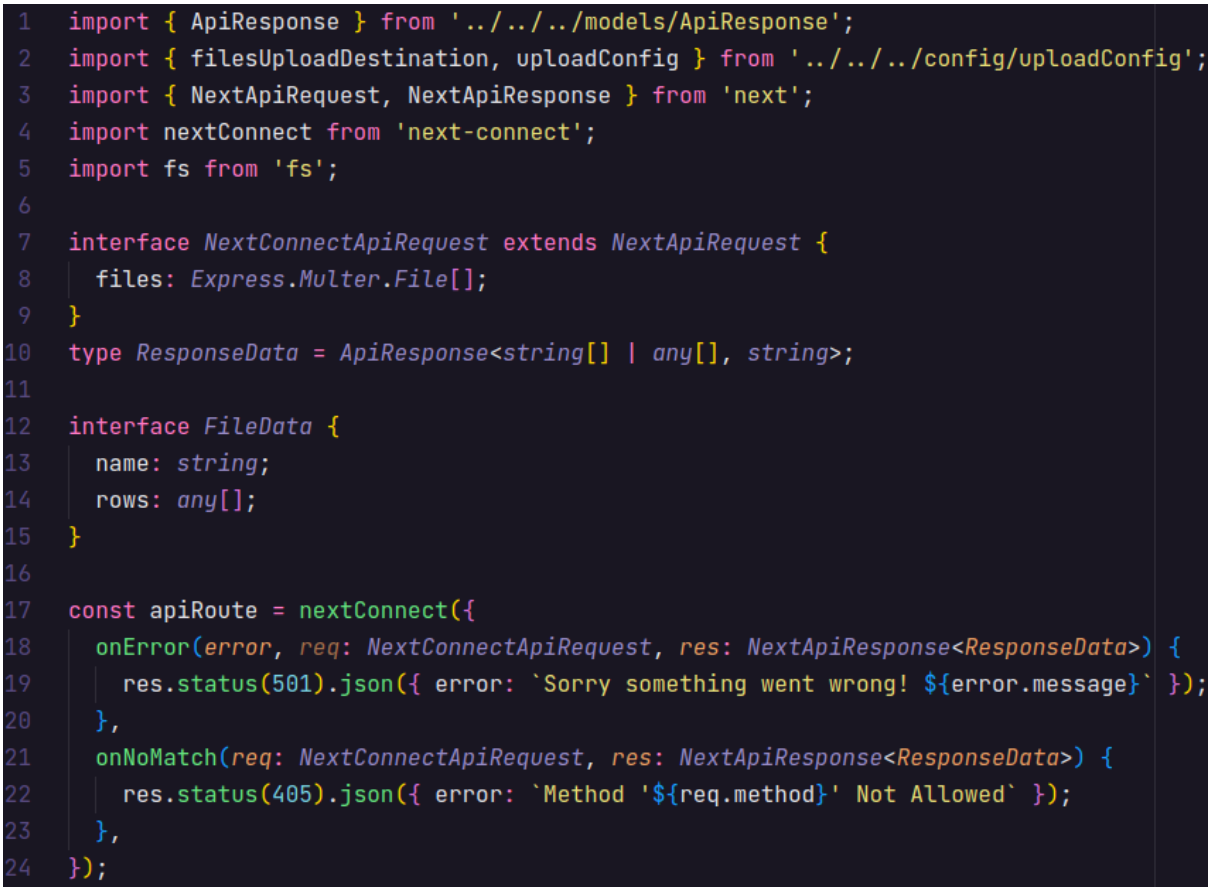
```
1  import multer from "multer"
2
3  export const filesUploadDestination = "./public/uploads"
4
5  export const uploadConfig = multer({
6    storage: multer.diskStorage({
7      destination: filesUploadDestination,
8      filename: (request, file, callback) => {
9        const date = new Date().getDate().toString();
10       const month:number = new Date().getMonth() + 1;
11       const year = new Date().getFullYear().toString();
12       const hours = new Date().getHours().toString();
13       const minutes = new Date().getMinutes().toString();
14       const seconds = new Date().getSeconds().toString();
15       const today = `${date}-${month}-${year}(${hours}:${minutes}:${seconds})`
16
17       const fileName = `${today}_${file.originalname}`;
18
19       return callback(null, fileName);
20     },
21   })
22 })
```

Fonte: Autor, 2022

**4.2.2** Rota de upload de arquivos

Algumas tipagens foram inseridas previamente para uso na tipagem da rota. As linhas de 7 a 9 mostram a tipagem da requisição adaptada para o *Multer*, já a linha 10 a tipagem do retorno. Na Figura 3, as linhas de 17 a 24 nos mostram a adaptação da nossa rota com o *next-connect*.

Figura 3 - Tipagens e uso do *nextConnect*



```
1  import { ApiResponse } from '../../../models/ApiResponse';
2  import { filesUploadDestination, uploadConfig } from '../../../config/uploadConfig';
3  import { NextApiRequest, NextApiResponse } from 'next';
4  import nextConnect from 'next-connect';
5  import fs from 'fs';
6
7  interface NextConnectApiRequest extends NextApiRequest {
8    files: Express.Multer.File[];
9  }
10 type ResponseData = ApiResponse<string[] | any[], string>;
11
12 interface FileData {
13   name: string;
14   rows: any[];
15 }
16
17 const apiRoute = nextConnect({
18   onError(error, req: NextConnectApiRequest, res: NextApiResponse<ResponseData>) {
19     res.status(501).json({ error: `Sorry something went wrong! ${error.message}` });
20   },
21   onNoMatch(req: NextConnectApiRequest, res: NextApiResponse<ResponseData>) {
22     res.status(405).json({ error: `Method '${req.method}' Not Allowed` });
23   },
24 });
```

Fonte: Autor, 2022

A linha 26 da Figura 4 representa o uso do *Multer*. O *array* significa que foi habilitado o upload de múltiplos arquivos, caso optássemos pelo upload de apenas um por vez, teríamos utilizado o *single* logo após *uploadConfig*.

Logo após a inserção dos arquivos no sistema, a rota passa a ser responsável por lê-los e devolvê-los de maneira organizada. Isso ocorre da seguinte forma:

1. Linha 33: responsável por fazer a leitura de um arquivo
2. Linha 34: converte para o resultado da linha 33 para um padrão legível, o *utf-8*
3. Linha 35: o retorno da linha 34 é separado por nova linha (*regex* \n) e em seguida cada linha é separada por vírgula (,)
4. Linha 36: toda linha com ‘\r’ no final tem seu valor substituído por uma *string* vazia
5. Linhas 37-40: um objeto contendo a tipagem *FileData*, linha 12 da Figura 3, é inserido no array *filesData* que possui a mesma tipagem

6. Linha 43: Após a conclusão da leitura de todos os arquivos, o retorno da rota será o array *filesData* atribuído à variável data.

Toda rota API pode exportar um objeto de *config* com o intuito de alterar as configurações padrões, em consequência disto, as linhas 46 a 50 desabilita o *body parsing*. Por fim, a linha 52 exporta a rota para uso.

Figura 4 - Análise da rota de upload

```
26  apiRoute.use(uploadConfig.array('files'));
27
28  apiRoute.post((req: NextConnectApiRequest, res: NextApiResponse<ResponseData>) => {
29    const filesData: FileData[] = []
30    const filenames = fs.readdirSync(filesUploadDestination);
31
32    filenames.forEach((name, index) => {
33      const fileBuffer = fs.readFileSync(`${filesUploadDestination}/${name}`)
34      const fileContent = fileBuffer.toString("utf-8")
35      const fileSplit = fileContent.split("\n").map(file => file.split(","))
36      const fileMap = fileSplit.map(line => line.map(column => column.replace("\r", "")))
37      filesData.push({
38        name: filenames[index],
39        rows: fileMap
40      })
41    });
42
43    res.status(200).json({ data: filesData });
44  });
45
46  export const config = {
47    api: {
48      bodyParser: false, // Disallow body parsing, consume as stream
49    },
50  };
51
52  export default apiRoute;
```

Fonte: Autor, 2022

### **4.2.3** Serviço de *upload* com *Axios*

O serviço de upload é uma função, com o nome *uploadFileRequest*, que solicita como parâmetro obrigatório um formData, cujo tipo é FormData, ou seja, utiliza o mesmo formato que um formulário usaria se o tipo de codificação fosse definido como "multipart/form-data".

Neste serviço foi necessário definir no cabeçalho da requisição o *content-type* como *multipart/form-data* para que a API soubesse o que fazer com os dados recebidos.

O retorno desse serviço são os dados obtidos da resposta da rota requisitada, cujo objetivo é o *upload* dos arquivos. O *Axios* foi utilizado para a requisição dos dados e recebeu como primeiro parâmetro o caminho (*path*) da rota, como segundo os dados que por sua vez é o *formData* e por último as configurações com o cabeçalho modificado para atender o propósito do serviço.

Figura 5 - Função uploadFileRequest

```
1  import axios, { AxiosRequestConfig } from 'axios';
2  import { ApiResponse } from '../models/ApiResponse';
3
4  export const uploadFileRequest = async (
5    formData: FormData,
6    progressCallback?: (progressEvent: ProgressEvent) => void
7  ): Promise<ApiResponse<string[]>> => {
8    const config: AxiosRequestConfig = {
9      headers: { 'content-type': 'multipart/form-data' },
10     onUploadProgress: progressCallback,
11     validateStatus: (status) => true,
12   };
13   const response = await axios.post('/api/uploads', formData, config);
14
15   return response.data;
16 };
```

Fonte: Autor, 2022

A conclusão do ciclo de *upload* dos arquivos .csv continua com a chamada da função *uploadFileRequest* pela função *onChange*, utilizada em todas as páginas localizadas no diretório '*/pages/dashboards*', e consequentemente atribuída ao método *onChange* do componente *FileInput*, explicado na Seção 4.5.1.

## **4.3** Funções

### **4.3.1** Função *shapeCSVLines*

O uso desta função foi necessária para a criação de um objeto cujas chaves são as primeiras linhas do arquivo .csv que foi inserido, e os valores são as demais linhas de sua respectiva coluna. Em caso de nenhum valor naquela posição especifica, uma string vazia é inserida, caso contrário todos os valores existentes terão todos os espaçamentos existentes substituídos por *underscore* (_) e todos caracteres do alfabeto convertidos para letra minúscula.

A finalidade da função na criação do objeto com essas características é única e exclusivamente para que o usuário possa escolher um determinada linha e coluna para o gráfico da aplicação.

Figura 6 - função *shapeCSVLines*

```
export const shapeCSVLines = (array: string[][]) => {
    const object = {}
    array.forEach((row, rowIndex) => {
      rowIndex === 0 ?
        array[rowIndex].forEach((column) => {
          Object.assign(object, {
            ...object,
            [column.replace(/\s/g, "_").toLowerCase()]: [] as string[]
          })
        })
      : array[rowIndex].forEach((column, columnIndex) => {
        const column_name = object[`${array[0][columnIndex]}`.replace(/\s/g, "_").toLowerCase()]
        typeof(column) !== 'string' ?
          column_name?.push('')
          : column_name?.push(column)
      })
    })

    return object
  }
```

Fonte: Autor, 2022

### **4.3.2** Função *shapeEvasionCSVLines*

#### **4.3.2.1** Filtragem das colunas e criação da coluna *ingresso_evasao_total*

A função será explicitada a seguir através do seu uso na planilha com os dados referentes à evasão no âmbito do curso de graduação de Ciências da Computação do Centro de Informática da UFPE. Contudo, seu uso também funciona para analisar as planilhas referentes

à evasão dos cursos de graduação de Engenharia da Computação e Sistemas de Informação do mesmo centro acadêmico.

O início da função *shaveEvasionCSVLines* possui quase a mesma lógica da função da seção 4.3.1, porém com uma pequena diferença: é feita uma filtragem na primeira linha do arquivo para obter apenas as colunas necessárias para o cálculo da evasão, as quais são 'período_ingresso', 'semestre_da_evasão', 'número_de_evadidos'. Por fim, uma última coluna é criada, chamada 'ingresso_evasao_total'. A Figura 7 representa a explicação.

Figura 7 - função *shapeEvasionCSVLines*, filtragem e criação da nova coluna

```
22  export const shapeEvasionCSVLines = (array: string[][]) => {
23    const object = {}
24    const evasionColumns = ['período_ingresso', 'semestre_da_evasão', 'número_de_evadidos_por_turma']
25    // here i start by filtering which columns will be worked on
26    array.forEach((row, rowIndex) => {
27      rowIndex === 0 ?
28        array[rowIndex].forEach((column) => {
29          if(column) {
30            const column_name = column.replace(/\s/g, "_").toLowerCase()
31            evasionColumns.includes(column_name) ?
32              Object.assign(object, {
33                ...object,
34                [column_name]: [] as string[]
35              }) : null;
36          }
37        })
38      : array[rowIndex].forEach((column, columnIndex) => {
39        const column_name = object[`${array[0][columnIndex]}`.replace(/\s/g, "_").toLowerCase()]
40        if (column_name){
41          if (typeof(column) !== 'string') {
42            column_name?.push('')
43          } else {
44            column_name?.push(column)
45          }
46        } else {
47          return null;
48        }
49      })
50    })
51
52    // here i create a new column named 'ingresso_evasao_total'
53    Object.assign(object, {
54      ...object,
55      ['ingresso_evasao_total']: []
56    })
```

Fonte: Autor, 2022

**4.3.2.2** Inserção de valores na coluna *ingresso_evasao_total*

De posse das chaves e valores necessários é possível prosseguir para a próxima etapa, cuja funcionalidade é a de percorrer as linhas das três primeiras colunas. Os valores de cada linha agrupados são colocados em um array, e por fim o array é inserido com o valor daquela linha na nova coluna ‘ingresso_evasao_total’.

O processo se inicia pela criação de um array na primeira coluna da linha, a inserção do valor no array e do array em sua respectiva linha da coluna ‘ingresso_evasao_total’. Já os valores das próximas colunas ‘semestre_da_evasão’ e ‘número_de_evadidos’ serão inseridos dentro do array que já possui um valor inserido em ‘ingresso_evasao_total’. O último valor será inserido no formato de número caso a string completa não possua algum caractere do tipo string, caso contrário será inserido um valor nulo. A Figura 8 expõe esta argumentação.

Figura 8 - função *shapeEvasionCSVLines*, agrupamento dos valores em um array

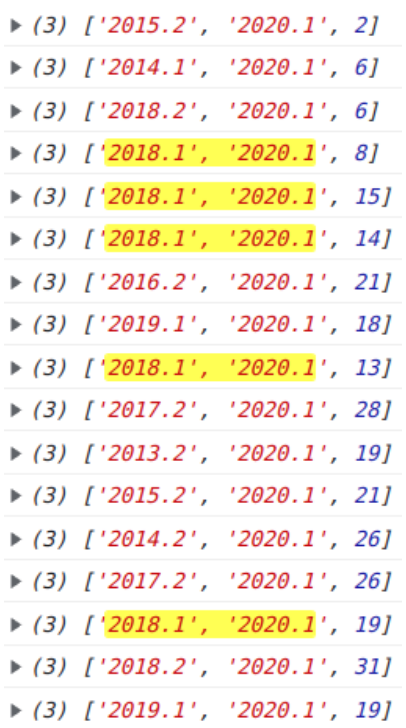
```
58    const ing_ev_tot_array = object['ingresso_evasao_total'] as any[]
59
60    // here i walk through each element of the original columns that is necessary
61    // to be worked on and toss them together to the new column
62    Object.keys(object).forEach((key, index) => {
63      if (index === 0) {
64        Object.keys(object[key]).forEach(value => {
65          const ing_ev_tot_data = []
66          ing_ev_tot_data.push(object[key][value]);
67          ing_ev_tot_array.push(ing_ev_tot_data);
68        })
69      }
70      if (index === 1) {
71        Object.keys(object[key]).forEach(value => {
72          ing_ev_tot_array[value].push(object[key][value]);
73        })
74      }
75      if (index === 2) {
76        Object.keys(object[key]).forEach(value => {
77          const total = object[key][value] as string;
78          // in case total string has any character other than a digit,
79          // a null value will be sent to the array, else the value will be sent
80          // as a number
81          if (Boolean(total.match(/(\D)|(^$)/g))) {
82            ing_ev_tot_array[value].push(null)
83          } else {
84            ing_ev_tot_array[value].push(Number(total));
85          }
86        })
87      }
88    })
```

Fonte: Autor, 2022

**4.3.2.3** Agrupamento de valores repetidos e somatório do último índice desses valores

Após a conclusão da etapa apresentada anteriormente foi possível visualizar que alguns valores agrupados do primeiro e segundo índice dos arrays de ‘ingresso_evasao_total’ estão repetidos. Um exemplo é apresentado na Figura 9, onde podemos visualizar a repetição de ‘2018.1’ e ‘2020.1’.

Figura 9 - resultado da função *console.log* na variável *ing_ev_tot* inseridas após as linhas 101 e 103

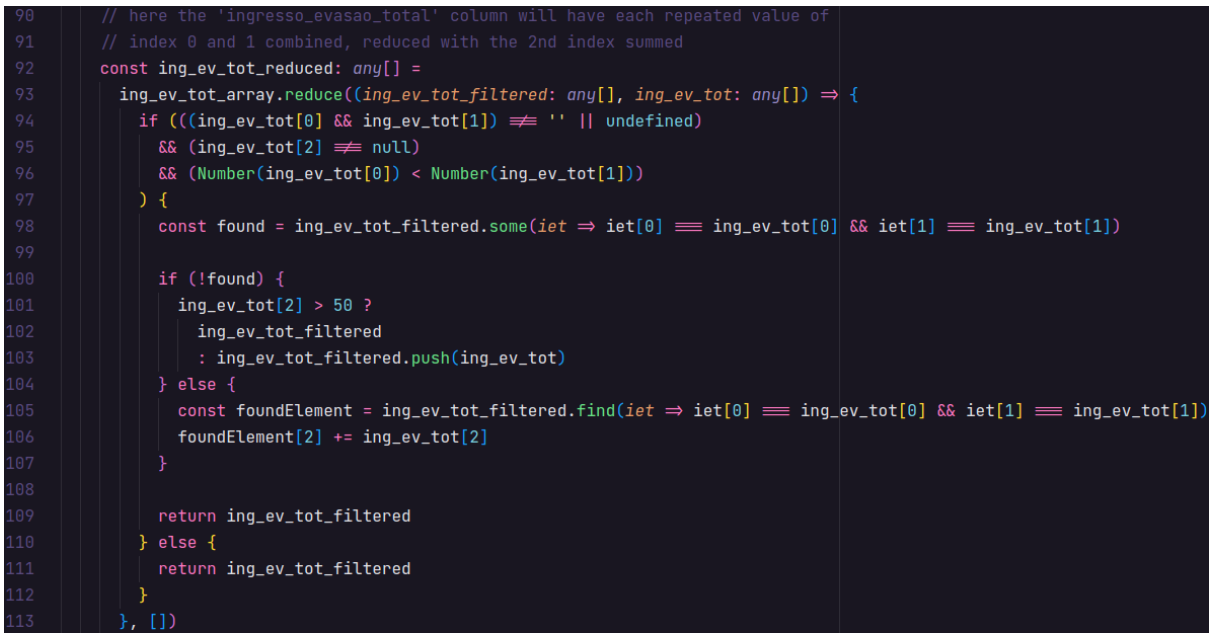



Fonte: Autor, 2022

A etapa apresentada anteriormente foi fundamental para possibilitar o agrupamento dos valores repetidos, do primeiro e segundo índice combinados, e realizar o somatório do terceiro através da função nativa do *JavaScript*, o *reduce*.

Algumas condições foram feitas para que somente valores condizentes permanecessem, linhas 94-96 da Figura 10. Não permaneceram arrays com string vazia em qualquer um dos dois primeiros índices, nem com o último índice possuindo valor nulo e, por fim, o valor do primeiro índice que é referente ao período de ingresso não pode ser maior ou igual ao valor do segundo, que corresponde ao semestre da evasão.

Figura 10 - função *shapeEvasionCSVLines*, agrupamento dos valores repetidos, do primeiro e segundo índice combinados, e realizar o somatório do terceiro



```
// here the 'ingresso_evasao_total' column will have each repeated value of
// index 0 and 1 combined, reduced with the 2nd index summed
const ing_ev_tot_reduced: any[] =
  ing_ev_tot_array.reduce((ing_ev_tot_filtered: any[], ing_ev_tot: any[]) => {
    if (((ing_ev_tot[0] && ing_ev_tot[1]) !== '' || undefined)
      && (ing_ev_tot[2] !== null)
      && (Number(ing_ev_tot[0]) < Number(ing_ev_tot[1]))
    ) {
      const found = ing_ev_tot_filtered.some(iet => iet[0] === ing_ev_tot[0] && iet[1] === ing_ev_tot[1])

      if (!found) {
        ing_ev_tot[2] > 50 ?
          ing_ev_tot_filtered
          : ing_ev_tot_filtered.push(ing_ev_tot)
      } else {
        const foundElement = ing_ev_tot_filtered.find(iet => iet[0] === ing_ev_tot[0] && iet[1] === ing_ev_tot[1])
        foundElement[2] += ing_ev_tot[2]
      }

      return ing_ev_tot_filtered
    } else {
      return ing_ev_tot_filtered
    }
  }, [])
```

Fonte: Autor, 2022

Outra condição foi a de que caso não existisse tal combinação entre o primeiro e segundos índices no novo array reduzido, porém o terceiro índice possuísse valor superior a 50, esses dados não seriam inclusos. Esse critério foi utilizado porque dados “sujos” estão inseridos na planilha original onde é possível visualizar um valor total dos alunos que evadiram em determinado período. Esse valor atrapalha a análise dos dados e portanto essa metodologia foi utilizada sem prejudicar nenhuma das três planilhas analisadas, visto que os demais valores não ultrapassam o valor de 50.

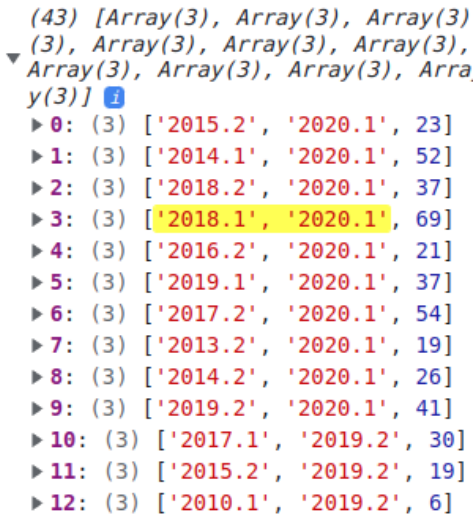
Figura 11 - Dados “sujos” na linha selecionada da tabela de evadidos de Ciências da Computação do Centro de Informática da UFPE

| Período Ingresso | Tipo de Ingresso | Perfil Curricular | Turno | Semestre da Evasão | SEMESTRE DE INGRESSO | Número de evadidos por turma |
|---|---|---|---|---|---|---|
| | | | | 2020.2 | 2020.0 | 0 |
| 2015.2 | SISU | 2002-1 | INTEGRAL | 2020.1 | 2020.1 | 2 |
| 2014.1 | VESTIBULAR | 2002-1 | INTEGRAL | 2020.1 | 2019.2 | 6 |
| 2018.2 | SISU | 2002-1 | INTEGRAL | 2020.1 | 2019.1 | 6 |
| 2018.1 | SISU | 2002-1 | INTEGRAL | 2020.1 | 2018.2 | 8 |
| 2018.1 | SISU | 2002-1 | INTEGRAL | 2020.1 | 2018.1 | 15 |
| 2018.1 | SISU | 2002-1 | INTEGRAL | 2020.1 | 2017.2 | 14 |
| 2016.2 | SISU | 2002-1 | INTEGRAL | 2020.1 | 2017.1 | 21 |
| 2019.1 | SISU | 2002-1 | INTEGRAL | 2020.1 | 2016.2 | 18 |
| 2018.1 | SISU | 2002-1 | INTEGRAL | 2020.1 | 2016.1 | 13 |
| 2017.2 | SISU | 2002-1 | INTEGRAL | 2020.1 | 2015.2 | 28 |
| 2013.2 | VESTIBULAR | 2002-1 | INTEGRAL | 2020.1 | 2015.1 | 19 |
| 2015.2 | SISU | 2002-1 | INTEGRAL | 2020.1 | 2014.2 | 21 |
| 2014.2 | VESTIBULAR | 2002-1 | INTEGRAL | 2020.1 | 2014.1 | 26 |
| 2017.2 | SISU | 2002-1 | INTEGRAL | 2020.1 | 2013.2 | 26 |
| 2018.1 | SISU | 2002-1 | INTEGRAL | 2020.1 | 2013.1 | 19 |
| 2018.2 | SISU | 2002-1 | INTEGRAL | 2020.1 | 2012.2 | 31 |
| 2019.1 | SISU | 2002-1 | INTEGRAL | 2020.1 | 2012.1 | 19 |
| 2014.1 | VESTIBULAR | 2002-1 | INTEGRAL | 2020.1 | 2011.2 | 22 |
| 2014.1 | VESTIBULAR | 2002-1 | INTEGRAL | 2020.1 | 2011.1 | 24 |
| 2019.2 | SISU | 2002-1 | INTEGRAL | 2020.1 | 2010.2 | 18 |
| 2019.2 | SISU | 2002-1 | INTEGRAL | 2020.1 | 2010.1 | 23 |
| 2016.1 | SISU | 2002-1 | INTEGRAL | 2020.1 | total | 379 |
| 2015.2 | TRANSFERENC | 2002-1 | INTEGRAL | 2020.1 | | |
| 2011.1 | VESTIBULAR | 2002-1 | INTEGRAL | 2020.1 | | |
| 2010.2 | VESTIBULAR | 2002-1 | INTEGRAL | 2019.2 | | |
| 2013.2 | VESTIBULAR | 2002-1 | INTEGRAL | 2019.2 | Evasão turma do período de 2010.1 a 2020.1 | |

Fonte: Autor, 2022.

Após o método de *reduce* podemos referenciar novamente o intervalo entre ‘2018.1’ e ‘2020.1’ para exemplificar que o agrupamento dos dois primeiros índices resultou o somatório esperado de 69 no último índice. Vemos também que o array com valor de 379 foi removido

Figura 12 - resultado após o método de reduce no array de arrays *ing_ev_tot_array*




Fonte: Autor, 2022

Ainda assim, como podemos observar na Figura 12, temos apenas os valores absolutos da evasão, o que não representa a taxa da evasão para aquele determinado período de tempo. A fórmula para obtenção da taxa de evasão na qual se baseia esse trabalho encontra-se na seção seguinte.

**4.3.2.4** Cálculo da evasão

Figura 13 - Cálculo da taxa de evasão

$$TDA_{j,T,t} = \frac{\sum_{w=T}^{t} \sum_{i=1}^{n_{3,j,w}} Des_{i,j,t} + \sum_{w=T}^{t} \sum_{i=1}^{n_{4,j,w}} Transf_{i,j,t}}{\sum_{i=1}^{n} IG_{i=j}^{T} - \sum_{w=T}^{t} \sum_{i=1}^{n_{6,j,w}} Fal_{i,j,t}}$$

Fonte: [CHAGAS, 2019]

Onde:

Des = Estudante Desvinculado do curso j no ano t.

Transf = Estudante Transferido para outro curso da mesma IES no curso j no ano t.

IG = Número total de ingressantes no curso j no ano T.

Fal = Estudante com situação de vínculo igual a “Falecido” no curso j no ano t.

Como essa equação utilizada por Chagas [CHAGAS, 2019] reproduz a equação original proposta em INEP [2017], que por sua vez possui uma metodologia para acompanhamento do fluxo de entrada e saída de alunos, o presente projeto importou tal equação por permitir a obtenção de dados consistentes para os fins a que se propõe. Não é a única equação que existe, porém em razão da sua praticidade e da semelhança de escopos com o projeto de Chagas, foi a eleita.

De posse do número do total de evadidos no curso j no ano T, é preciso obter IG. Neste trabalho não foi possível trabalhar sobre Fal, pois os dados referentes são inexistentes nos arquivos analisados. Para o cálculo de IG e TDA a figura e as etapas explicitadas a seguir poderão fornecer uma maior clareza ao leitor.

**4.3.2.4.1** Etapas para o cálculo da taxa de evasão

A explicação das linhas a seguir fazem referência à Figura 14, disposta na página seguinte.

1. Linha 123: Como solicitado, o número de estudantes que ingressam no curso j no ano t foi definido como sendo o número 50.
2. Linhas 131-133 e 135-137: Se subtrairmos o semestre de evasão pelo período de ingresso não obteremos de maneira precisa o tempo em alguns dos casos exemplificados a seguir:
    a. 2021.2 - 2020.1 = 1.1 / Esperado: 1.5 anos
    b. 2021.1 - 2019.2 = 1.9 / Esperado: 1.5 anos

Para solucionar o problema a abordagem adotada foi a adição de 0.8 para o segundo semestre do ano e 0.4 para o primeiro. Logo, foi possível obter o tempo correto como os exemplos a seguir nos mostram:

    c. (2021.2 + 0.8) - (2020.1 + 0.4) = 2022 - 2020.5 = 1.5 / Esperado: 1.5 anos
    d. (2021.1 + 0.4) - (2019.2 + 0.8) = 2021.5 - 2020 = 1.5 / Esperado: 1.5 anos
    e. (2021.2 + 0.8) - (2020.2 + 0.8) = 2022 - 2021 = 1 / Esperado: 1 ano
    f. (2021.1 + 0.4) - (2019.1 + 0.4) = 2021.5 - 2019.5 = 2 / Esperado: 2 anos

3. Linha 139: Após a obtenção do tempo de maneira precisa, precisamos apenas multiplicá-lo por 2 para obtermos a quantidade de períodos.
4. Linha 140: IG foi adquirido através da multiplicação do resultado da Etapa 3 pelo da Etapa 1, ou seja, o número de estudantes que ingressam a cada período pelo total de períodos.
5. Linhas 142-143: Se o total de evadidos no curso j no ano T for superior que IG é retornado *null* na iteração atual.
6. Linhas 144-146: Caso a condição da Etapa 5 não seja verdadeira é retornado o TDA, percentual do número de estudantes que desistiram, foi obtido pela razão entre o total de evadidos no curso j no ano T por IGA cada iteração do array ‘ing_ev_tot_reduced’ é retornado um novo array contendo o período de ingresso, semestre da evasão e por último a taxa de evasão no lugar do total de evadidos no curso j no ano T.
7. Linha 117: O array ‘ing_ev_tot_reduced’ é mapeado em um novo array chamado ‘ing_ev_tot_result’, cujo conteúdo são novos arrays descritos na etapa anterior.

Figura 14 - função *shapeEvasionCSVLines*, Cálculo de IG e TDA

```
115     // here the dropout rate will be calculated according to the number of
116     // students who entered during that period of time
117     const ing_ev_tot_result = ing_ev_tot_reduced.map((iet: any[]): any[] => {
118       const entry_period: string = iet[0]
119       const semester_dropout: string = iet[1]
120       const students_dropout_amount: number = iet[2]
121
122       // as required the number of students who entered per semester is equal 50
123       const number_of_students_entered_per_period = 50
124
125       let entry_period_as_number = 0
126       let semester_dropout_as_number = 0
127
128       // if the period is the 2nd of the year we add .8 otherwise .4 since the
129       // other period would be the 1st. This happens to get the correct
130       // difference between periods
131       entry_period.includes(".2") ?
132         entry_period_as_number = Number(entry_period) + 0.8
133         : entry_period_as_number = Number(entry_period) + 0.4
134
135       semester_dropout.includes(".2") ?
136         semester_dropout_as_number = Number(semester_dropout) + 0.8
137         : semester_dropout_as_number = Number(semester_dropout) + 0.4
138
139       const evasion_time_in_periods = (semester_dropout_as_number - entry_period_as_number) * 2
140       const total_number_of_students_entered = evasion_time_in_periods * number_of_students_entered_per_period
141
142       if (students_dropout_amount > total_number_of_students_entered) {
143         return null
144       } else {
145         const evasion_rate = students_dropout_amount / total_number_of_students_entered
146         return [entry_period, semester_dropout, evasion_rate]
147       }
148     })
```

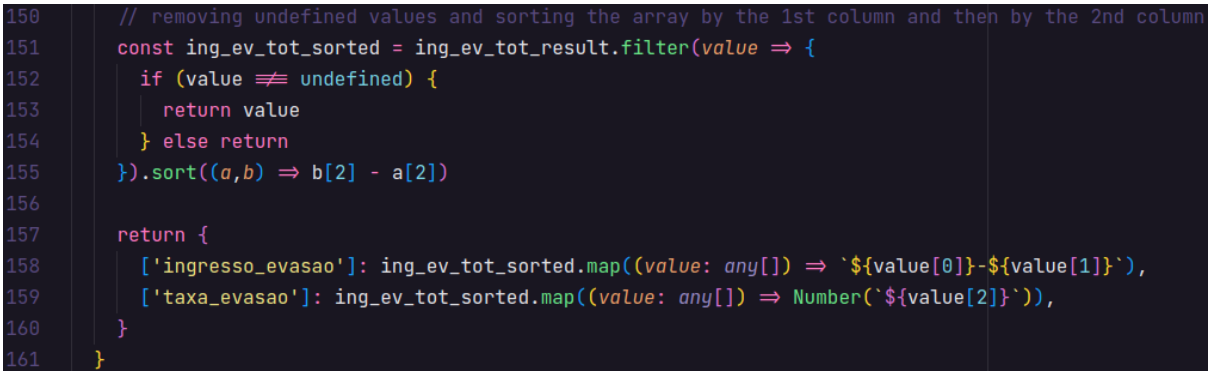
Fonte: Autor, 2022.

As etapas 5 e 6 foram essenciais para a remoção de valores inconsistentes aos esperados, pois não é possível que o número total de alunos que evadiram em um determinado período de tempo seja superior ao total de ingressantes do respectivo período.

#### **4.3.2.5** Retorno da função *shapeEvasionCSVLines*

Para finalizarmos a função, ‘ing_ev_tot_result’ é filtrado para a remoção de valores indefinidos e consecutivamente ordenado em um novo objeto chamado ‘ing_ev_tot_sorted’. Para finalizar a função *shapeEvasionCSVLines*, um novo objeto contendo novos conjuntos de *chave:valor* é retornado.

Figura 15 - Retorno da função *shapeEvasionCSVLines*



```
150     // removing undefined values and sorting the array by the 1st column and then by the 2nd column
151     const ing_ev_tot_sorted = ing_ev_tot_result.filter(value => {
152       if (value !== undefined) {
153         return value
154       } else return
155     }).sort((a,b) => b[2] - a[2])
156
157     return {
158       ['ingresso_evasao']: ing_ev_tot_sorted.map((value: any[]) => `${value[0]}-${value[1]}`),
159       ['taxa_evasao']: ing_ev_tot_sorted.map((value: any[]) => Number(`${value[2]}`)),
160     }
161   }
```

Fonte: Autor, 2022.

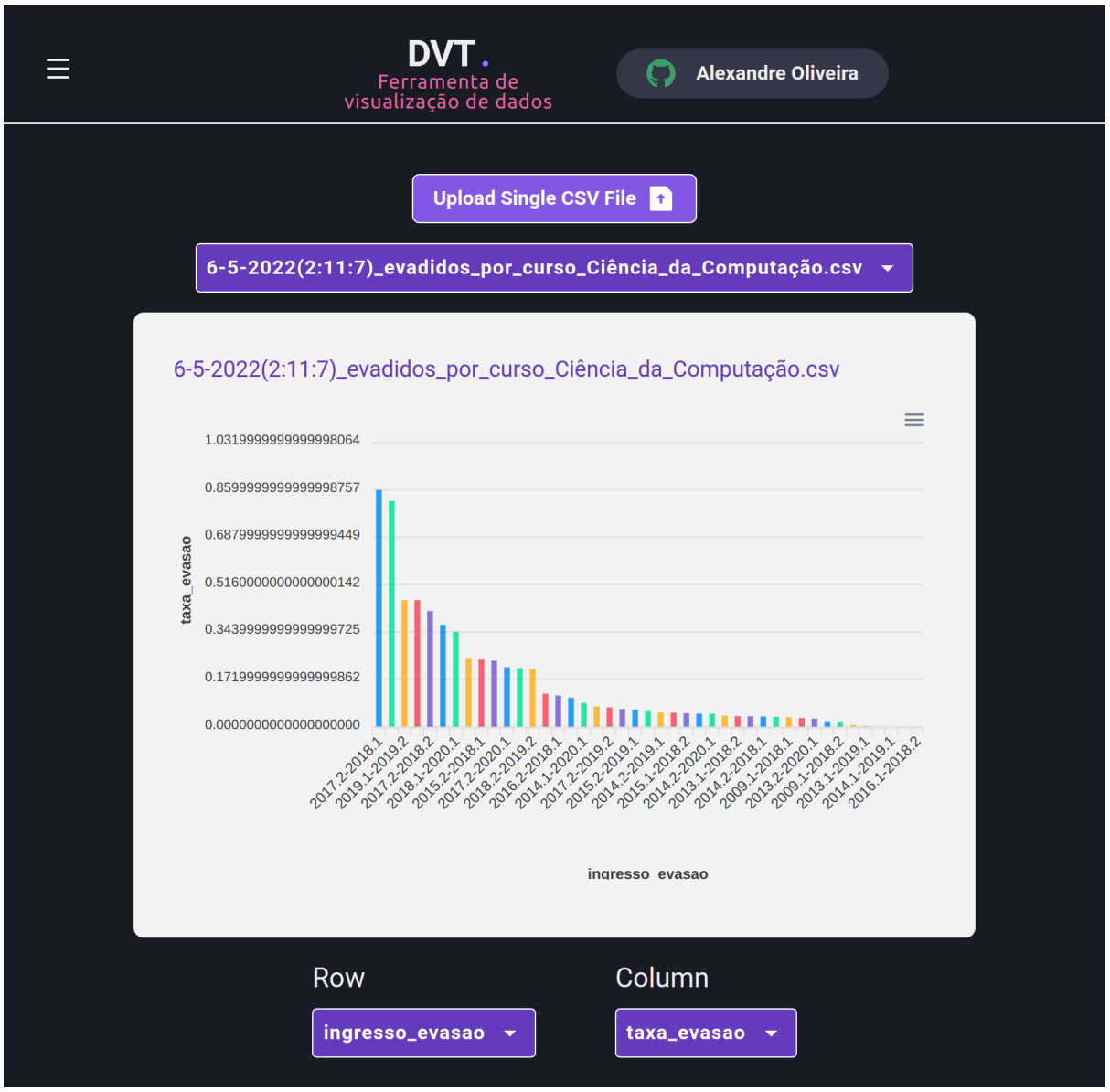
**4.3.2.5.1** Conjunto de *chave:valor* do objeto retornado

1. *chave*: ‘ingresso_evasao’.
   *valor*: Array contendo a seguinte string: ‘(valor do período de ingresso)-(valor do semestre da evasão)’.
2. *chave*: ‘taxa_evasao’.
   *valor*: Array contendo os percentuais do número de estudantes que desistiram.

A escolha da ordem de ordenação da função deve-se à necessidade de mapear em quais períodos concentram-se as maiores taxas de evasão. Portanto, os gráficos a seguir retratam em ordem decrescente as taxas de evasão nos cursos de graduação do Centro de Informática por período de tempo cursado, explicitando o semestre de ingresso e o de evasão do aluno.

**4.3.2.6** Gráficos obtidos através do retorno da função *shapeEvasionCSVLines*

Figura 16 - Gráfico referente à taxa de evasão pelo período de ingresso-período de evasão do curso de Ciências da Computação do Centro de Informática da UFPE

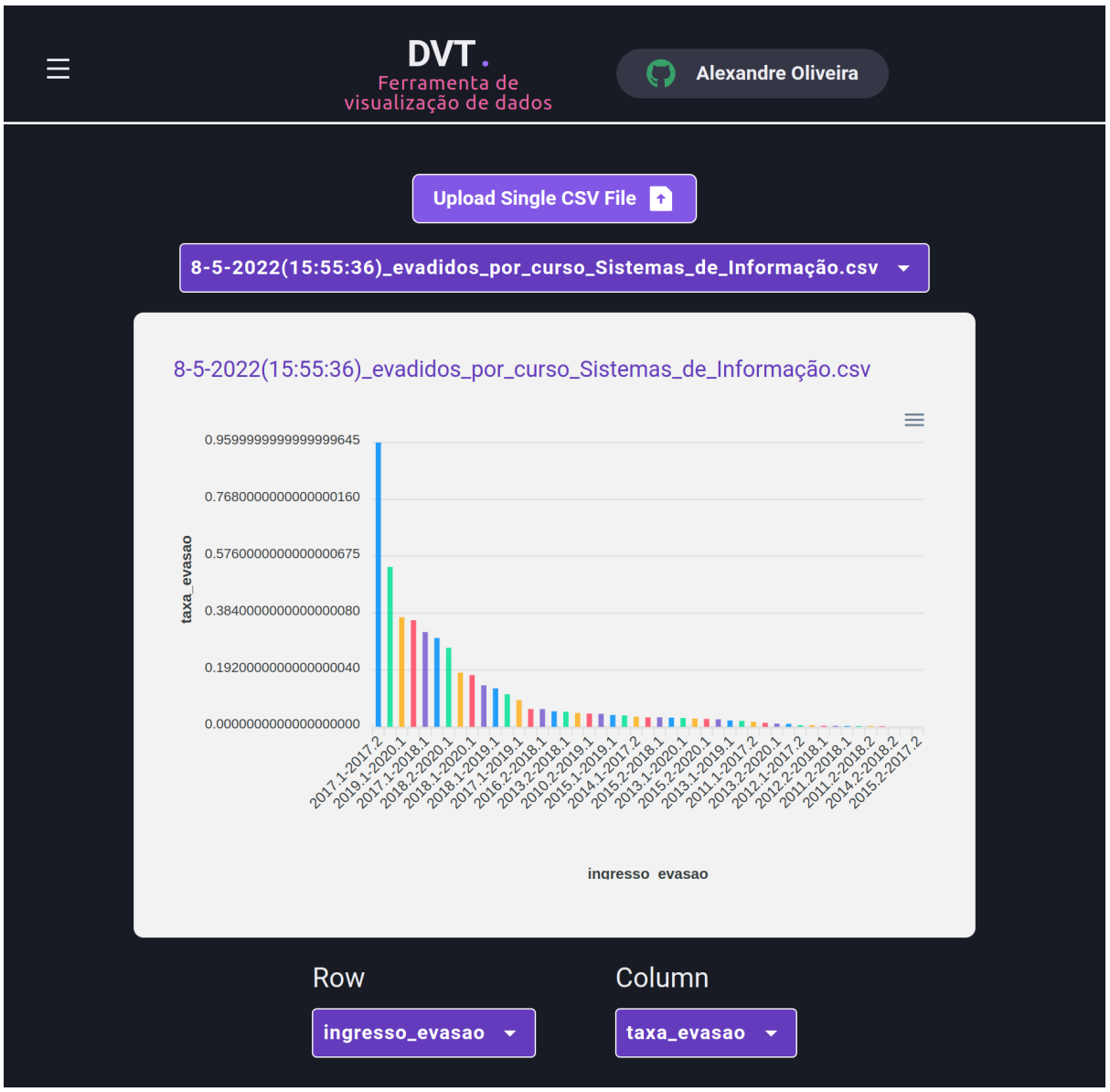



Fonte: Autor, 2022.

Figura 17 - Gráfico referente à taxa de evasão pelo período de ingresso-período de evasão do curso de Engenharia da Computação do Centro de Informática da UFPE


Fonte: Autor, 2022.

Figura 18 - Gráfico referente à taxa de evasão pelo período de ingresso-período de evasão do curso de Sistemas de Informação do Centro de Informática da UFPE




Fonte: Autor, 2022.

Da leitura dos referidos gráficos pode-se perceber que as maiores taxas de evasão dos cursos de graduação do Centro de Informática concentram-se nos 2 primeiros anos dos três cursos analisados (Engenharia da Computação, Ciência da Computação e Sistemas de Informação), o que corrobora com a hipótese apresentada na revisão de literatura sobre o tema da evasão, no Capítulo 2 desse trabalho.

Destaca-se o alarmante dado que aponta uma porcentagem de 96% dos ingressantes do período ‘2017.1’ do curso de Sistemas de Informação evadidos no período seguinte (2017.2).

## **4.4** Autenticação e banco de dados

O serviço de autenticação do cliente foi realizado através do uso do *NextAuth.js* integrado com o serviço de *OAuth* do *Github*. Para prover a sessão do usuário em um contexto global da aplicação onde só precisamos inserir o *provider* do *NextAuth.js* no arquivo '_app.tsx'. Quanto à explicação do uso de contextos na aplicação, a Seção 4.5.6 entra no mérito dessa temática. Por fim, fazemos o registro do usuário logado no banco de dados do *Fauna*.

### **4.4.1** Autenticação com o *NextAuth.js*

Ao utilizarmos o *provider* do *GitHub* para autenticar-se na aplicação é necessário atribuir ao *clientId* e ao *clientSecret* seus respectivos valores que foram armazenados como variáveis de ambiente e são fornecidos pelo próprio *git* do administrador do sistema. É possível sanar quaisquer dúvidas de implementação através da análise da documentação do [NextAuth.js].

Figura 19 - *NextAuth*, *GitHubProvider* (Parte 1)

```
 8  export default NextAuth({
 9    providers: [
10      GitHubProvider({
11        clientId: process.env.GITHUB_CLIENT_ID,
12        clientSecret: process.env.GITHUB_CLIENT_SECRET,
13      })
14    ],
```

Fonte: Autor, 2022.

### **4.4.2** Registro do usuário no *Fauna*

O registro do usuário no *Fauna* é possível através do método signIn fornecido pela propriedade das *callbacks* do *NextAuth*. Caso não exista um usuário cadastrado no banco com o email utilizado durante a autenticação, um novo usuário é criado, caso contrário o usuário existente é retornado como é possível verificar na Figura 20 através da sintaxe do *Fauna*.

Figura 20 - *NextAuth*, *GitHubProvider* (Parte 2)

```
callbacks: {
  async signIn({ user, account, profile }) {
    const { email } = user;

    try {
      await fauna.query(
        q.If(
          q.Not(
            q.Exists(
              q.Match(
                q.Index('user_by_email'),
                q.Casefold(user.email)
              )
            )
          ),
          q.Create(
            q.Collection('users'),
            { data: { email } }
          ),
          q.Get(
            q.Match(
              q.Index('user_by_email'),
              q.Casefold(user.email)
            )
          )
        )
      )

      return true
    } catch {
      return false
    }
  },
}
```

Fonte: Autor, 2022.

## **4.5** *Componentes*

Os *componentes*, como explicitado na Seção 4.1, foram segregados de acordo com suas responsabilidades e alguns deles serão mencionados e explicados. Porém, alguns outros não serão abordados, pois *componentes* como o *Header*, *SignInButton* e o *Sidebar*, são de fácil implementação e compreensão, assim como não foram cruciais para a funcionalidade de *upload* de arquivos. Todavia, todo o código [OLIVEIRA, 2022] para análise está disponível nas referências desse artigo caso seja de interesse do leitor.

### **4.5.1** *FileInput*

Este *componente* é um formulário que servirá para iniciar o processo de *upload* de arquivos. Ao clicar no botão desse formulário, o método *onClickHandler* é invocado e caso o usuário clique para adicionar algum arquivo o método *onChangeHandler* é chamado pelo método *onChange* da *tag* HTML *input*. Consecutivamente arquivos do tipo *FormData* são adicionados caso possuam o serviço de inserção coerente, como possui-se e o mesmo foi explicado na Seção 4.2, todo o processo é concluído com sucesso.

Figura 21 - Métodos *onClickHandler* e *onChangeHandler* do *componente FileInput*

```
export interface FileInputProps {
  acceptedFileTypes?: string;
  allowMultipleFiles?: boolean;
  label: string;
  onChange: (formData: FormData) => void;
  uploadFileName: string;
}

export const FileInput =
  ({ label, onChange, uploadFileName, acceptedFileTypes, allowMultipleFiles}: FileInputProps) => {
  const fileInputRef = useRef<HTMLInputElement | null>(null);
  const formRef = useRef<HTMLFormElement | null>(null);

  const onClickHandler = () => {
    fileInputRef.current?.click();
  };

  const onChangeHandler = (event: ChangeEvent<HTMLInputElement>) => {
    if (!event.target.files?.length) {
      return;
    }

    const formData = new FormData();

    Array.from(event.target.files).forEach((file) => {
      formData.append(event.target.name, file);
    });

    onChange(formData);

    formRef.current?.reset();
  };
```

Fonte: Autor, 2022.

Figura 22 - *tags* de formulário, botão e *input* do *componente FileInput*

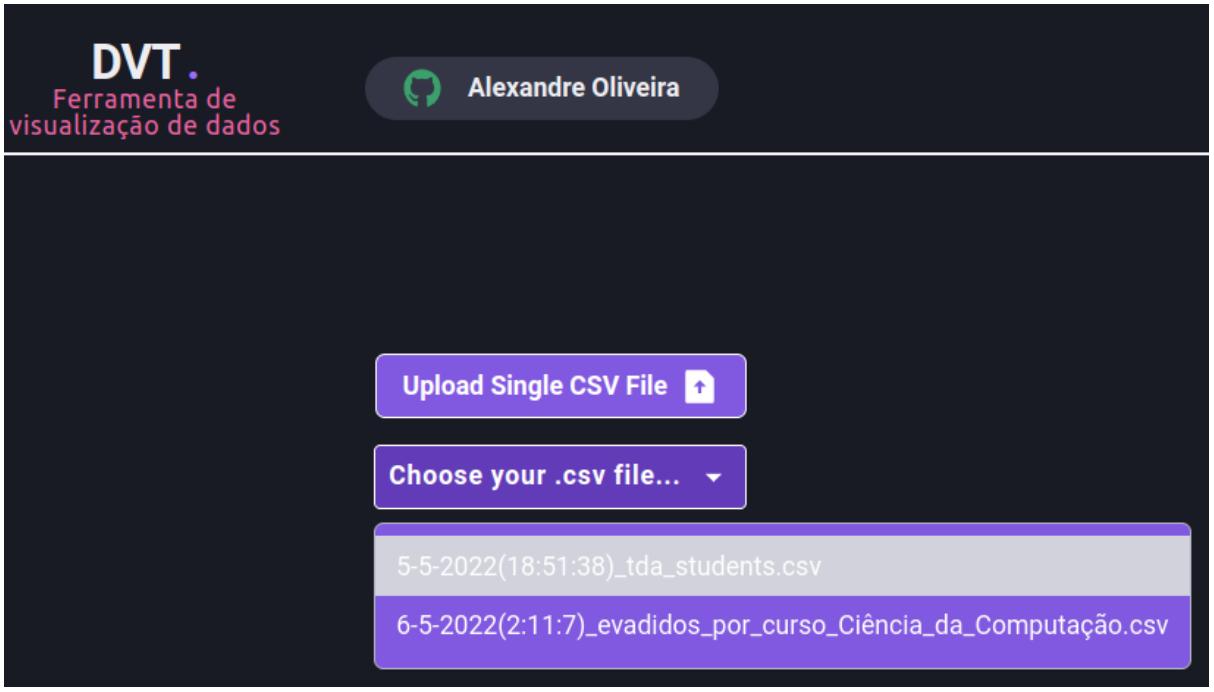
```
38    return (
39      <form ref={formRef}>
40        <Button
41          type="button"
42          onClick={onClickHandler}
43          color="white"
44          bgColor="purple.dvt-mid"
45          borderColor="white"
46          borderWidth={1}
47          _hover={{
48            bgColor: "purple.dvt-dark",
49            borderColor: "white.800"
50          }}
51          rightIcon={<Icon as={RiFileUploadFill} fontSize={24}/>}
52        >
53          {label}
54        </Button>
55        <input
56          accept={acceptedFileTypes}
57          multiple={allowMultipleFiles}
58          name={uploadFileName}
59          onChange={onChangeHandler}
60          ref={fileInputRef}
61          style={{ display: 'none' }}
62          type="file"
63        />
64      </form>
65    );
66  };
67
68  FileInput.defaultProps = {
69    acceptedFileTypes: '',
70    allowMultipleFiles: false,
71  };
```

Fonte: Autor, 2022.

### **4.5.2** *CustomCSVSelect*

Seu uso é fundamental para a listagem e seleção dos arquivos do sistema. A depender do tamanho da tela o *CustomCSVSelect* localiza-se abaixo ou ao lado do *componente FileInput*.

Figura 23 - *componente CustomCSVSelect* abaixo do *componente FileInput*




Fonte: Autor, 2022.

### **4.5.3** *CustomObjectSelect*

Sua funcionalidade é similar ao *componente CustomCSVSelect*, porém é utilizado para listar e selecionar a linha e coluna dos gráficos da aplicação. É possível visualizá-lo nas Figuras 16, 17 e 18 logo abaixo dos gráficos.

### **4.5.4** *AreaDashboard*, *ColumnDashboard* e *LineDashboard*

Estes três *componentes* possuem o mesmo propósito, o de prover um gráfico para visualização de dados através do uso da biblioteca *apexcharts*. Como ambas possuem construção semelhante, apenas diferenciando-se no método de visualização do gráfico, abordaremos nessa seção o *componente ColumnDashboard*, cujas figuras relacionadas são as 16, 17 e 18.

O *ColumnDashboard* possui internamente outro *componente* para a escolha da linha e coluna do gráfico, o *CustomObjectSelect*. Essa escolha é possível através do *hook useState* e do método *onChange* do *CustomObjectSelect*. Para a visualização do gráfico é utilizado o *componente Chart*, fornecido pelo *apexcharts*, o qual necessita dos atributos *options*, *series* e

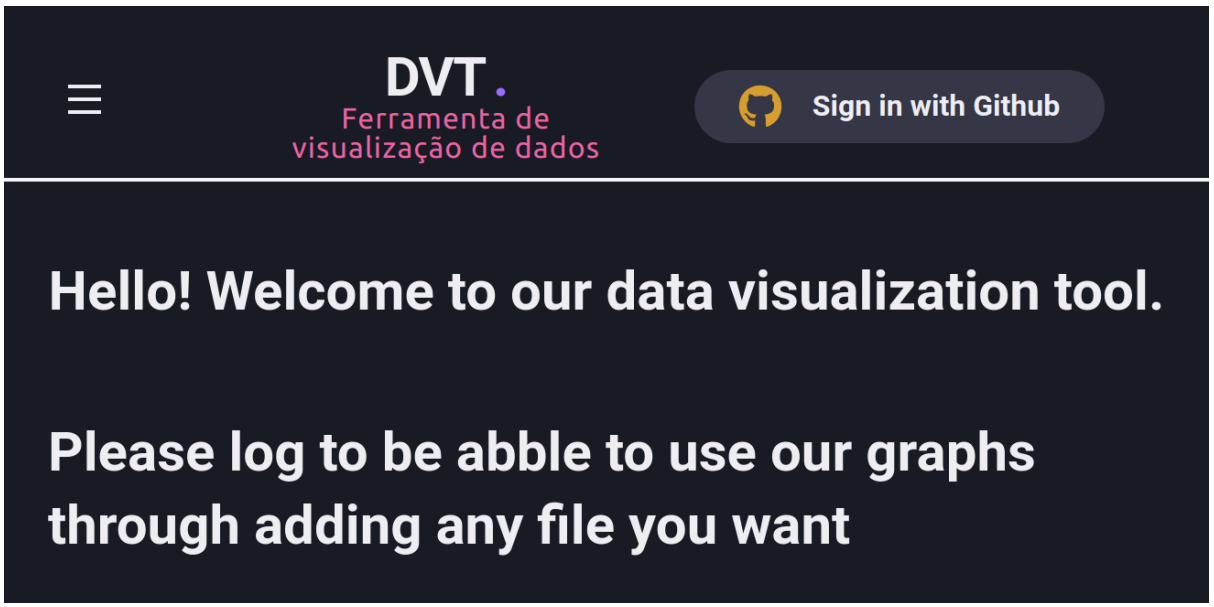
*type* para que nesse caso forneça o gráfico do tipo bar, o qual dispõe os dados no formato de colunas. Outros *componentes* como o *Box*, *Stack* e *Text* são fornecidos pelo *Chakra UI* e personalizados para se adequarem ao design da aplicação, assim como nos demais *componentes* construídos.

### **4.5.5** *Header*, *Logo*, *SignInButton*, *Sidebar* e *PageCompose*

Todos os *componentes* desse tópico, assim os demais, possuem seu nome de acordo com sua funcionalidade. Portanto, *Logo* entrega a logo da aplicação, o *SignInButton* o botão para a autenticação do usuário com sua conta do *GitHub*, o *Header* é o cabeçalho e contém os *componentes* previamente citados, o *Sidebar* serve para a visualização da barra lateral com os links das rotas da aplicação, e por fim o *PageCompose* que acomoda o Header e o *Sidebar*. No *PageCompose* é possível inserir qualquer outro componente filho e seu conteúdo fica disponível ao lado direito da barra lateral.

Vale salientar que a tela apresentada a seguir, na Figura 24, possui todos os componentes citados previamente com breve ressalva ao componente *SignInButton*, que está implicitamente solicitando o *login* do usuário, para que possa prover as principais funcionalidades do sistema.

Figura 24 - Tela inicial no contexto do usuário não autenticado




Fonte: Autor, 2022.

### **4.5.6** *SidebarDrawerContext*

Com o interesse de explicar contexto no *React*, esse componente será um bom exemplo para dar alguma clareza a respeito.

O *SidebarDrawerContext* é o único componente dessa aplicação que não está localizado no diretório '*components*', visto que ele não é um componente propriamente dito, mas sim um *context provider*. Isso significa que ele é utilizado no arquivo '_app.tsx' para prover um contexto à aplicação, assim como o *SessionProvider* do *NextAuth.js*. Esse contexto possui como finalidade fornecer o *Sidebar* de maneira dinâmica quando a tela estiver em proporção reduzida.

### **4.5.7** *AreaDashboardPage*, *ColumnDashboardPage* e *LineDashboardPage*

Assim como os componentes *AreaDashboard*, *ColumnDashboard* e *LineDashboard*, esses três componentes são similares e apenas o *ColumnDashboardPage* será explicado. Ele possui quatro outros componentes construídos, o *PageCompose*, *FileInput*, *CustomCSVSelect* e o *ColumnDashboard*. Esse último utiliza dois outros componentes internamente, são eles o *CustomObjectSelect*, explicitado na Seção 4.5.6 e o *Chart*, fornecido pelo *ApexCharts*, a fim de prover os gráficos da aplicação.

O arquivo onde *ColumnDashboardPage* está inserido será tratado como rota da aplicação, pois o mesmo localiza-se no diretório '/pages/dashboards'. Isso ocorre porque temos funcionalidade *file-system routing* do *Next.js*. Nele foi possível trabalhar com *upload* de arquivos, listagem e seleção dos arquivos do sistema, visualização dos dados e *server-side rendering* (SSR).

Para facilitar o entendimento, o *upload* de arquivos através do *componente ColumnDashboardPage* será esclarecido nas seguintes etapas:

1. É adicionado um arquivo com o uso do *componente* FileInput. Figura 26, linha 60.
2. O método *onChange* (Figura 26, Linha 63) é chamado e seu ciclo explicado na etapa 3.

3. *onChange* recebe um *formData* e o repassa para o método *uploadFileRequest*, que por sua vez terá os dados de sua resposta atribuído à variável *filesData*.
4. O usuário estará apto a escolher um dos arquivos do sistema com o *componente CustomCSVSelect*.
5. Assim que o arquivo for escolhido pelo usuário, seu valor é atribuído a variável *file* através do método *onChange* do *CustomCSVSelect* (Figura 26, Linha 69) e do *hook useState* (Figura 25, Linha 19).
6. Com o *hook useEffect* (Figura 25, Linhas 37-49) é possível atribuir à variável *fileObject* uma das duas funções existentes que fazem a filtragem dos arquivos, a cada mudança da variável *file*. Caso o arquivo trabalhado seja o esperado para o cálculo da evasão, a função *shapeEvasionCSVLines* é utilizada, caso contrário *shapeEvasionLines* será utilizada para o *parsing* das linhas de *file*.
7. Após o uso de uma das funções a variável *fileObject* é atribuída a propriedade *object* do componente *ColumnDashboard*. Figura 26, Linha 74.
8. *ColumnDashboard* disponibiliza o gráfico a partir dos dados disponibilizados. Figura 26, Linha 74.

Figura 25 - *ColumnDashboardPage* (Parte 1)

```tsx
export default function ColumnDashboardPage() {
  const [filesData, setFilesData] = useState<any[]>()
  const [file, setFile] = useState<FileProps>()
  const [fileObject, setFileObject] = useState({})

  const onChange = async (formData: FormData) => {
    const response = await uploadFileRequest(formData, (event) => {
      console.log(`Current progress:`, Math.round((event.loaded * 100) / event.total));
    });

    setFilesData(response.data)
  };

  useEffect(() => {
    const response = getUploadedFiles()
    Promise.resolve(response).then(res => {
      setFilesData(res.data)
    }).catch(err => console.error(err))
  }, [])

  useEffect(() => {
    if (file) {
      const column_names = file.rows[0] as string[]
      column_names.includes('Período Ingresso')
      && column_names.includes('Semestre da Evasão')
      && column_names.includes('Número de evadidos por turma') ?
        setFileObject(shapeEvasionCSVLines(file.rows))
        : setFileObject(shapeCSVLines(file.rows))

    } else {
      return null
    };
  }, [file])
```

Fonte: Autor, 2022.

Figura 26 - *ColumnDashboardPage* (Parte 2)

```
return (
  <PageCompose header_title="DVT | Column Dashboard">
    <Stack flex="1" direction="column" spacing={[4, 4, 4, 4, 6]} justify="center" align="center">
      <Stack
        mt={4}
        align="center"
        direction={["column", "column", "column", "column" ,"row"]}
        spacing={[4, 4, 4, 4, 16]}
      >
        <FileInput
          label="Upload Single CSV File"
          uploadFileName="files"
          onChange={onChange}
        />

        <CustomCSVSelect
          value={file}
          items={filesData}
          onChange={(newValue) => setFile(newValue)}
        />
      </Stack>

      {file && (<Stack w="100%" maxW="680px" h="100%">
        <ColumnDashboard name={file?.name} object={fileObject}/>
      </Stack>)}
    </Stack>
  </PageCompose>
)
}
```

Fonte: Autor, 2022.

Figura 27 - SSR da página ‘/dashboards/column’

```
export const getServerSideProps: GetServerSideProps = async(ctx) => {
  const data = await getSession({
    ctx
  })

  if (data?.user) {
    return {
      props: {}
    }
  } else {
    return {
      redirect: {
        destination: "/",
        permanent: false
      }
    }
  }
}
```

Fonte: Autor, 2022.

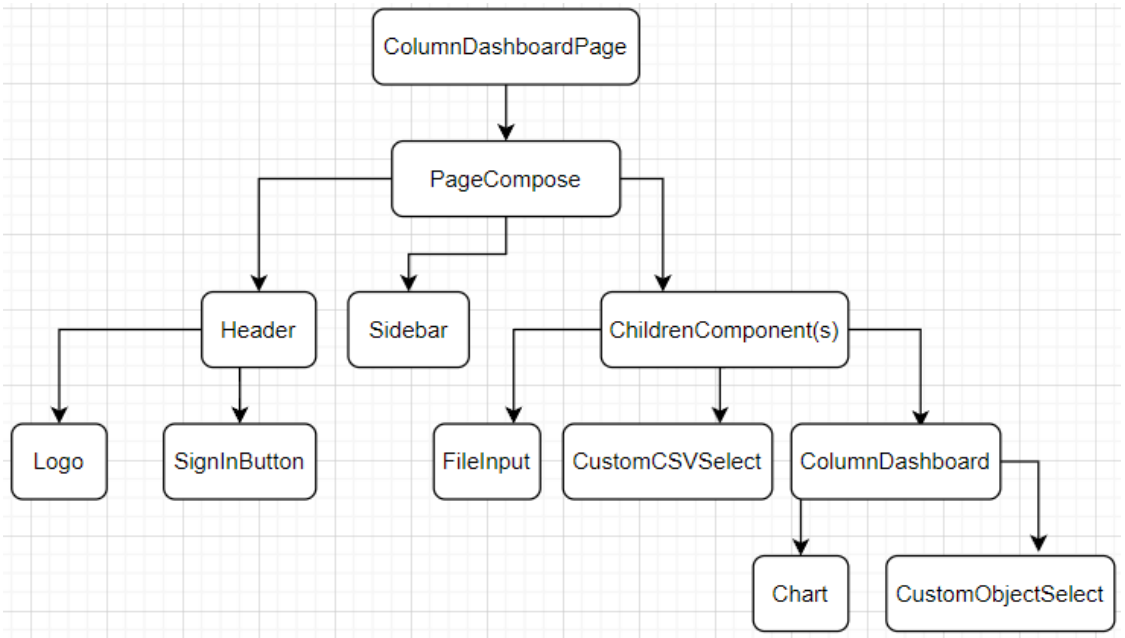
As etapas citadas anteriormente só serão possíveis caso o usuário esteja logado. Isso acontece pois o uso do método *getServerSideProps* (Figura 27), irá garantir a funcionalidade de SSR.

A sessão do usuário é obtida passando o contexto como parâmetro para o método *getSession*, fornecido pelo *NextAuth.js*. Se o usuário estiver autenticado sua permanência na página é garantida, se não o usuário da aplicação será redirecionado para a página principal onde precisará se autenticar para poder ter acesso a qualquer uma das páginas de visualização de gráficos.

Com o exemplo da Figura 27 é possível ter uma melhor compreensão de como utilizar o SSR no *Next.js*.

Para finalizar esse capítulo, dispõe-se uma visualização do fluxo da maioria dos componentes criados, a fim de fornecer clareza ao leitor. Uma melhor compreensão será obtida a partir de um diagrama de dependência entre os componentes que compõem a página '/dashboards/column', a qual disponibilizará o conteúdo presente no componente *ColumnDashboardPage*. A Figura 28 a seguir representa o que foi dito.

Figura 28 - Diagrama de dependência entre os componentes que compõem a página '/dashboards/column'




Fonte: Autor, 2022.

# 5.CONSIDERAÇÕES FINAIS

A evasão nas universidades é um problema social que requer maiores estudos e proposições de soluções por parte de pesquisadores e administradores, especialmente quando envolve o emprego de verbas públicas de interesse social.

Foi proposto como problema de pesquisa a capacidade de a ferramenta desenvolvida nesse trabalho possibilitar uma visualização satisfatória dos cálculos de evasão, a fim de contribuir para mitigá-la. Sobre isso, cabe tecer algumas considerações.

## **5.1** Contribuições

As funcionalidades dessa ferramenta são: o fornecimento de gráficos através das funcionalidades dos componentes construídos; a possibilidade do upload de arquivos com o *Next.js* e outras bibliotecas do *JavaScript*; a tratativa dos dados dos arquivos inseridos no sistema e, finalmente, a escolha da linha e coluna para a visualização dos dados. Vale lembrar que para a possibilidade das etapas citadas é necessário que o usuário esteja logado. Isso foi possível com o uso do *NextAuth.js* e o SSR do *Next.js*.

Conforme achados da pesquisa contidos na Seção 4 desse trabalho, a hipótese ventilada pela literatura, segundo a qual as maiores taxas de evasão concentram-se nos primeiros anos de cursos de graduação, se confirma pela testagem dos dados obtidos do Centro de Informática da UFPE na ferramenta desenvolvida, os quais verificam-se na Seção 4.3.2.6.

## **5.2** Limitações

Ainda, cabe realizar uma autocrítica da ferramenta, que merece um aprimoramento ante as limitações impostas pelo upload local dos arquivos e pela não possibilidade de agrupamento de campos repetidos na função *shapeCSVLines*. Como o upload dos arquivos foi feito localmente, quando realizado o *deploy* da ferramenta em algum serviço de computação em nuvem, todos os arquivos inseridos no sistema por um usuário estarão disponíveis para os demais, o que é inviável para uso em larga escala.

Já em relação à não possibilidade de agrupamento de campos repetidos na função *shapeCSVLines*, nos referimos à opção do emprego do método *reduce*, assim como foi feito na função *shapeEvasionCSVLines*. Porém não de modo igual, pois na função da evasão sabemos em quais colunas queremos fazer o agrupamento e somatório dos dados, contudo na função de uso geral da aplicação, a *shapeCSVLines*, não foi obtida clareza para sua manipulação até o momento.

A Interface Gráfica do Usuário (GUI) também pode ser melhorada, pois por uma questão temporal o período de desenvolvimento da aplicação foi focado na entrega do resultado esperado, e consequentemente resultou em uma GUI simples e sem muito foco na experiência do usuário.

Vale ressaltar que a ferramenta depende de um formato específico de entrada para que o cálculo da taxa de evasão performe. Logo, a ferramenta como está posta possui condições de realizar o cálculo da taxa de evasão nos moldes das planilhas relacionadas ao CIn. Porém, caso gestores de outros centros ou universidades possuam dados equivalentes e queiram usufruir do que foi construído, será necessário alterar o nome das colunas para ‘período_ingresso’, ‘semestre_da_evasão’, ‘número_de_evadidos’, dispondo os dados na planilha respeitando essa ordem .

## **5.3** Trabalhos futuros

Como dito anteriormente, a ferramenta precisa de uma entrada específica para performar o cálculo da evasão. Portanto, a aplicação desenvolvida poderia ser útil em qualquer outro curso, departamento, ou até mesmo em um contexto de evasão geral nas universidades, pois apenas precisaria de dados concisos sobre o período de ingresso, semestre da evasão e o número total de evadidos durante esse período.

Além disso, o projeto permite ir além do que foi feito a partir da implementação de melhorias por terceiros, uma vez que o código é aberto. Algumas dessas melhorias seriam: i) a busca por um banco de dados mais adequado ao *Next.js*; ii) a inserção de arquivos em algum serviço de computação em nuvem como o *S3* da *AWS*; iii) modularização das funções *shapeCSVLines* e *shapeEvasionCSVLines* e iv) o uso de conceitos mais profundos da Engenharia de Software.

Quanto aos desafios os quais pessoas ou instituições podem enfrentar ao adotar a ferramenta verifica-se a própria adequação dos objetivos buscados à ferramenta, visto que algumas práticas não foram aprofundadas. A aplicação ainda possui uma experiência do usuário relativamente fraca quanto à autenticação, falta uma tela inicial que explique com detalhes como utilizar a ferramenta, sem falar no uso de uma GUI simples, assim como a falta da implementação de testes, fator que pode resultar em falhas após o desenvolvimento da aplicação.

Feitos os devidos esclarecimentos, ressalta-se que, sem prejuízo dos obstáculos encontrados, a proposta de desenvolvimento da ferramenta sugerida para um trabalho a nível de graduação foi executada, pelo que se propõe o desenvolvimento de futuros trabalhos visando o aprimoramento dessa ferramenta, a fim de torná-la mais funcional e eficaz para a consecução dos fins a que se propõe. Uma direção de trabalho diz respeito ao uso de técnicas de aprendizagem de máquina para identificação de padrões e também como forma de antecipar tendências para evasão.

## 6.REFERÊNCIAS

ANDRADE, Ana. Conhecendo o Next.js. Disponível em: <https://www.treinaweb.com.br/blog/conhecendo-o-next-js>. Acesso em: 01/05/2022.

ANDRADE, António. Como usar a IU do Chakra com Next.js e React. Disponível em: <https://cibersistemas.pt/tecnologia/como-usar-a-iu-do-chakra-com-next-js-e-react/#:~:text=Chakra%20UI%20%C3%A9%20uma%20biblioteca,para%20come%C3%A7ar%20a%20funcionar%20rapidamente.>. Acesso em: 01/05/2022.

AURELIO, Italo. O que é e para que serve o Next.js?. Disponível em: <https://segredo.dev/o-que-e-next-js/>. Acesso em: 01/05/2022.

BRAGA, M.; PEIXOTO, M.; BOGUTCHI, T. A Evasão no Ensino Superior Brasileiro: o Caso da UFMG. 2003. Disponível em: <http://periodicos.uniso.br/ojs/index.php/avaliacao/article/view/1237>. Acesso em: 07/05/2022.

BRASIL. **Constituição** (1988). **Constituição** da República Federativa do Brasil. Brasília, DF: Senado Federal: Centro Gráfico, 1988.

CHAGAS, Tiago Medina. ANÁLISE DA EVASÃO DOS ALUNOS DOS CURSOS DA UnB: Um estudo no âmbito da graduação. Dissertação (Mestrado em Economia). Universidade de Brasília - UnB, Faculdade de Economia, Administração e Contabilidade. Brasília, DF, 2019.

Chakra, Chakra UI, 2022. Create accessible React apps with speed. Disponível em: <https://chakra-ui.com/>. Acesso em: 01/05/2022.

Fauna. Fauna, 2022. Build applications without limits. Disponível em: <https://fauna.com/>. Acesso em: 02/05/2022.

FILHO, Roberto L. L. S.; MONTEJUNAS, Paulo R.; HIPÓLITO, Oscar; LOBO, Maria B.D.C.M.A evasão no ensino superior brasileiro. Cadernos de Pesquisa, SciELO Brasil, v. 37, n. 132, p. 641–659, 2007. Disponível em <https://www.scielo.br/j/cp/a/x44X6CZfd7hqF5vFNnHhVWg/?format=pdf&lang=pt>. Acesso em 07/05/2022.

FREIRES, Naélio. Um guia para usar ReactJS. Disponível em: <https://blog.geekhunter.com.br/um-guia-para-usar-react-js/>. Acesso em 03/05/2022

INEP, I. N. d. E. e. P. E. A. T. Metodologia de Cálculo dos indicadores de Fluxo da educação superior. 2017. Disponível em: <https://download.inep.gov.br/informacoes_estatisticas/indicadores_educacionais/2017/metodologia_indicadores_trajetoria_curso.pdf>. Acesso em: 01/05/2022.

L., Andrei. O Que é React e Como Funciona?. Disponível em: <https://www.hostinger.com.br/tutoriais/o-que-e-react-javascript>. Acesso em: 01/05/2022.

LOURENÇO, Igor. Chakra UI - Facilidade no front-end JavaScript. Disponível em: <https://medium.com/igor-js/chakra-ui-facilitando-o-front-end-javascript-aabcade75f09>. Acesso em: 01/05/2022.

LÜDER, Amanda. Quase 3,5 milhões de alunos evadiram de universidades privadas no Brasil em 2021. Disponível em: <https://g1.globo.com/educacao/noticia/2022/01/02/quase-35-milhoes-de-alunos-evadiram-de-universidades-privadas-no-brasil-em-2021.ghtml>. Acesso em 03/05/2022.

MENEZES, Tassia; MAIA, Sthefani.Evasão de cursos Universidade: um fenômeno complexo. Conexão UFRJ, 11 de fevereiro de 2022. Disponível <https://conexao.ufrj.br/2022/02/evasao-de-cursos-na-universidade-um-fenomeno-complexo/>. Acesso em 06/05/2022.

Multer, Multer, 2022. Multer is a node.js middleware for handling multipart/form-data, which is primarily used for uploading files. It is written on top of **busboy** for maximum efficiency. Disponível em: <https://www.npmjs.com/package/multer>. Acesso em: 01/05/2022.

MUSSLINER, Bruno O.; MUSSLINER, Monica de S. e S.; MEZA, Edwin B.M.; RODRÍGUEZ, Guillermo L. O problema da evasão universitária no sistema público de ensino superior: uma proposta de ação com base na atuação de uma equipe multidisciplinar. Brazilian Journal of Development, Curitiba, v. 7, n.4, p. 42674-42692, abril de 2021. Disponível em <https://www.brazilianjournals.com/index.php/BRJD/article/download/28957/22870>. Acesso em: 07/05/2022.

NextAuth.js, NextAuth.js, 2022. Authentication for Next.js. Disponível em: <https://next-auth.js.org/providers/github>. Acesso em: 02/05/2022.

Next-connect, next-connect, 2022. Disponível em <https://www.npmjs.com/package/next-connect>. Acesso em: 01/05/2022.

Next.js. Next.js, 2022. What is Next.js?. Disponível em: <https://nextjs.org/learn/foundations/about-nextjs/what-is-nextjs>. Acesso em: 01/05/2022.

Node, Node.js, 2022. About Node.js. Disponível em: <https://nodejs.org/en/about/>. Acesso em: 01/05/2022.

OLIVEIRA, Alexandre V. A. Código da aplicação dvt-next. Disponível em: <https://github.com/alexandrevoliveira/dvt-next>. Acesso em: 06/05/2022.

PRESIDÊNCIA DA REPÚBLICA (Brasil); CONTROLADORIA-GERAL DA UNIÃO - (CGU). Portal da Transparência do Governo Federal, Despesas, Áreas de atuação (funções) do governo, Educação. Disponível em <https://www.portaltransparencia.gov.br/funcoes/12-educacao?ano=2021>. Acesso em 03/05/2022.

React. React, 2022. A JavaScript library for building user interfaces. Disponível em: <https://reactjs.org/>. Acesso em: 01/05/2022.

ROCHA, Alberto. Entendendo Next.js e aplicando suas funcionalidades. Disponível em: <https://blog.geekhunter.com.br/o-que-e-next-js/>. Acesso em: 01/05/2022.

ROCHA, MARIA M.R.D.; LIMA, Alberto S.; ANDRIOLA, Wagner B. Avaliação da evasão discente em cursos de graduação da área de engenharia. Educação & Linguagem · ISSN: 2359-277X · ano 7 · nº 2 · p. 116-146. MAI-AGO. 2020. Disponível em <https://www.fvj.br/revista/wp-content/uploads/2021/02/10_REdLi_2020.2.pdf>. Acesso em 07/05/2022.

RÕÕM, M.; LEPP, M.; LUIK, P. Dropout Time and Learners' Performance in Computer Programming MOOCs. Educ. Sci.2021, 11, 643. Disponível em: <https://www.researchgate.net/publication/355215951_Dropout_Time_and_Learners'_Performance_in_Computer_Programming_MOOCs>. Acesso em 07/05/2022.

ROVEDA, Ugo. REACT: O QUE É, COMO FUNCIONA E PORQUE USAR E COMO APRENDER. Disponível em: <https://kenzie.com.br/blog/react/>. Acesso em: 01/05/2022.

UFPE, Pró-Reitoria de Planejamento, Orçamento e Finanças (PROPLAN). Causas da evasão de alunos dos cursos de graduação presencial da UFPE. Recife, outubro de 2016. Disponível

em
<https://www.ufpe.br/documents/38954/371376/r_evaso_16.pdf/53642e52-41fb-4b43-b098-98db6a470176>. Acesso em 07/05/2022.