UNIVERSIDADE ESTADUAL PAULISTA “JÚLIO DE MESQUITA FILHO”
FACULDADE DE CIÊNCIAS
CAMPUS DE BAURU – SÃO PAULO

GABRIEL AUGUSTO MORELI

# WIDGET DE ACESSIBILIDADE:
# COMO OTIMIZAR O ACESSO DE USUÁRIOS COM DEFICIÊNCIAS OU TRANSTORNOS COGNITIVOS

BAURU
2023

**GABRIEL AUGUSTO MORELI**

# WIDGET DE ACESSIBILIDADE:
# COMO OTIMIZAR O ACESSO DE USUÁRIOS COM DEFICIÊNCIAS OU TRANSTORNOS COGNITIVOS

Trabalho de Conclusão de Curso apresentado à Universidade Estadual Paulista “Júlio de Mesquita Filho”, como exigência parcial para conclusão do curso de graduação de Bacharelado em Sistemas de Informação, da Faculdade de Ciências.

Orientador: Prof.º Jose Remo Ferreira Brega.

**BAURU**

**2023**

Moreli, Gabriel Augusto
M839w Widget de acessibilidade : como otimizar o acesso de usuários com deficiências ou transtornos cognitivos / Gabriel Augusto Moreli. -- Bauru, 2023
60 f. : il.

Trabalho de conclusão de curso (Bacharelado - Sistemas de Informação) - Universidade Estadual Paulista (Unesp), Faculdade de Ciências, Bauru
Orientador: Jose Remo Ferreira Brega

1. Acessibilidade na web. 2. Widget. 3. Deficiências. 4. Inteligência artificial. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca da Faculdade de Ciências, Bauru. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

## AGRADECIMENTOS

Início expressando minha profunda gratidão a Deus, que iluminou meu caminho e me sustentou nos momentos mais desafiadores deste árduo processo, especialmente quando a desistência parecia a única opção. Agradeço imensamente à UNESP, um ambiente que fomenta o saber e acolhe todas as ideias com respeito e interesse.

Um agradecimento especial é dirigido à minha parceira, Bianca Fida Corrêa, cujo apoio inabalável e força incansável foram fundamentais para a realização deste trabalho. Sem ela, esta pesquisa não teria se concretizado.

Aos meus pais, devo minha existência e as inúmeras oportunidades que tive na vida, esperando ansiosamente o dia em que poderei recompensá-los adequadamente.

Quero expressar minha gratidão ao professor Jose Remo Ferreira Brega, cuja orientação e sabedoria foram essenciais na minha jornada acadêmica, iluminando e delineando o caminho a seguir.

Por fim, mas não menos importante, agradeço aos meus amigos e familiares, cujo incentivo e apoio contínuos tornaram esta etapa uma das mais memoráveis e enriquecedoras da minha vida.

## RESUMO

As dificuldades enfrentadas por usuários com deficiências ou transtornos cognitivos ao acessar a rede como um todo ainda são muitas. A leitura de conteúdos em texto e a interação com a interface de *sites web* ainda são pouco acessíveis ou confortáveis para esses grupos de pessoas. Pensando em uma experiência de usuário mais inclusiva e completa, este trabalho buscou desenvolver um *widget* que foca nas necessidades específicas de leitura e interação de pessoas com deficiências ou transtornos cognitivos ao navegar na web. Dessa forma, a pesquisa se iniciou com uma análise das dificuldades específicas enfrentadas por cada grupo de usuários; dentre eles, pessoas com diferentes níveis de deficiência auditiva, dislexia, transtorno de déficit de atenção, daltonismo, entre outros. Com essas necessidades específicas em mente, este trabalho buscou desenvolver um *widget* cujas funções incluíssem soluções já provadas eficazes. Essas soluções, ao serem aplicadas, alteram o *layout* e a fonte dos conteúdos de *sites web*, facilitando e otimizando a experiência desses usuários, tornando-a, portanto, mais acessível.



Palavras-chave: acessibilidade, web, deficiência, widget, tecnologia, inclusão digital

## ABSTRACT

The struggles faced by users with disabilities or cognitive disorders when navigating the web are still numerous. Reading text content and interacting with the web interface are still not very accessible or comfortable for these groups of people. With the goal of creating a more inclusive and comprehensive user experience, this work developed a widget that focuses on the specific reading and interaction needs of people with disabilities or cognitive disorders when browsing the web. The research began by analyzing the specific challenges faced by each user group, including people with different levels of hearing impairment, dyslexia, attention deficit disorder, color blindness, among others. With these specific needs in mind, this work sought to develop a widget featuring solutions that have been previously proven effective. When applied, these solutions alter the layout and the font of web content, making the experience of these users easier and more optimized, and therefore, more accessible.



Keywords: accessibility, web, disabilities, widget, technology, digital inclusion

## LISTA DE ILUSTRAÇÕES

## LISTA DE TABELAS

**SUMÁRIO**

# 1 INTRODUÇÃO

Nos últimos anos, a sociedade tem testemunhado avanços significativos na tecnologia, especialmente no que diz respeito à acessibilidade digital. No entanto, mesmo com esses avanços, a inclusão de pessoas com deficiências e transtornos cognitivos no ambiente online continua sendo um desafio premente.

De acordo com a Organização Mundial da Saúde (OMS), estima-se que aproximadamente 1,3 bilhões de pessoas, cerca de 16% da população mundial, possui algum tipo de deficiência. Esse número, segundo a Organização, só tende a crescer (OMS, 2023). Só no Brasil, há cerca de 18,6 milhões de pessoas com deficiência, segundo levantamento do Instituto Brasileiro de Geografia e Estatística (IBGE), através da Pesquisa Nacional de Amostra de Domicílios Contínua (PNAD Contínua) de 2022 (IBGE, 2022).

Ainda de acordo com o IBGE, os tipos de deficiência prevalentes na população brasileira são as motoras, visuais e de cognição. Dos indivíduos com deficiências que podem comprometer sua acessibilidade digital, estima-se que: 3,1% da população tenha dificuldade para enxergar; 2,6% para aprender, lembrar-se das coisas ou se concentrar; 1,2% para ouvir mesmo com aparelhos auditivos e; 1,1% para se comunicar, compreender e ser compreendido (IBGE, 2022).

Para os indivíduos com deficiências visuais, auditivas, motoras ou transtornos cognitivos, a barreira digital é acentuada. A dificuldade de navegação, compreensão e interação com conteúdo online impacta diretamente a participação desses indivíduos na sociedade digital.

Nesse contexto, a acessibilidade na *web* desempenha um papel crucial. O termo “acessibilidade”, no contexto da Internet, também chamado “acessibilidade digital”, refere-se a uma apresentação de informações que seja acessível a todas as pessoas, independentemente das capacidades sensoriais e funcionais de cada usuário (NUNES, 2022). No entanto, infelizmente, estatísticas revelam que a experiência online para muitos indivíduos com deficiências é frequentemente limitada e frustrante. De acordo com o estudo mais recente da WebAIM em 2023, que avaliou a acessibilidade das páginas iniciais dos top 1.000.000 sites, foram detectados um total de 49.991.225 erros de acessibilidade distintos, resultando em uma média de 50,0 erros por página. Esse número representa uma diminuição ligeira (1,6%) em

comparação com a análise de 2022, que encontrou uma média de 50,8 erros por página. Os 'erros' são barreiras de acessibilidade detectadas pelo WAVE (*Web Accessibility Evaluation Tool*), ferramenta online usada para ajudar na avaliação da acessibilidade de websites, com impacto notável no usuário final. (WebAIM, 2023)

Diante desse cenário, a implementação de soluções específicas se torna fundamental. É nesse contexto que se destaca a importância de um *widget* de acessibilidade, ferramenta que visa superar as barreiras digitais ao oferecer recursos adaptativos e personalizáveis, facilitando o acesso e a utilização de plataformas online para esse público.

O presente trabalho foi desenvolvido após observação da falta de acessibilidade digital presente na plataforma de cursos online da empresa YesLMS. Ao notar que plataformas similares apresentavam *widgets* e soluções para as necessidades específicas desses indivíduos, o desenvolvimento de um *widget* similar foi proposto para a plataforma da YesLMS, a fim de torná-la o mais acessível possível com os recursos disponíveis, mitigando as dificuldades encontradas e promovendo uma experiência inclusiva em seu ambiente virtual.

O *widget* foi desenvolvido utilizando um conjunto diversificado de tecnologias modernas e eficazes. Centralmente, o *widget* foi criado com o Angular 16, aproveitando a tecnologia Angular Elements para garantir compatibilidade e reutilização em vários *frameworks* JavaScript, incluindo React, Vue, e até mesmo JavaScript puro (Vanilla JS). Foi empregado o Webpack para compilação e a eficiência do arquivo accessibility-widget.js foi aprimorada com a compressão via Gzipper, sendo disponibilizado através de um *blob* público no Azure. Adicionalmente, foi integrada uma chamada REST a uma API desenvolvida em .NET Core 7, essencial para a interação com a API da OpenAI, assegurando assim uma camada robusta de segurança e funcionalidades avançadas para o *widget* de acessibilidade.

Este trabalho está dividido da seguinte maneira: apresentação dos *widgets* já existentes no mercado e suas soluções de acessibilidade; as tecnologias utilizadas para o desenvolvimento do *widget* objeto deste trabalho; o resultado do *widget* aqui desenvolvido e suas soluções de acessibilidade; o diagrama de atividades do *widget*; os resultados preliminares da implementação do *widget*; e conclusão.

# 2 SOLUÇÕES EXISTENTES

No momento, já existem *widgets* que buscam solucionar as necessidades específicas de pessoas com deficiências e transtornos cognitivos. No entanto, esses *widgets* não apresentam o mesmo conjunto de funcionalidades e, portanto, abrangência, que o *widget* desenvolvido neste trabalho.

Alguns dos *widgets* já existentes no mercado são:

## 2.1 USERWAY

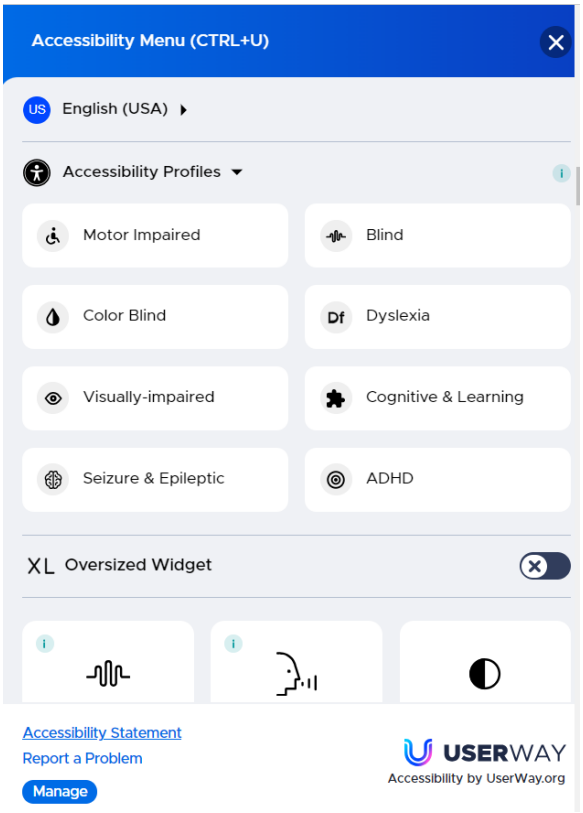
A UserWay foi fundada em setembro de 2016 por Allon Mason, impulsionada pela visão de simplificar e democratizar a acessibilidade digital para empresas e organizações. Lançando seu *widget* de acessibilidade em 2017, a UserWay rapidamente se destacou como uma plataforma dedicada a tornar a *web* mais inclusiva e acessível.

Desde seu lançamento, o *widget* de acessibilidade da UserWay tem ampliado sua presença e se adaptado para enfrentar novos desafios e demandas no campo da acessibilidade digital. A plataforma busca oferecer soluções abrangentes para uma série de deficiências e transtornos cognitivos.

Atualmente, o aplicativo UserWay aborda uma variedade de condições, incluindo deficiência motora, visual, cegueira, daltonismo, dislexia, transtornos cognitivos e de aprendizagem, transtorno de déficit de atenção e hiperatividade, assim como convulsões e epilepsia.

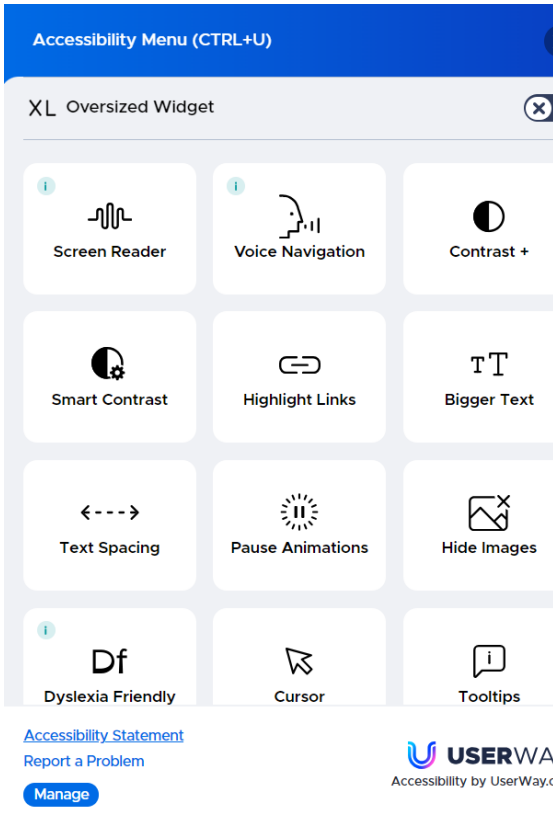
A Figura 1 apresenta os perfis de acessibilidade do *widget* da UserWay, enquanto a Figura 2 apresenta suas funcionalidades.

**Figura 1:** Perfis de Acessibilidade – Widget UserWay




Fonte: https://userway.org/

**Figura 2:** Funcionalidades – Widget UserWay




Fonte: https://userway.org/

## 2.2 ACCESSIBE

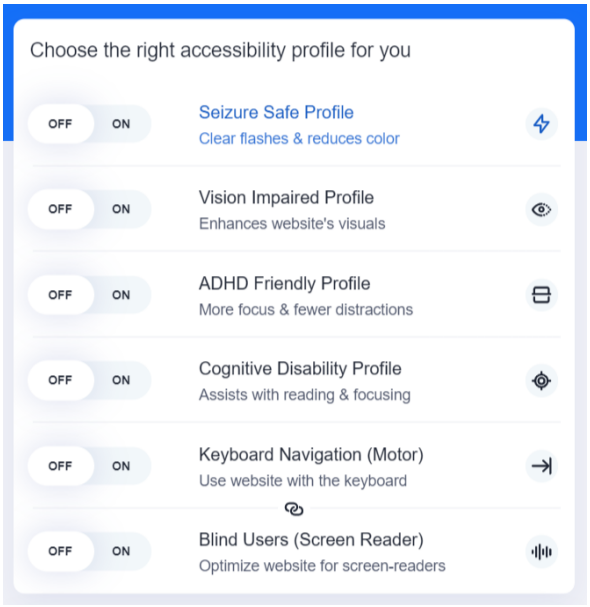
O accessiBe é uma plataforma comprometida com a acessibilidade digital, fundada em 2019 com a visão de tornar a internet inclusiva para todos. Desde sua criação, a plataforma tem sido pioneira na oferta de soluções abrangentes para a acessibilidade na *web*. Lançado com esse propósito, seu *widget* de acessibilidade é uma ferramenta poderosa que tem evoluído constantemente para atender às necessidades em constante mudança do cenário digital.

O aplicativo accessiBe, desde sua introdução, expandiu suas funcionalidades para abranger uma ampla gama de deficiências e transtornos cognitivos. A plataforma se destaca por oferecer soluções inclusivas para diversas condições, algumas das quais incluem deficiência motora, visual, cegueira, daltonismo, dislexia, transtornos cognitivos e de aprendizagem, transtorno de déficit de atenção e hiperatividade, bem como para convulsões e epilepsia.

Ao longo do tempo, o accessiBe tem se empenhado em aprimorar suas capacidades e se manter atualizado com as diretrizes de acessibilidade, proporcionando uma experiência acessível e adaptativa para todos os usuários na web.

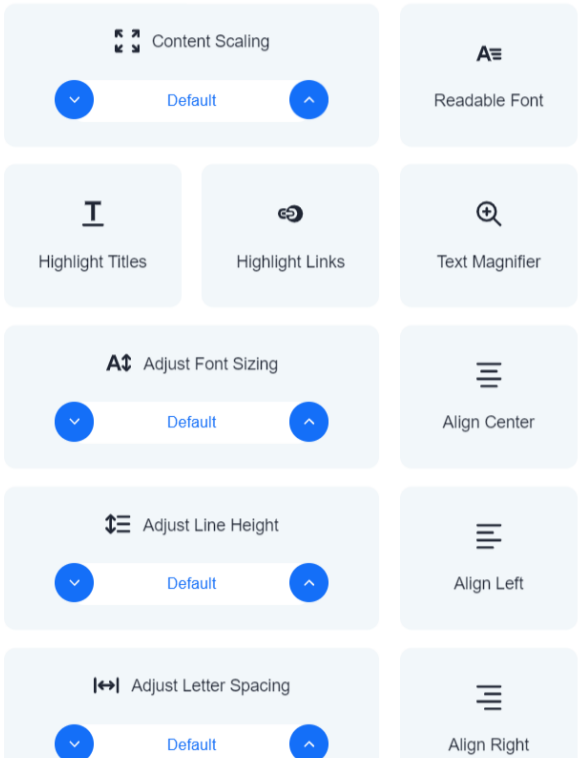
A Figura 3 apresenta os perfis de acessibilidade do *widget* da AccessiBe, a Figura 4 apresenta seus ajustes de conteúdo, a Figura 5 apresenta seus ajustes de cor e a Figura 6 traz seus ajustes de orientação.

**Figura 3:** Perfis de Acessibilidade – Widget accessiBe



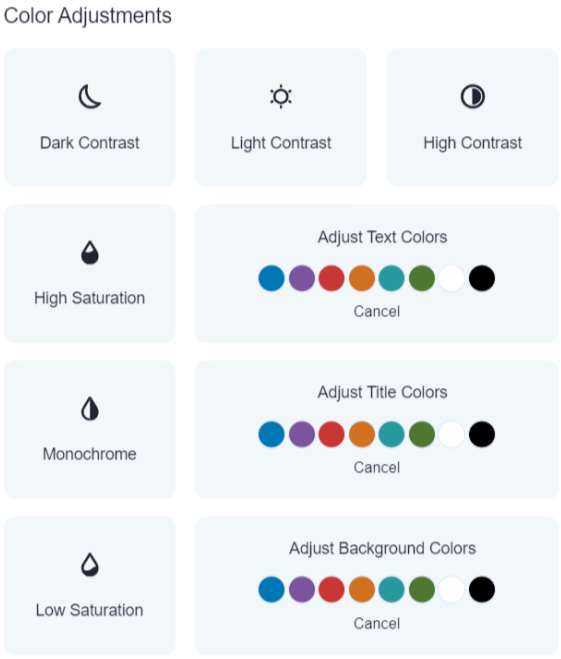

Fonte: https://accessibe.com/

**Figura 4:** Ajustes de Conteúdo – Widget accessiBe



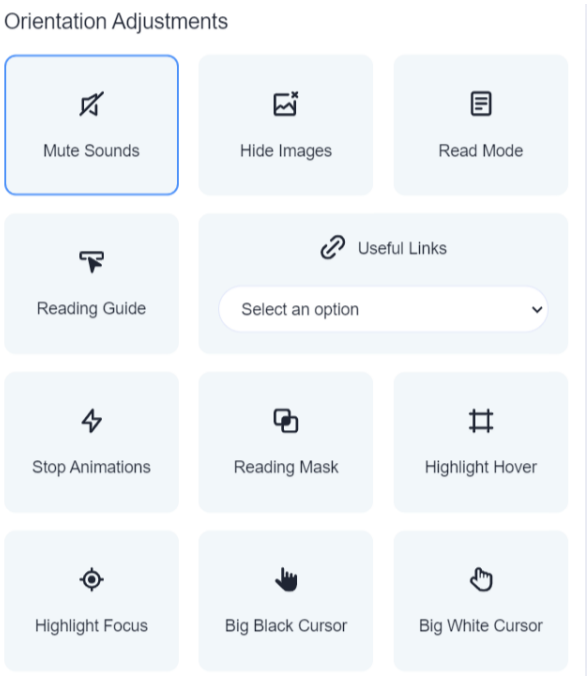

Fonte: https://accessibe.com/

**Figura 5:** Ajustes de Cor – Widget accessiBe




Fonte: https://accessibe.com/

**Figura 6:** Ajustes de Orientação – Widget accessiBe




Fonte: https://accessibe.com/

# 3 TECNOLOGIAS UTILIZADAS

## 3.1 ANGULAR 16 E ANGULAR ELEMENTS

O cerne do desenvolvimento do *widget* de acessibilidade foi baseado no Angular 16, um *framework* robusto e moderno para aplicações *web*. A escolha desse framework foi devido à sua capacidade de oferecer uma estrutura escalável e eficiente para o desenvolvimento de aplicações ricas em recursos. Utilizando as funcionalidades do Angular 16, foi possível criar um *widget* dinâmico e interativo, garantindo uma excelente experiência ao usuário.

Para integrar o *widget* em diferentes aplicações web, utilizou-se a tecnologia Angular Elements. Essa abordagem permitiu encapsular o componente do *widget* como um elemento HTML personalizado, tornando-o compatível com diversos ambientes de aplicação, como React, Vue, outros *frameworks* JavaScript ou até mesmo JavaScript puro (Vanilla JS). A versatilidade dos Angular Elements é fundamental para assegurar que o *widget* possa ser facilmente integrado em uma variedade de plataformas web sem restrições de compatibilidade.

## 3.2 CONCEITO E BENEFÍCIOS DO ANGULAR ELEMENTS

Há diversos benefícios advindos da utilização do Angular Elements. Entre eles, estão:

**Web Components:** Angular Elements são essencialmente *Web Components*, uma suíte de diferentes tecnologias que permitem criar elementos HTML personalizados. O Angular Elements encapsula a lógica e a visualização de um componente Angular dentro de um elemento da *web*, tornando-o reutilizável em qualquer ambiente que suporte HTML.

**Isolamento e Reutilização:** Um dos maiores benefícios dos Angular Elements é o isolamento que eles fornecem. Eles encapsulam a lógica e o estilo, evitando conflitos com o DOM externo ou estilos globais. Isso permite que os mesmos componentes sejam reutilizados em diferentes partes de uma aplicação ou em diferentes aplicações sem a preocupação de conflitos de estilo ou de comportamento.

**Interoperabilidade:** Angular Elements é ideal para desenvolvedores que desejam criar componentes que podem ser facilmente integrados em aplicações que não utilizam Angular como *framework* principal. Isso abre oportunidade para uma maior interoperabilidade entre sistemas.

## 3.3 IMPLEMENTAÇÃO DE ANGULAR ELEMENTS

O primeiro passo para a utilização do Angular Elements é a criação de um componente Angular convencional. Esse componente será posteriormente empacotado como um Elemento Angular.

Para transformar um componente Angular em um elemento da *Web*, é necessário incluir o pacote @angular/elements no projeto. Isso pode ser feito via NPM com o comando npm install @angular/elements.

Após a criação do componente, é necessário registrá-lo como um elemento personalizado no módulo principal (geralmente app.module.ts). Isso é feito utilizando a API createCustomElement do pacote @angular/elements.

**Figura 7:** Definição do Elemento na classe AppModule

```
import { createCustomElement } from '@angular/elements';
// ... outras importações

@NgModule({
  // ... configurações do módulo
})
export class AppModule {
  constructor(injector: Injector) {
    const customElement = createCustomElement(SeuComponente, { injector });
    customElements.define('nome-do-seu-elemento', customElement);
  }

  ngDoBootstrap() {}
}
```

Fonte: Exemplo capturado em ambiente local utilizando editor VS Code.

Aqui, SeuComponente é o componente Angular que será exposto como um elemento da *Web* e nome-do-seu-elemento é a *tag* que será usada para esse elemento no HTML.

Uma vez que o componente Angular é convertido em um elemento da *Web*, ele pode ser usado em qualquer projeto HTML, independentemente do *framework*. Isso é feito simplesmente adicionando a *tag* do elemento (por exemplo, <nome-do-seu-elemento></nome-do-seu-elemento>) no HTML e incluindo o *script* gerado pelo Angular.

## 3.4 COMPILAÇÃO E DISTRIBUIÇÃO

A compilação dos arquivos JavaScript do projeto foi realizada utilizando o Webpack, uma ferramenta para o processamento de módulos. O Webpack foi empregado para compilar todos os arquivos JS em um único arquivo denominado accessibility-widget.js. Esse processo de compilação é crucial para otimizar o desempenho do *widget*, reduzindo o tempo de carregamento e a complexidade na gestão dos *scripts*.

Posteriormente, o arquivo compilado foi comprimido utilizando Gzipper, uma ferramenta eficiente para a compressão de arquivos. A compressão Gzip é uma prática recomendada para reduzir o tamanho dos arquivos, acelerando o tempo de transferência de dados e diminuindo a carga nos servidores.

O arquivo comprimido foi então disponibilizado através de um *blob* público no Azure, uma plataforma de nuvem da Microsoft. Isso proporciona uma distribuição global e confiável, permitindo que os usuários acessem e integrem o *widget* em seus sites de maneira eficiente, seja fazendo referência direta ao arquivo no *blob* do Azure ou baixando o arquivo JS para referência local.

## 3.5 INTEGRAÇÃO COM API REST E OPENAI

Além das funcionalidades padrão, o *widget* realiza uma chamada REST para uma API desenvolvida em .NET Core 7. Essa chamada é essencial para uma das funcionalidades do *widget* que requer interação com a API da OpenAI. A utilização do .NET Core 7 oferece um desempenho elevado e uma integração segura com serviços

de terceiros, como a OpenAI, garantindo que o *widget* não apenas atenda às necessidades de acessibilidade, mas também incorpore funcionalidades avançadas e inovadoras.

A utilização de uma API *backend* para intermediar chamadas à API da OpenAI no desenvolvimento do *widget* de acessibilidade é uma decisão estratégica, principalmente por questões de segurança e gerenciamento de recursos. Este trabalho ainda explorará os motivos e benefícios dessa abordagem em mais detalhas.

## 3.6 SEGURANÇA DAS CHAVES DE API

Ao integrar diretamente a API da OpenAI em um aplicativo cliente (*frontend*), as chaves de API necessárias para autenticação ficariam expostas no código do cliente, tornando-as acessíveis a qualquer usuário que inspecione o código fonte. Em contraste, ao usar uma API *backend*, as chaves de API podem ser armazenadas com segurança no servidor, longe dos olhos do público. Isso é crucial, especialmente para APIs pagas como a OpenAI, onde o uso não autorizado das chaves poderia levar a custos indesejados e potenciais abusos.

Através de uma API *backend*, é possível implementar controles de acesso mais robustos. Por exemplo, pode-se configurar regras para limitar a frequência de chamadas à API da OpenAI ou restringir o acesso a certas funcionalidades com base no usuário ou no contexto. Isso ajuda a prevenir o uso excessivo ou indevido da API, garantindo que o *widget* permaneça eficiente e econômico.

Em um cenário *backend*, é possível implementar sistemas de rotação e atualização de chaves sem afetar o cliente. Isso significa que, em caso de uma chave ser comprometida, ela pode ser rapidamente alterada no servidor sem a necessidade de atualizações no lado do cliente.

## 3.7 EFICIÊNCIA E PERFORMANCE

Utilizar uma API *backend* permite a implementação de um sistema de cache para as respostas da API da OpenAI. Isso significa que, para consultas frequentes ou repetidas, o *widget* pode fornecer respostas quase instantâneas, melhorando a experiência do usuário e reduzindo o número de chamadas pagas à API da OpenAI.

Em relação ao balanceamento de carga, é mais fácil implementar estratégias para tal em um ambiente *backend*. Isso é especialmente útil quando se lida com um número significativo de usuários, garantindo que as chamadas para a API da OpenAI sejam gerenciadas de forma eficiente, evitando sobrecargas e mantendo a estabilidade do sistema.

## 3.8 CONFORMIDADE E AUDITORIA

Manter um registro de todas as chamadas feitas à API da OpenAI através do *backend* permite uma auditoria mais eficaz. Esses registros podem ser usados para monitorar o uso, identificar padrões anormais e fornecer *insights* valiosos sobre como o *widget* está sendo utilizado.

Ademais, ao lidar com dados sensíveis ou informações do usuário, a API *backend* pode assegurar que todas as transações estejam em conformidade com leis de privacidade e proteção de dados, como o GDPR na União Europeia ou a LGPD no Brasil.

Em resumo, a utilização de uma API *backend* para gerenciar a interação com a API da OpenAI não só fornece uma camada adicional de segurança para as chaves de API, como também oferece benefícios significativos em termos de eficiência operacional, controle de acesso, conformidade regulatória e possibilidades de auditoria. Esses fatores são essenciais para garantir a integridade, a segurança e a sustentabilidade do *widget* de acessibilidade em um ambiente de produção.

# 4 O WIDGET

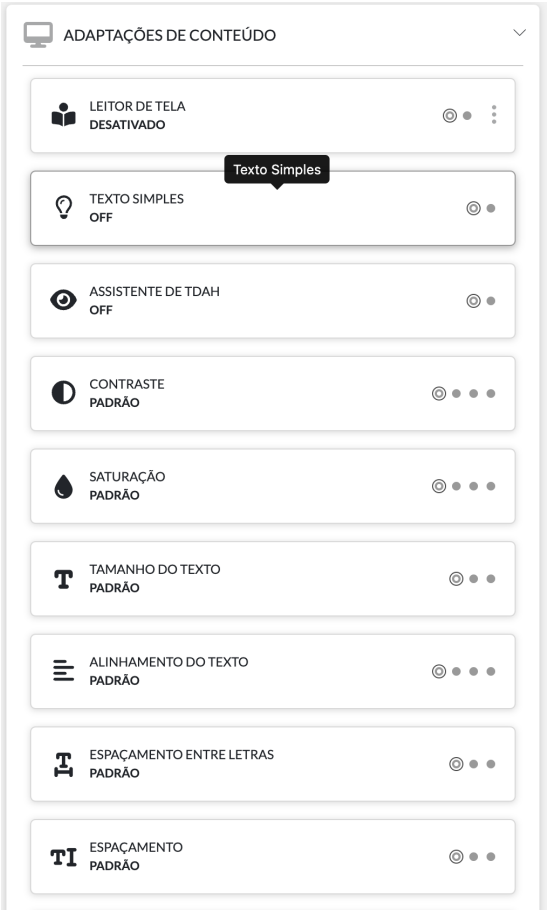
## 4.1 DESCRIÇÃO

O *widget* de acessibilidade é uma ferramenta abrangente projetada para aprimorar a experiência do usuário. Ele consiste em cinco seções principais: Adaptações de Conteúdo, Opções de Navegação, Perfis de Acessibilidade, Dicionário e Configurações.

## 4.2 ADAPTAÇÕES DE CONTEÚDO

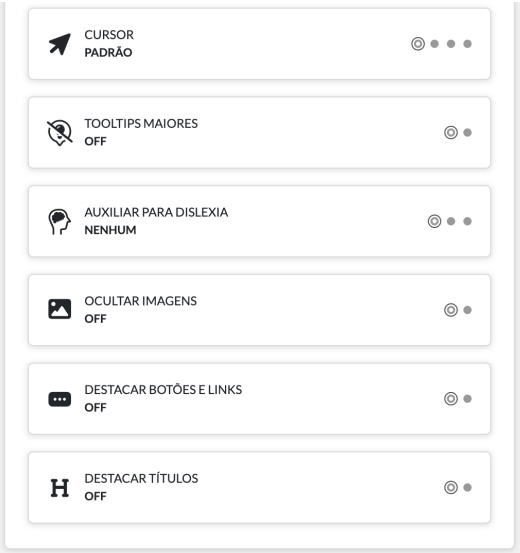
A seção inicial do *widget* é a área de Adaptações de Conteúdo. As Figuras 8 e 9 trazem as possibilidades de otimização de experiência disponíveis aos usuários.

**Figura 8:** Adaptações de Conteúdo #1 – YesLMS Widget




Fonte: https://test-vrdg.yeslms.dev

**Figura 9:** Adaptações de Conteúdo #2 – YesLMS Widget




Fonte: https://test-vrdg.yeslms.dev

Cada adaptação possui um número de variações. Ao oferecer essas adaptações, o *widget* garante que os usuários possam personalizar sua experiência de acordo com suas necessidades e preferências individuais.

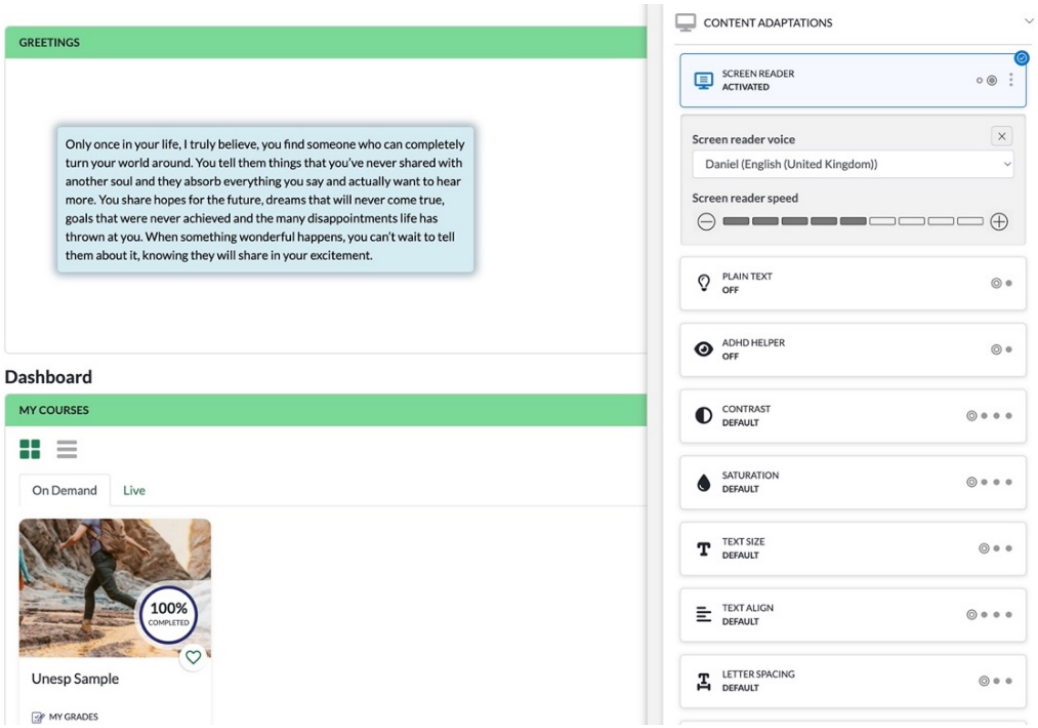
A seguir, cada adaptação será explorada em mais detalhes.

### 4.2.1 Leitor de Tela

O Leitor de Tela é uma tecnologia assistiva que lê em voz alta o texto na tela. Ele é projetado para ajudar usuários com deficiência visual ou aqueles que preferem aprendizado auditivo. O Leitor de Tela deste *widget* tem três configurações de velocidade: normal, rápida e lenta. Os usuários podem escolher a velocidade que se adequa ao seu ritmo de audição e compreensão. Além disso, o Leitor de Tela pode ser navegado usando atalhos de teclado predefinidos, permitindo que os usuários se movam pelo sistema com facilidade. Ele pode ler componentes selecionados, proporcionando um entendimento abrangente do conteúdo. Este recurso também é benéfico para usuários que podem ter dificuldade em ler textos devido à dislexia ou outras dificuldades de aprendizagem.

A Figura 10 apresenta o Leitor de Tela do *widget*.

**Figura 10:** Leitor de Tela – Widget YesLMS

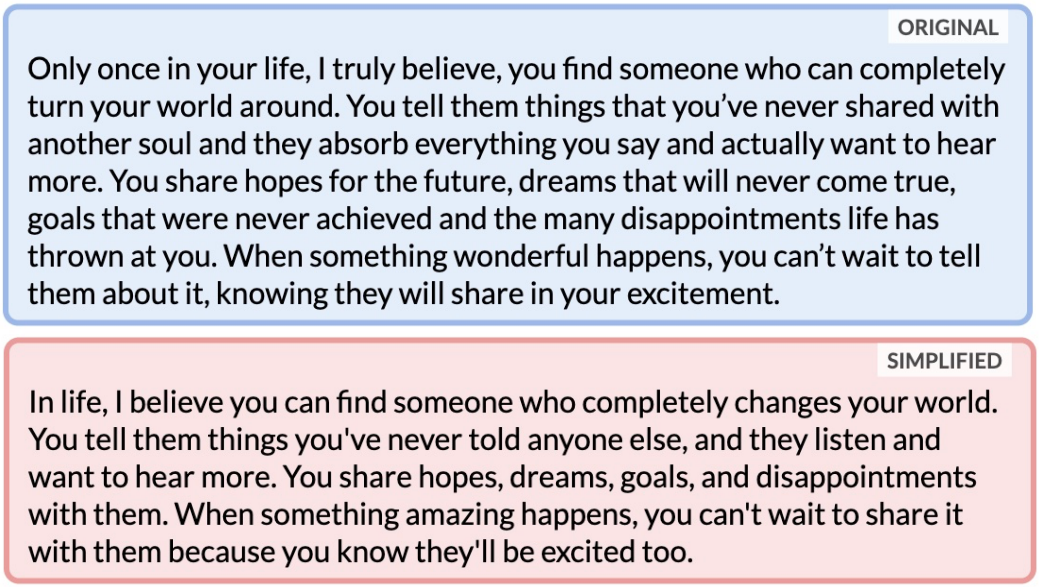



Fonte: https://test-vrdg.yeslms.dev

### 4.2.2 Simplificação de Texto

Ao ativar a simplificação de texto, textos longos e complicados são trocados por textos simplificados tanto em sua forma quanto em seu conteúdo, através do uso de inteligência artificial. A Figura 11 apresenta o parágrafo de um texto de um curso da plataforma antes e depois de sua simplificação.

**Figura 11:** Simplificação de Texto – Widget YesLMS

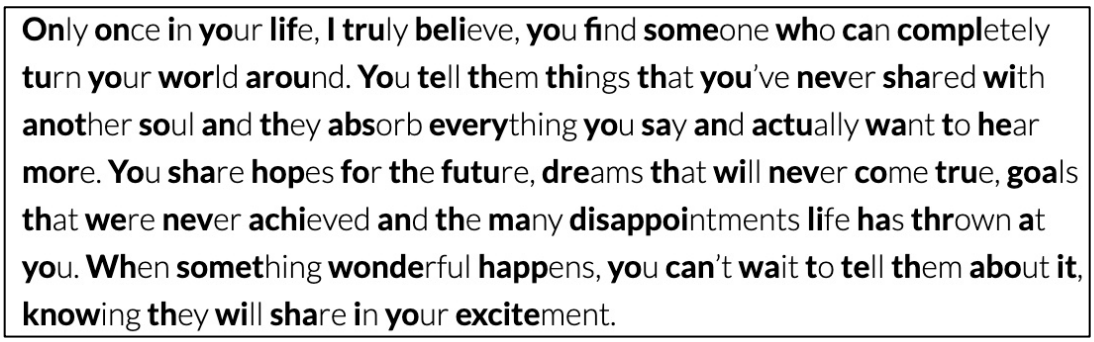



Fonte: https://test-vrdg.yeslms.dev

### 4.2.3 Assistente de TDAH

O Assistente de TDAH deixa em negrito um determinado número de caracteres de cada palavra para facilitar a visualização de pessoas com TDAH. A Figura 12 apresenta essa funcionalidade em uso.

**Figura 12:** Assistente de TDAH – Widget YesLMS

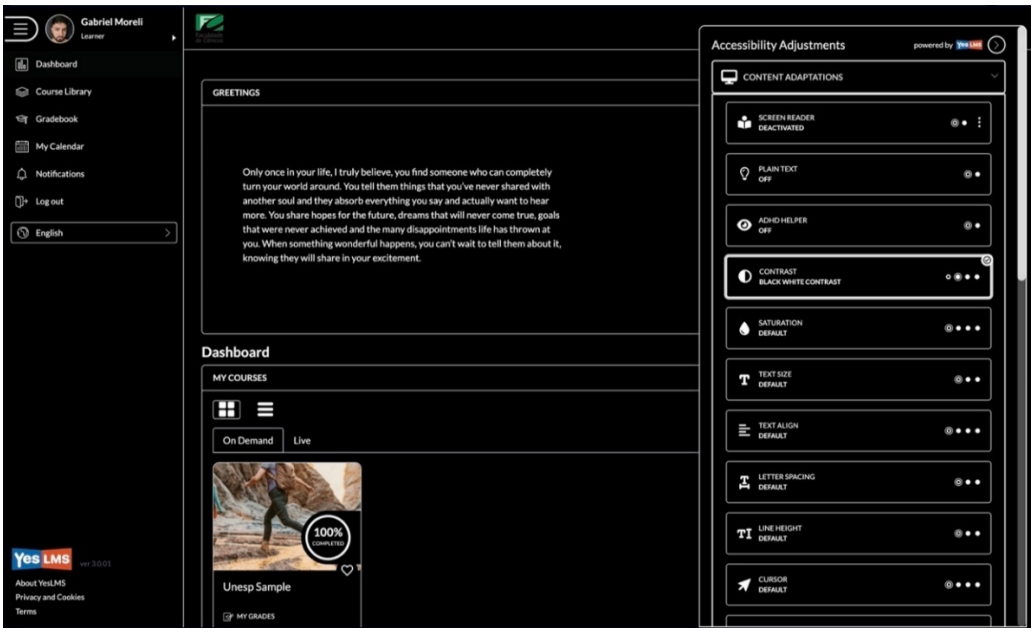



Fonte: https://test-vrdg.yeslms.dev

### 4.2.4 Alto Contraste

Os usuários podem escolher entre contraste preto/branco, preto/amarelo ou azul/amarelo. A Figura 13 traz a aparência da página com contraste preto/branco.

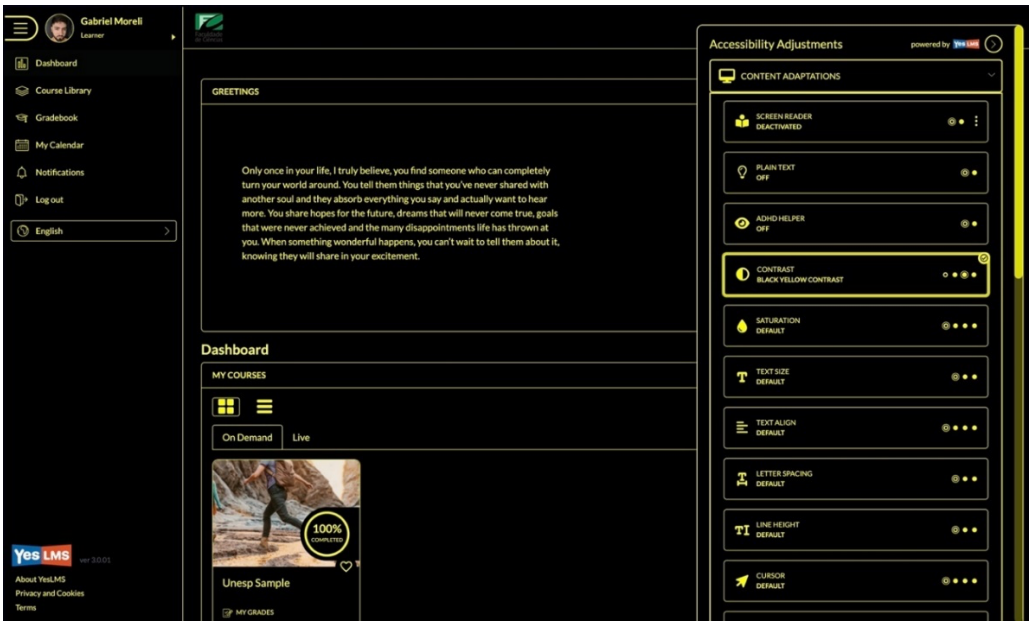
**Figura 13:** Contraste Preto e Branco – Widget YesLMS




Fonte: https://test-vrdg.yeslms.dev

A Figura 14 apresenta a aparência da página em contraste preto/amarelo.

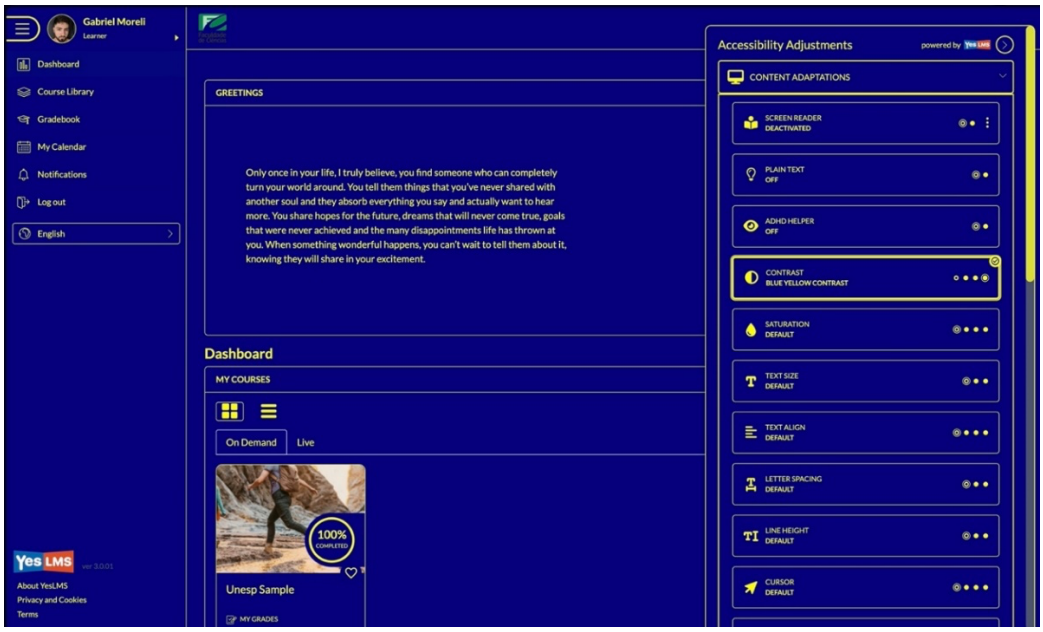
**Figura 14:** Contraste Preto e Amarelo – Widget YesLMS




Fonte: https://test-vrdg.yeslms.dev

E, por fim, a Figura 15 apresenta a aparência da página em contraste azul/amarelo.

**Figura 15:** Contraste Azul e Amarelo – Widget YesLMS

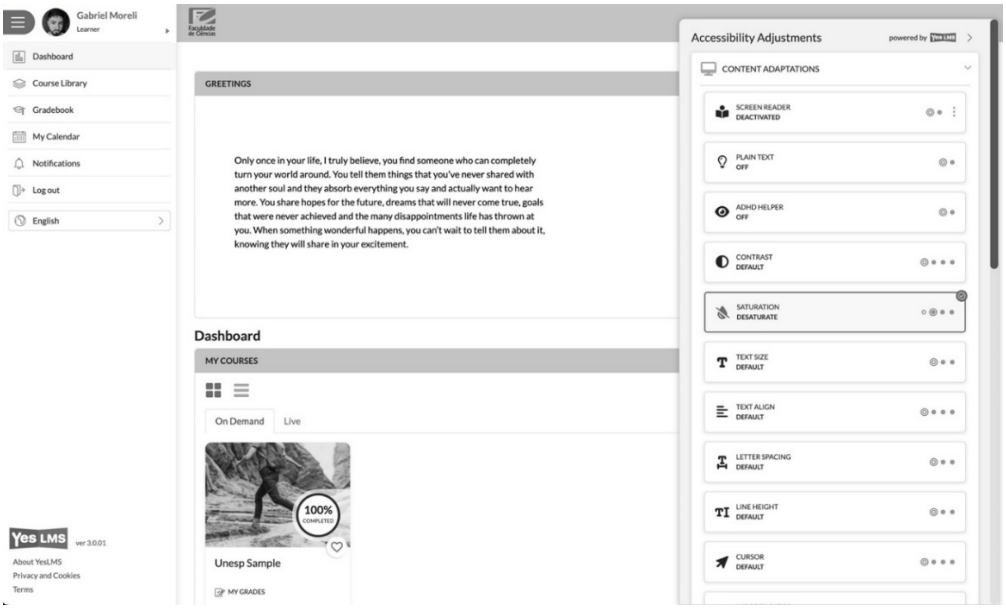



Fonte: https://test-vrdg.yeslms.dev

### 4.2.5 Saturação

Os usuários podem definir a saturação do conteúdo como dessaturada, baixa ou alta. A Figura 16 apresenta a aparência da página em dessaturação.

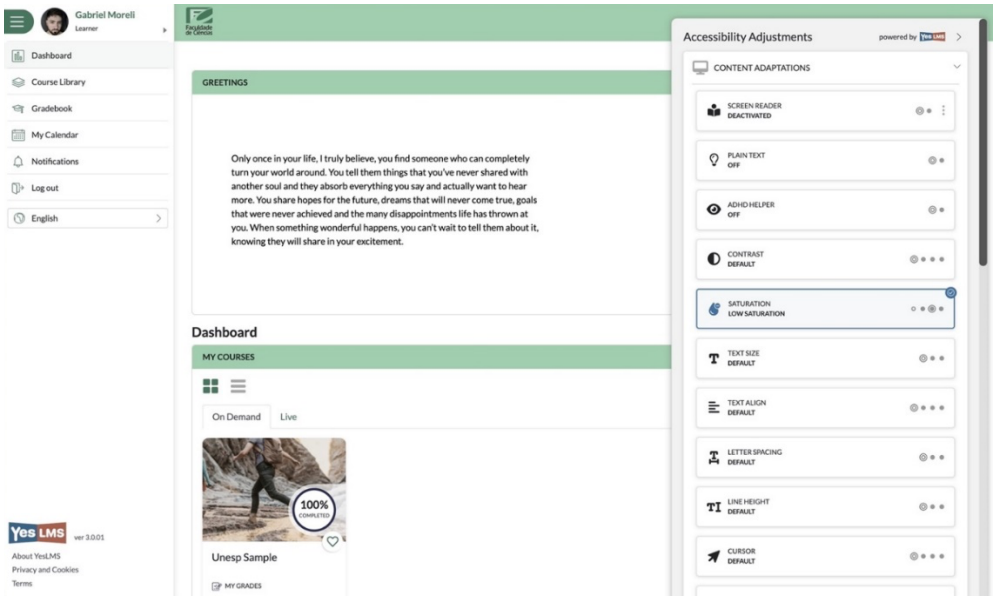
**Figura 16:** Página em Dessaturação – Widget YesLMS




Fonte: https://test-vrdg.yeslms.dev

A Figura 17 traz a aparência da página em baixa saturação.

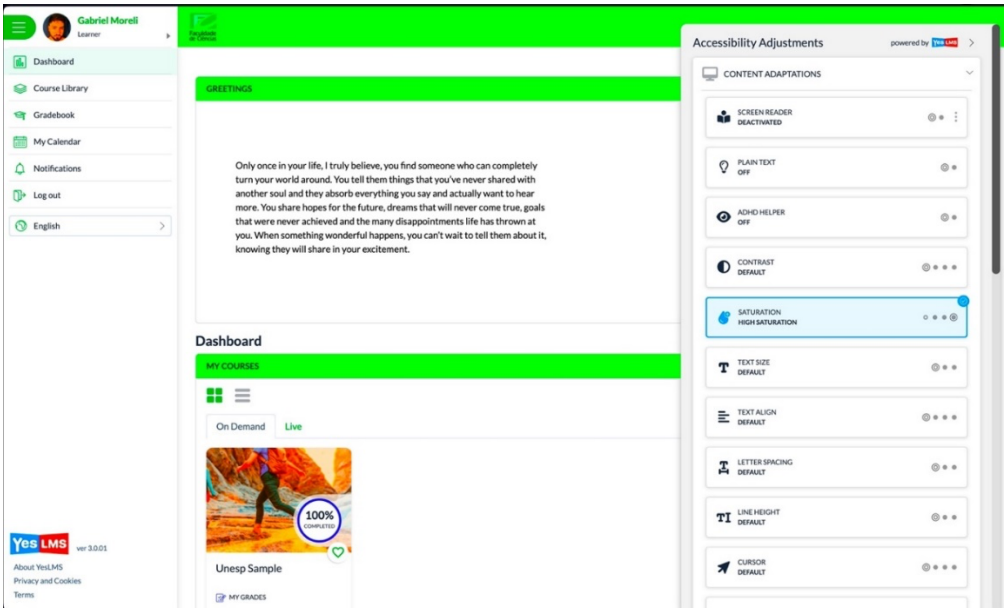
**Figura 17:** Baixa Saturação – Widget YesLMS




Fonte: https://test-vrdg.yeslms.dev

E, por fim, a Figura 18 traz a aparência da página em alta saturação.

**Figura 18:** Alta Saturação – Widget YesLMS

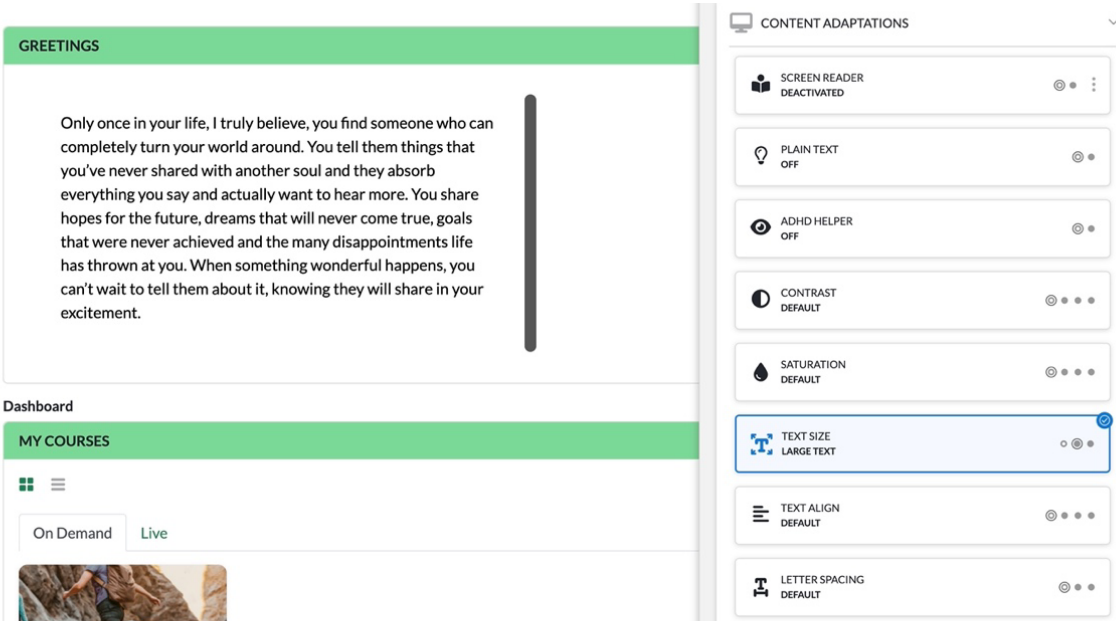



Fonte: https://test-vrdg.yeslms.dev

### 4.2.6 Tamanho do Texto

Os usuários podem aumentar o tamanho do texto para grande ou extragrande. A Figura 19 apresenta a página com texto em tamanho grande.

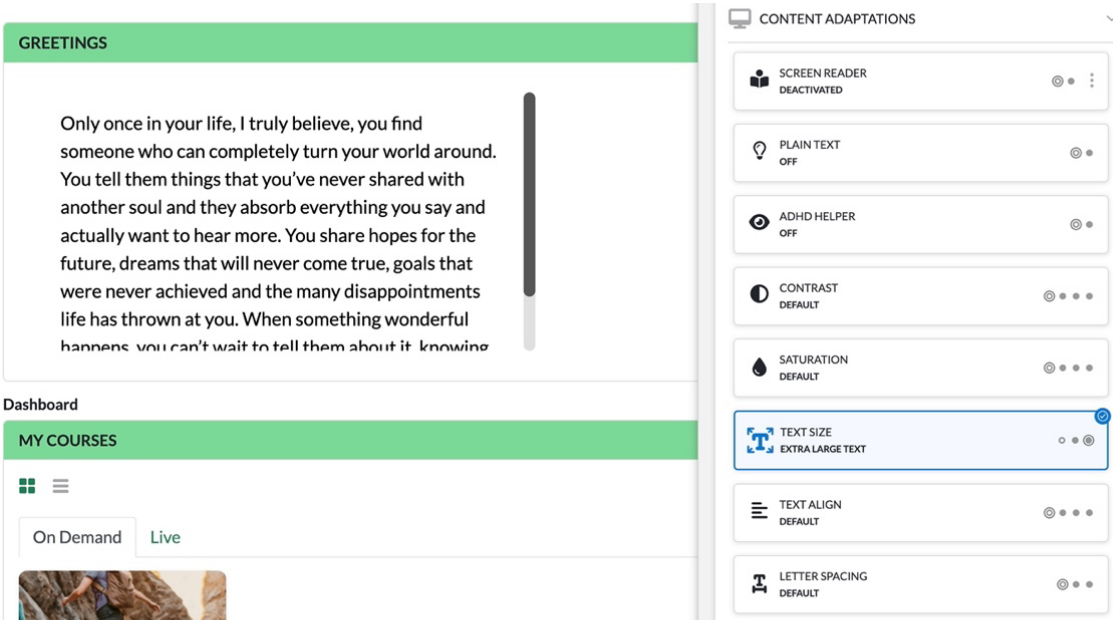
**Figura 19:** Texto Grande – Widget YesLMS




Fonte: https://test-vrdg.yeslms.dev

A Figura 20 apresenta a página com texto em tamanho extragrande.

**Figura 20:** Texto Extragrande – Widget YesLMS

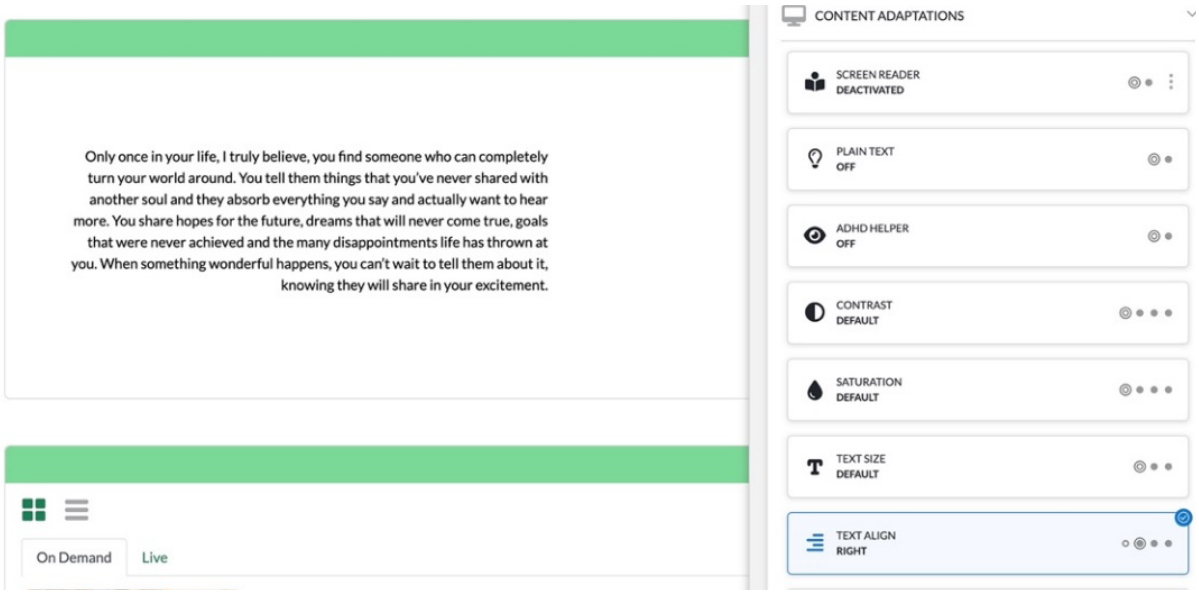



Fonte: https://test-vrdg.yeslms.dev

### 4.2.7 Alinhamento de Texto

Os usuários podem alinhar o texto à direita, à esquerda, ao centro ou justificá-lo. A Figura 21 traz o texto alinhado à direita.

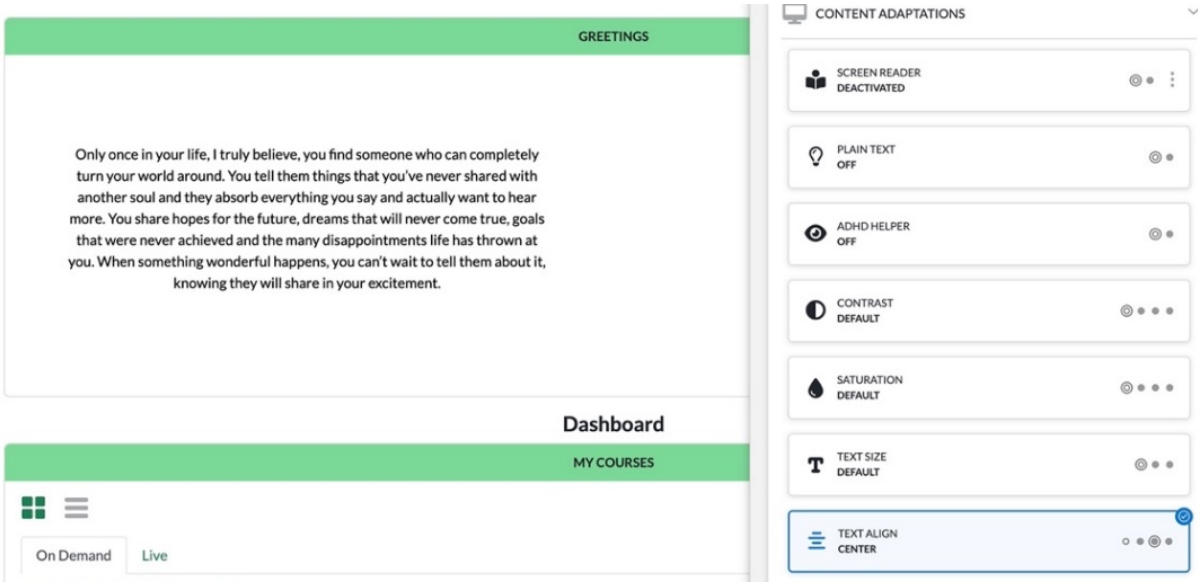
**Figura 21:** Texto à Direita – Widget YesLMS




Fonte: https://test-vrdg.yeslms.dev

A Figura 22 apresenta o texto centralizado.

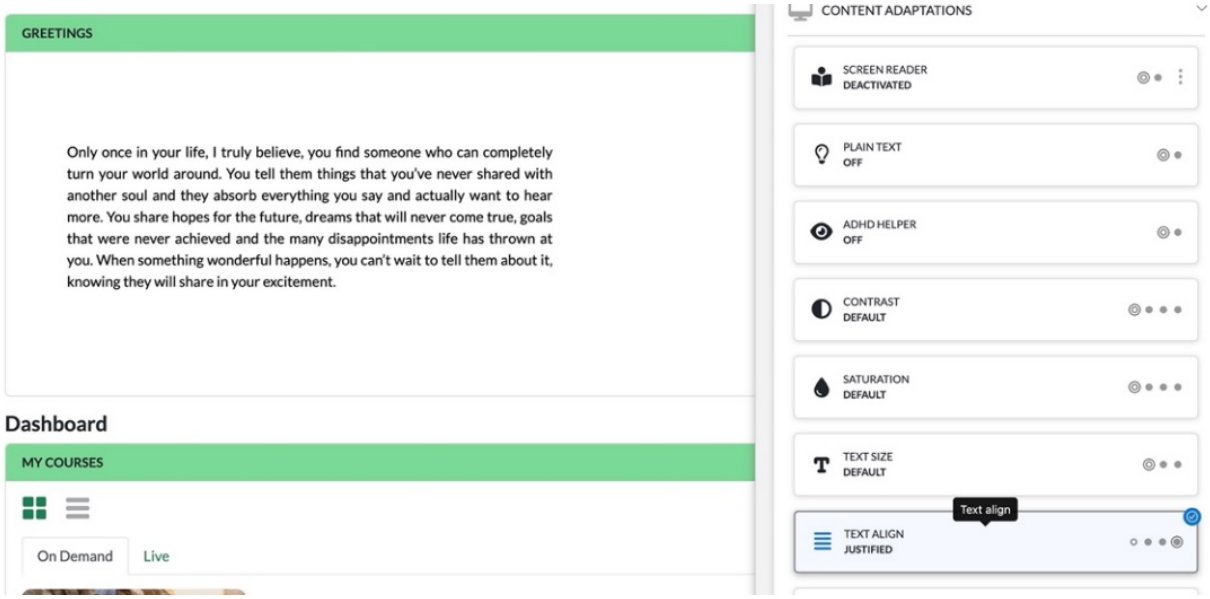
**Figura 22:** Texto no Centro – Widget YesLMS




Fonte: https://test-vrdg.yeslms.dev

E, por fim, a Figura 23 apresenta o texto justificado.

**Figura 23:** Texto Justificado – Widget YesLMS

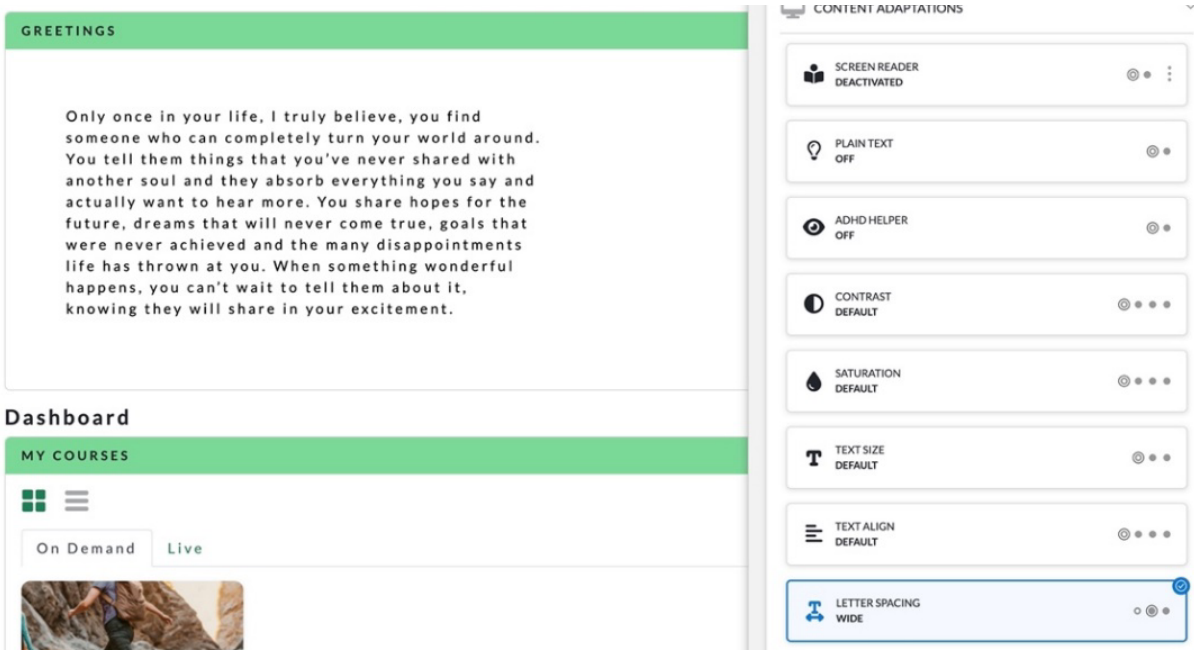



Fonte: https://test-vrdg.yeslms.dev

### 4.2.8 Espaçamento entre Letras

Os usuários podem definir o espaçamento entre as letras para largo ou extralargo. A Figura 24 apresenta espaçamento largo entre as letras.

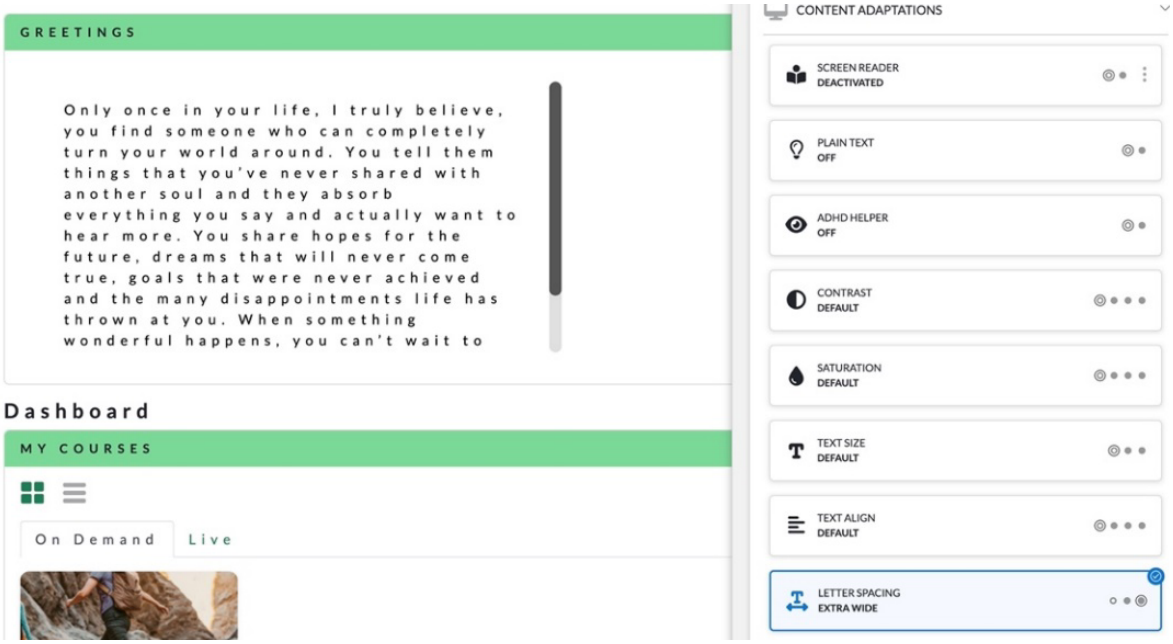
**Figura 24:** Espaçamento Largo – Widget YesLMS




Fonte: https://test-vrdg.yeslms.dev

A Figura 25 apresenta espaçamento extralargo entre as letras.

**Figura 25:** Espaçamento Extralargo – Widget YesLMS

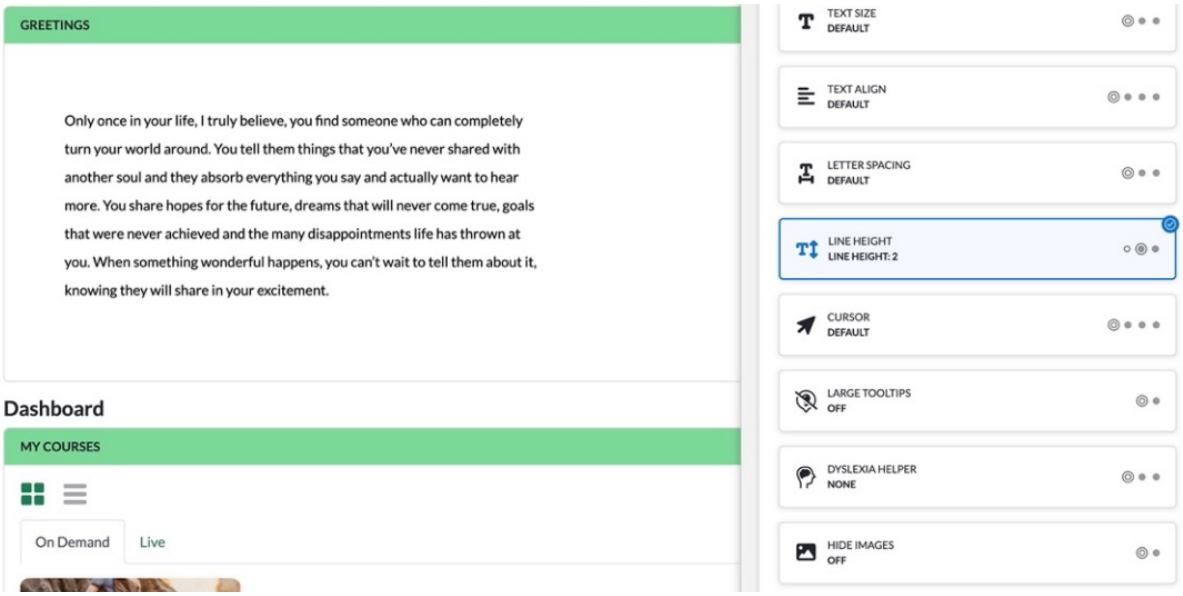



Fonte: https://test-vrdg.yeslms.dev

### 4.2.9 Altura da Linha

Os usuários podem ajustar o espaçamento entre as linhas para 2 ou 2,5. A Figura 26 apresenta espaçamento de 2 pontos entre as linhas.

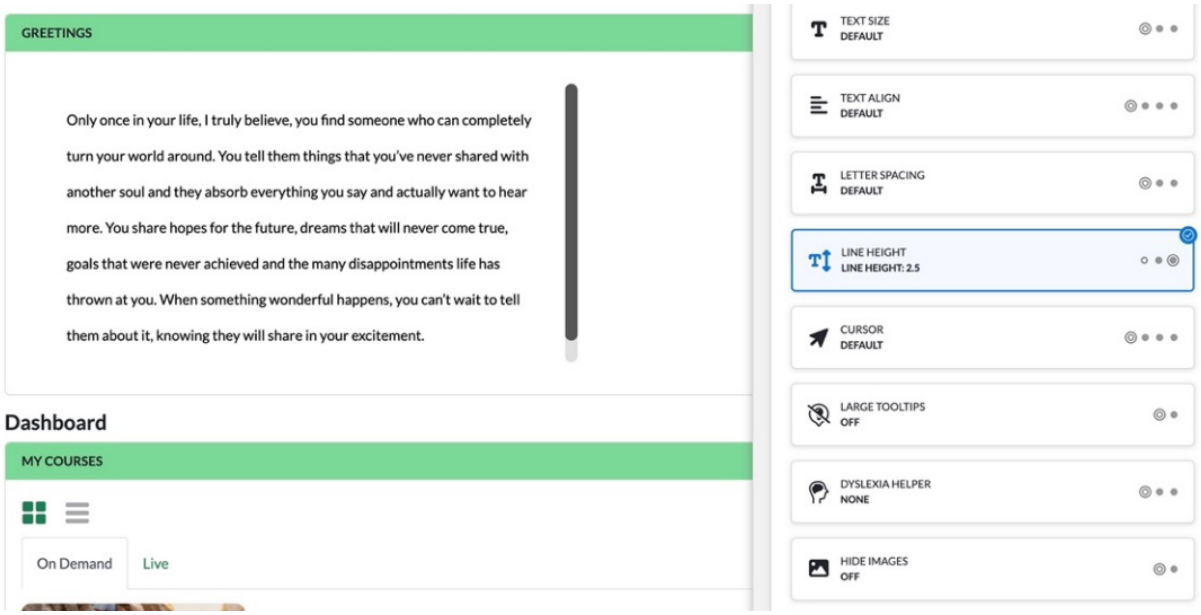
**Figura 26:** Altura de 2 Pontos – Widget YesLMS




Fonte: https://test-vrdg.yeslms.dev

A Figura 27 apresenta espaçamento de 2,5 entre as linhas.

**Figura 27:** Altura de 2,5 Pontos – Widget YesLMS

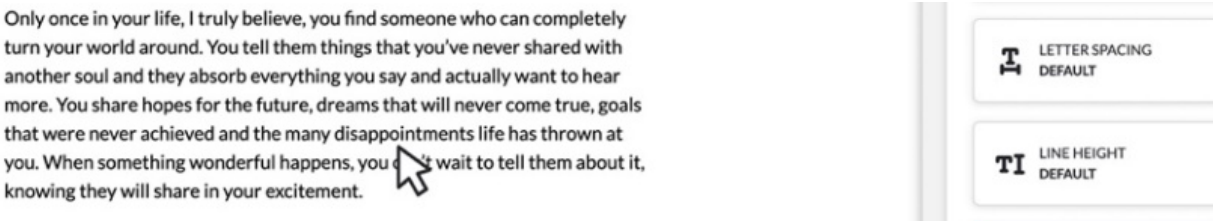



Fonte: https://test-vrdg.yeslms.dev

### 4.2.10 Cursor

Os usuários podem escolher entre dois modelos de cursores grandes e entre as cores branca e preta. A Figura 28 apresenta o primeiro modelo de cursor grande selecionado na cor branca.

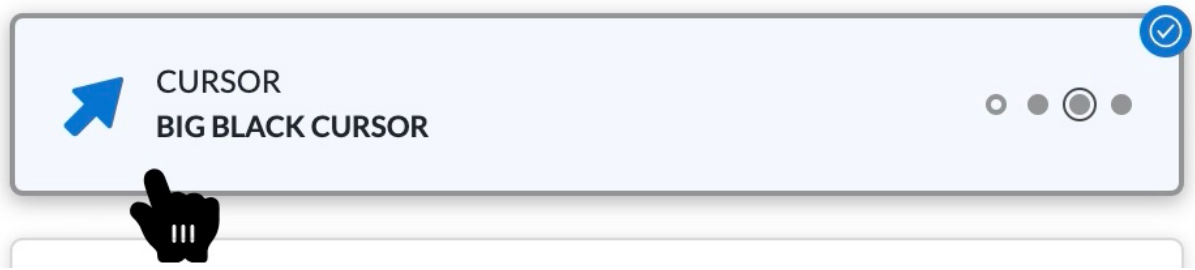
**Figura 28:** Modelo 1 de Cursor Grande na Cor Branca – Widget YesLMS




Fonte: https://test-vrdg.yeslms.dev

A Figura 29 apresenta o segundo modelo de cursor grande na cor preta.

**Figura 29:** Modelo 2 de Cursor Grande na cor Preta – Widget YesLMS



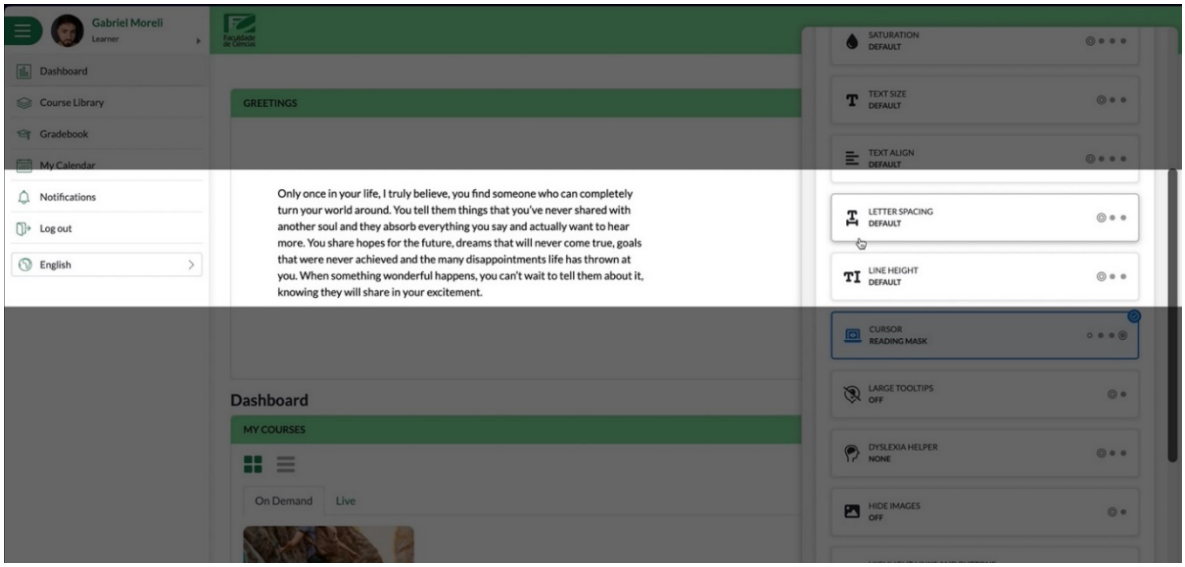

Fonte: https://test-vrdg.yeslms.dev

### 4.2.11 Máscara de Leitura

Além de cursores maiores, o *widget* também oferece a opção de máscara de leitura. A Máscara de Leitura é um recurso especial projetado para ajudar os usuários a se concentrarem em seções específicas da tela. Quando ativada, ela escurece toda a tela, exceto por uma área retangular ao redor do cursor. Essa "máscara" se move com o cursor, permitindo que o usuário se concentre na parte da tela com a qual está interagindo no momento. Este recurso pode ser particularmente útil para usuários com TDAH ou outras condições relacionadas à atenção, pois minimiza distrações e ajuda a guiar o foco.

A Figura 30 apresenta a máscara de leitura em uso na página.

**Figura 30:** Máscara de Leitura – Widget YesLMS

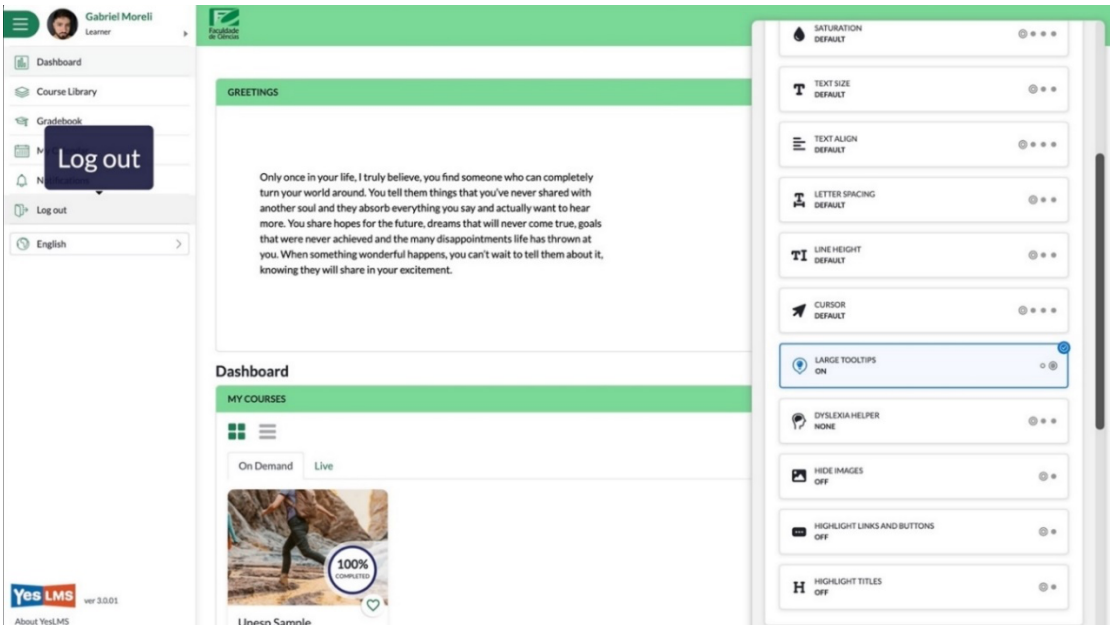



Fonte: https://test-vrdg.yeslms.dev

### 4.2.12 Aumento de *Tooltips*

Os usuários também podem ampliar os *tooltips* da página. A Figura 31 demonstra como os *tooltips* ficam ao serem aumentados.

**Figura 31:** *Tooltips* Maiores – Widget YesLMS

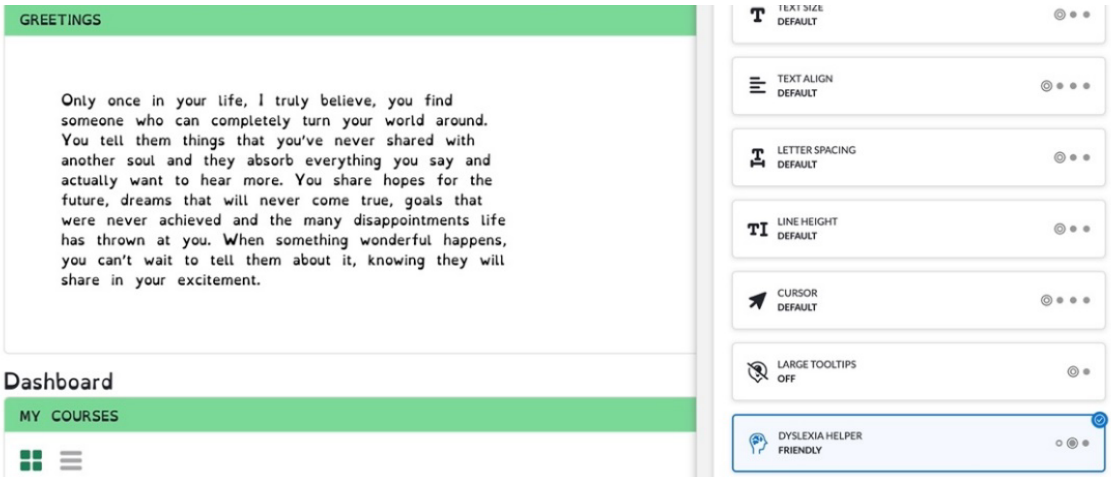



Fonte: https://test-vrdg.yeslms.dev

### 4.2.13 Auxiliar de Dislexia

Os usuários também podem escolher uma fonte legível ou amigável para dislexia. A Figura 32 apresenta um texto com fonte amigável para dislexia.

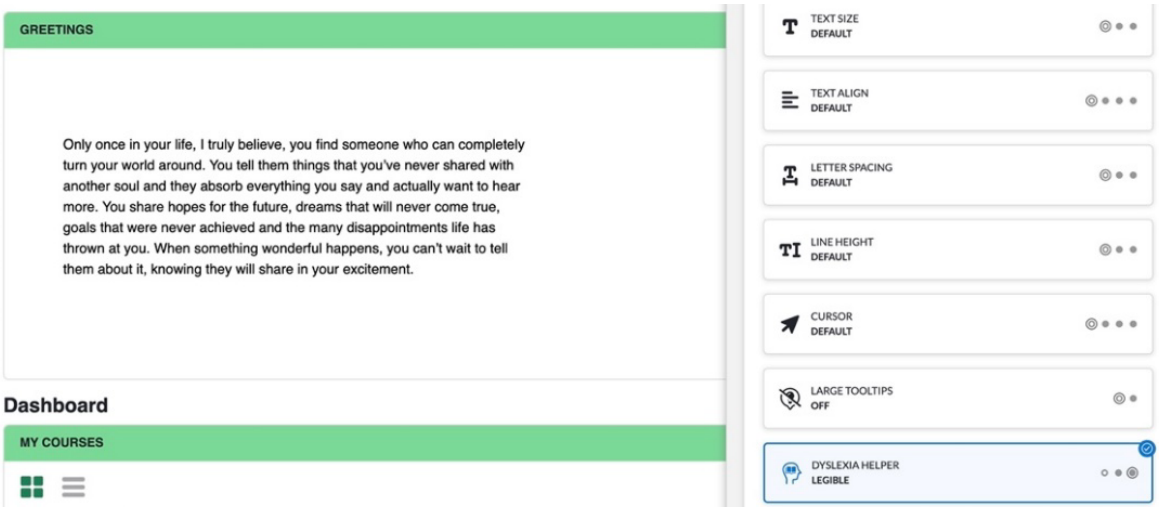
**Figura 32:** Fonte Amigável para Dislexia – Widget YesLMS




Fonte: https://test-vrdg.yeslms.dev

A Figura 33 apresenta o mesmo texto em fonte legível para dislexia.

**Figura 33:** Fonte Legível para Dislexia – Widget YesLMS

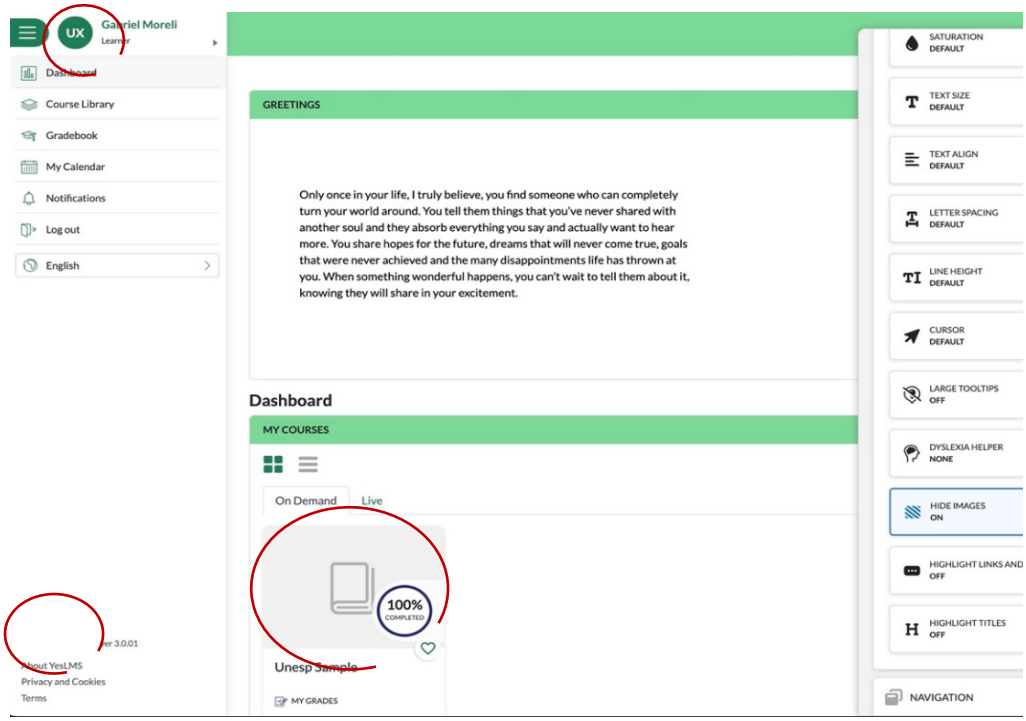



Fonte: https://test-vrdg.yeslms.dev

### 4.2.14 Ocultação de Imagens

Os usuários podem optar por ocultar ou exibir as imagens de uma página. A Figura 34 mostra uma página com as imagens ocultas.

**Figura 34:** Imagens Ocultas – Widget YesLMS

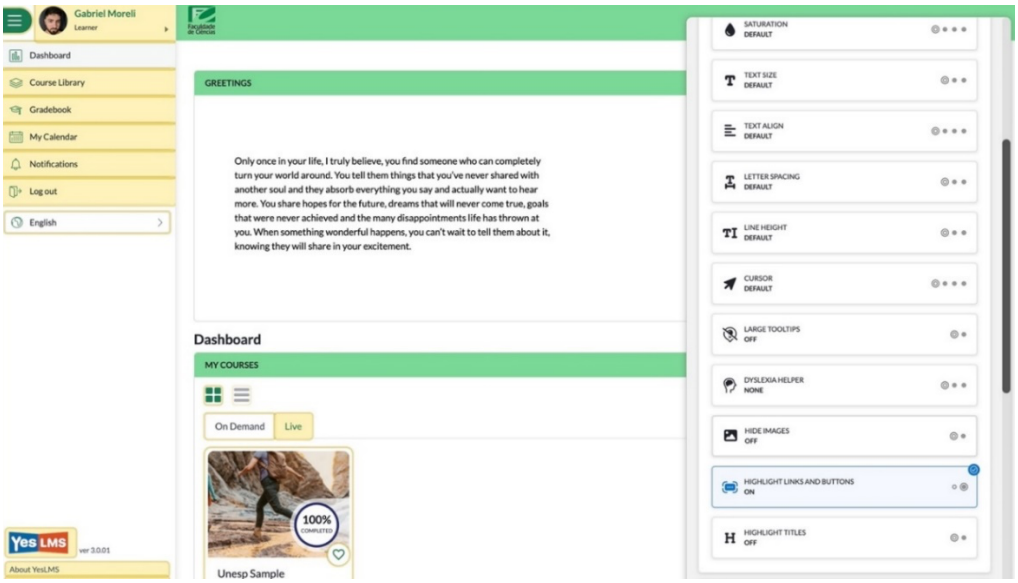



Fonte: https://test-vrdg.yeslms.dev, imagem editada.

### 4.2.15 Links e Botões em Destaque

Os usuários podem optar por destacar ou não destacar links e botões. A Figura 35 mostra a página com links e botões em destaque.

**Figura 35:** Links e Botões em Destaque – Widget YesLMS

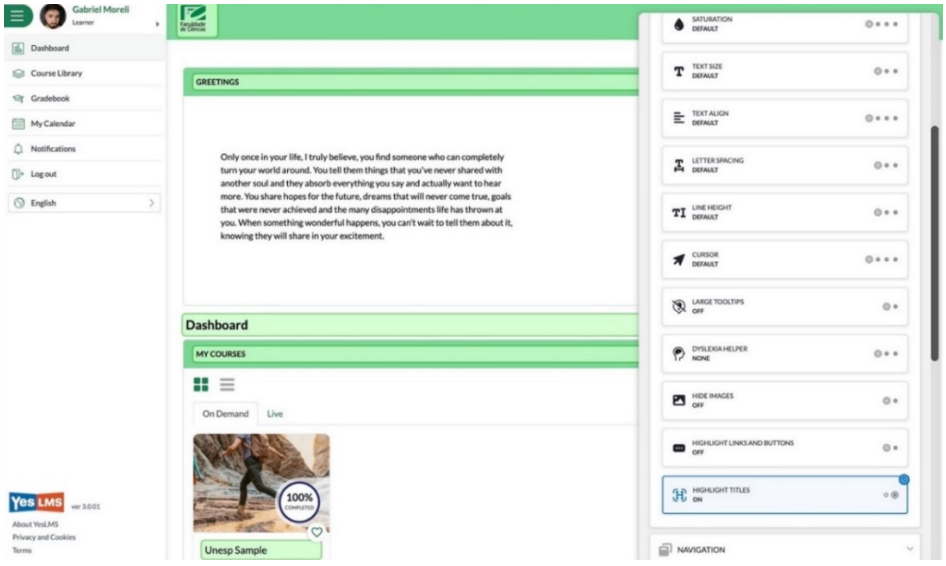



Fonte: https://test-vrdg.yeslms.dev

### 4.2.16 Títulos em Destaque

Os usuários também podem optar por destacar ou não destacar títulos. A Figura 36 mostra a página com títulos em destaque.

**Figura 36:** Títulos em Destaque – Widget YesLMS



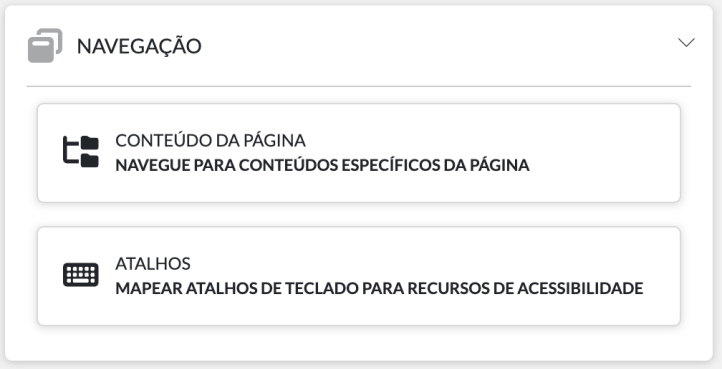

Fonte: https://test-vrdg.yeslms.dev

## 4.3 NAVEGAÇÃO

A seção de navegação permite aos usuários navegar para conteúdos específicos da página, separados entre cabeçalhos, marcos e links, além de poder verificar os atalhos de teclado disponíveis ao *widget*.

A Figura 37 é uma captura da tela de Navegação.

**Figura 37:** Navegação – Widget YesLMS




Fonte: https://test-vrdg.yeslms.dev

## 4.4 PERFIS DE ACESSIBILIDADE

A seção de Perfis de Acessibilidade permite aos usuários alternar entre perfis de acessibilidade pré-definidos. Cada perfil vem com um conjunto único de adaptações de conteúdo para atender a diferentes necessidades. Os perfis disponíveis são:

- Cego: Este perfil ativa o leitor de tela na velocidade padrão.
- Daltônico: Este perfil configura a saturação do conteúdo para alta.
- Deficiente Visual: Este perfil inclui alta saturação, texto grande, cursor grande, dicas de ferramentas grandes e um auxiliar de dislexia legível.
- Deficiência Cognitiva: Este perfil aumenta o tamanho do texto.
- Convulsão e Epilepsia: Este perfil configura a saturação do conteúdo para baixa.
- TDAH: Este perfil ativa o cursor de máscara de leitura.

Os usuários também podem criar perfis personalizados clicando em "Adicionar perfil de acessibilidade". Uma janela será aberta, apresentando todas as adaptações de conteúdo disponíveis. Os usuários podem selecionar quantas quiserem e atribuir um nome e ícone personalizados ao seu perfil.

A Figura 38 é uma captura da tela dos Perfis de Acessibilidade.

**Figura 38:** Perfis de Acessibilidade – Widget YesLMS

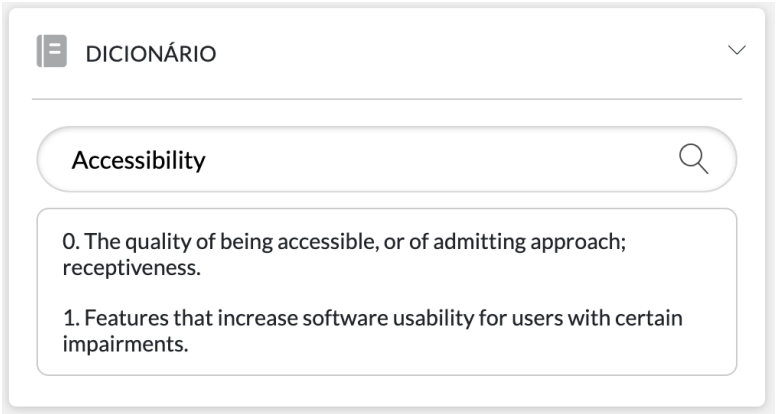

Fonte: https://test-vrdg.yeslms.dev

## 4.5 DICIONÁRIO

A quarta seção do *widget* é um dicionário inglês integrado. Esse recurso permite que os usuários procurem definições de palavras com as quais podem não estar familiarizados, melhorando seu entendimento do conteúdo.

A Figura 39 é uma captura da tela do Dicionário.

**Figura 39:** Dicionário – YesLMS Widget




Fonte: https://test-vrdg.yeslms.dev

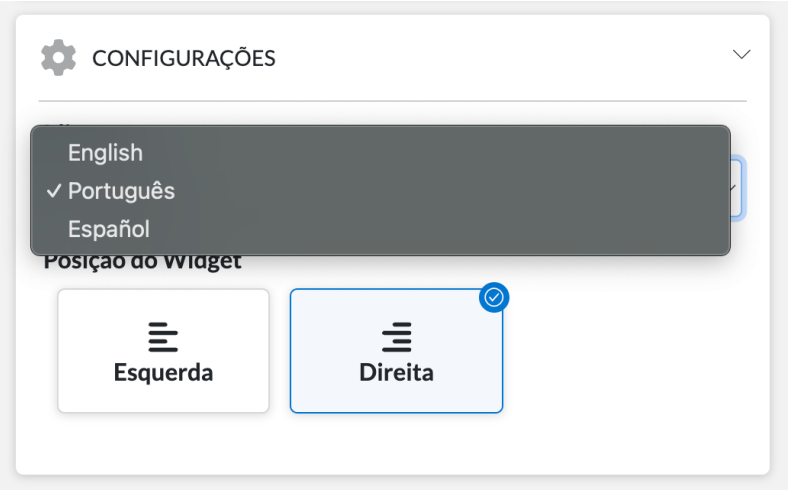
## 4.6 CONFIGURAÇÕES

A quinta seção é a área de Configurações. Aqui, os usuários podem personalizar sua experiência ajustando duas configurações principais:

- Idiomas: Os usuários podem selecionar entre três idiomas - Inglês Americano, Português Brasileiro e Espanhol. Uma vez escolhido um idioma, ele será aplicado em todo o sistema, não apenas no *widget*.
- Posição do *Widget*: Por padrão, o *widget* é posicionado no lado direito da tela. No entanto, os usuários têm a opção de movê-lo para o lado esquerdo, se preferirem.

A Figura 40 é uma captura da tela de Configurações.

**Figura 40:** Configurações – Widget YesLMS




Fonte: https://test-vrdg.yeslms.dev

## 4.7 ATALHOS

O recurso Atalhos oferece aos usuários um conjunto de combinações de teclas para navegar e interagir com o sistema de maneira mais eficiente. Esse recurso pode ser acessado pressionando SHIFT + TAB até que uma janela *pop-up* apareça com várias opções. A última opção é o mapeamento dos atalhos.

Ao selecionar essa opção, outra janela *pop-up* se abrirá, exibindo os atalhos pré-definidos do sistema. Os usuários têm a capacidade de ativar ou desativar esses atalhos, com exceção daqueles relacionados ao Leitor de Tela. Esses atalhos específicos estão disponíveis apenas quando a opção Leitor de Tela está ativada.

Os usuários também têm a flexibilidade de modificar os atalhos existentes e criar para perfis personalizados. No entanto, é importante observar que alguns atalhos escolhidos podem entrar em conflito com os atalhos existentes do sistema/navegador. Se um atalho falhar ao ser acionado, recomenda-se alterá-lo para uma combinação diferente. Também é aconselhável limitar as combinações de atalhos a um máximo de três teclas para garantir a leitura precisa das teclas pressionadas.

Aqui está uma visão geral rápida dos atalhos pré-definidos disponíveis:

- Leitor de Tela
- Navegar para o elemento nativo anterior: Shift + Tab
- Navegar para o próximo elemento nativo: Tab
- Navegar para o elemento anterior: Shift + Alt + Seta para a Esquerda
- Navegar para o próximo elemento: Shift + Alt + Seta para a Direita
- Navegar para o próximo Título: H
- Navegar para o Título anterior: Shift + H
- Navegar para o próximo Parágrafo: P
- Navegar para o Parágrafo anterior: Shift + P
- Navegar para a próxima Lista: L
- Navegar para a Lista anterior: Shift + L
- Navegar para a próxima Tabela: T
- Navegar para a Tabela anterior: Shift + T
- Ativar botão ou âncora: Enter

Além destes, também existem atalhos disponíveis para todos os perfis e adaptações de conteúdo. Os usuários simplesmente precisam ativá-los e clicar em salvar antes de usá-los. Como mencionado anteriormente, os usuários podem alterar a combinação e criar combinações para perfis personalizados.

Para fazer isso, basta clicar no campo de texto que exibe a combinação e pressionar os botões escolhidos juntos. Se a combinação não existir, o sistema imprimirá as teclas pressionadas. Se a combinação já existir, a última tecla pressionada não será mostrada, indicando que o usuário precisa escolher outra tecla para combinar com a última.

Para criar um atalho para um perfil personalizado, basta abrir o *widget* de acessibilidade. A opção "Adicionar atalho" será mostrada na parte inferior do perfil

personalizado. Ao clicar na nela, a janela de atalho se abrirá. O usuário então só precisa pressionar a combinação escolhida e clicar em salvar.

# 5 DESENVOLVIMENTO DO WIDGET

Por ser uma aplicação voltada a uma empresa privada, o código não pode ser compartilhado, mas os conceitos podem ser explicados. Abaixo, há um resumo de como cada adaptação de conteúdo foi desenvolvida.

## 5.1 LEITOR DE TELA

No desenvolvimento do Leitor de Tela, foram primeiramente identificados os elementos HTML relevantes. Utilizando ferramentas JavaScript para seleção de elementos de tela, eles foram categorizados em dois grupos: elementos de conjunto (como 'nav', 'header', 'ul', 'ol', 'table') e elementos únicos ('div', 'span', 'li', 'td', 'tr'). Para cada um desses elementos, um texto descritivo específico foi gerado. Através da funcionalidade 'speechSynthesis' do Angular, que aproveita vozes fornecidas pelos navegadores, o texto descritivo é convertido em áudio.

Além disso, a ferramenta leva em consideração atributos especiais como 'aria-label' e 'aria-hidden', adaptando ou omitindo a fala quando necessário. Utilizando a biblioteca Renderer do Angular, um retângulo azul é exibido ao redor do elemento focado na tela, melhorando a identificação visual do elemento selecionado pelo usuário.

## 5.2 SIMPLIFICAÇÃO DE TEXTO

Para a funcionalidade de Simplificação de Texto, foi adotado um método semelhante ao do Leitor de Tela para capturar elementos de texto na tela, porém com um foco especial em textos extensos. Expressões regulares foram utilizadas para filtrar esses textos. Esses elementos são então substituídos por um componente Angular específico, que apresenta o estilo de um retângulo azul contendo o texto original.

Quando o usuário clica nesse componente, uma requisição HTTP é enviada para uma API .NET, que, por sua vez, envia o texto para a API da OpenAI para simplificação. O texto simplificado, ao retornar, substitui o original no componente. Além disso, ao passar o mouse sobre o componente, um *pop-up* é exibido, permitindo ao usuário alternar entre o texto original e o simplificado.

## 5.3 ASSISTENTE DE TDAH

Para a funcionalidade do Assistente de TDAH, os elementos de texto são novamente capturados, sem a limitação de quantidade de caracteres. Estes elementos são substituídos por um componente Angular específico. Este componente tem como objetivo enfatizar partes de cada palavra, enquanto o restante é apresentado com um traço mais fino.

Essa abordagem visa destacar as partes mais importantes de cada palavra, facilitando a leitura para usuários com TDAH.

## 5.4 ALTO CONTRASTE

Esta funcionalidade foi projetada para melhorar a acessibilidade visual de páginas da *web* para usuários com dificuldades visuais. O código começa definindo variáveis para cor de fundo (*bgColor*), cor do texto (*textColor*) e estilo de borda (*border*). Com base no valor de uma variável de controle (*variable*), diferentes combinações de cores de fundo e texto são aplicadas. Por exemplo, pode-se configurar um fundo preto com texto branco ou amarelo, ou um fundo azul escuro com texto amarelo, dependendo da opção escolhida.

O próximo passo é a criação de uma folha de estilo CSS que utiliza essas cores definidas para aplicar um esquema de alto contraste em vários elementos HTML da página, como corpo, cabeçalho, botões e parágrafos. Se uma folha de estilo para ajustes de contraste não existir, o script cria uma nova e a adiciona ao cabeçalho do documento.

## 5.5 SATURAÇÃO

Esta funcionalidade ajusta a saturação de cores em páginas *web* para atender às preferências ou necessidades visuais dos usuários. Utilizando a biblioteca Renderer, o código configura o estilo global do documento, estabelecendo inicialmente um filtro de saturação padrão. A intensidade da saturação é modificada de acordo com o valor de uma variável chamada *variable*. As opções incluem:

- Normal: Restaura as cores ao seu estado original, removendo o filtro de saturação.
- *Low*: Reduz a saturação para 50%, diminuindo a intensidade das cores.
- *High*: Aumenta a saturação para 500%, tornando as cores mais intensas.
- *Desaturate*: Remove completamente a saturação, convertendo as cores para tons de cinza.

## 5.6 TAMANHO DO TEXTO

Esta funcionalidade permite aos usuários aumentar o tamanho da fonte em páginas *web*, melhorando a acessibilidade. Dependendo da configuração escolhida, *'large'* ou *'extra-large'*, a ferramenta ajusta o tamanho da fonte para um percentual levemente maior do que o padrão.

O código cria uma folha de estilo que aplica esse aumento de tamanho da fonte a todos os elementos de texto da página. Essa folha de estilo é então adicionada ao cabeçalho do documento, resultando em um aumento geral do tamanho da fonte para facilitar a leitura para usuários com dificuldades visuais ou preferência por textos maiores.

## 5.7 ALINHAMENTO DE TEXTO

Esta funcionalidade oferece aos usuários a opção de alterar o alinhamento do texto em páginas *web* à direita, à esquerda, ao centro ou justificado. O alinhamento é definido com base no valor de uma variável, que pode ser *'right'*, *'center'*, *'justify'* ou *'left'*.

O código cria uma folha de estilo e nela aplica a regra de alinhamento de texto para todos os elementos de texto da página. Essa folha de estilo é, então, adicionada ao cabeçalho do documento, o que resulta na aplicação do alinhamento de texto escolhido a toda a página.

## 5.8 ESPAÇAMENTO ENTRE LETRAS

Esta funcionalidade permite aos usuários ajustar o espaçamento entre as letras em páginas web para melhorar a legibilidade. O ajuste é realizado com base no valor de uma variável, que pode ser *'normal'*, *'wide'* (amplo) ou *'extra-wide'* (extra amplo).

Quando a variável é *'normal'*, o código remove qualquer estilo de espaçamento entre letras previamente aplicado, retornando o texto ao seu estado padrão.

Se a variável for *'wide'*, o espaçamento entre as letras é aumentado para 0.2em, proporcionando um espaçamento mais amplo entre as letras.

Se a variável for *'extra-wide'*, o espaçamento é ainda mais ampliado, para 0.4em, oferecendo um espaçamento substancialmente maior entre as letras.

O código utiliza a biblioteca Renderer para aplicar o estilo de espaçamento entre letras ao corpo do documento.

## 5.9 ALTURA DA LINHA

Esta funcionalidade oferece a opção de ajustar o espaçamento entre as linhas de texto em páginas web. O ajuste depende do valor de uma variável, que pode ser *'normal'*, *'medium'* ou *'large'*.

Quando a variável é 'normal', o espaçamento entre as linhas é ajustado para 1.5, o que é considerado um espaçamento padrão para uma boa legibilidade.

Se a variável for *'medium'*, o espaçamento entre as linhas é aumentado para 2, proporcionando um espaçamento mais confortável para a leitura.

Se a variável for *'large'*, o espaçamento é ajustado para 2.5, oferecendo um espaçamento ainda maior entre as linhas.

Esses ajustes são feitos utilizando a biblioteca Renderer, que aplica o estilo de espaçamento de linha desejado ao corpo do documento.

## 5.10 CURSOR

Esta funcionalidade é projetada para permitir aos usuários personalizar o cursor na página web, oferecendo opções como um cursor maior ou uma máscara de leitura. A ação principal do código é verificar a escolha do usuário e aplicar as mudanças correspondentes no cursor.

Para um cursor maior, o código substitui o cursor padrão por uma versão ampliada, disponível em branco ou preto, para melhorar a visibilidade. Isso é feito adicionando uma folha de estilo ao documento que altera o cursor para links e botões.

A opção de 'máscara de leitura' adiciona uma sobreposição escura à página que se move com o cursor, destacando a linha de texto sobre a qual o usuário está passando, facilitando a leitura.

Além disso, o código inclui funções para resetar o cursor para o padrão e remover estilos de cursor personalizados quando necessário. Essas funções garantem que as mudanças possam ser revertidas ou ajustadas conforme a escolha do usuário.

## 5.11 AUMENTO DE *TOOLTIPS*

Esta funcionalidade permite a criação de *tooltips* personalizadas e ampliadas em páginas *web*, que são ativadas quando o usuário passa o mouse sobre elementos específicos. A funcionalidade é controlada por uma variável que, quando ativada, inicia a observação de *tooltips* na página.

Ao passar o mouse sobre um elemento com um atributo de *'title'*, o código substitui o *tooltip* padrão por um personalizado. Esse *tooltip* personalizado é maior, com estilos específicos para maior visibilidade e legibilidade, incluindo cor de fundo escura, texto claro, bordas definidas e tamanho de fonte ampliado.

A ferramenta usa um observador (MutationObserver) para monitorar mudanças no DOM e aplicar os estilos personalizados a qualquer novo *tooltip* que apareça na página.

## 5.12 AUXILIAR DE DISLEXIA

Esta funcionalidade foi desenvolvida para melhorar a legibilidade do texto em páginas *web* para usuários com dislexia. Ela permite que os usuários escolham entre diferentes fontes que são consideradas mais fáceis de ler para pessoas com dislexia.

Quando a variável é definida como 'normal', a ferramenta remove qualquer estilo de fonte previamente aplicado ao corpo do documento, retornando a fonte ao seu estilo padrão.

Se a variável for *'friendly'*, a ferramenta aplica a fonte 'OpenDyslexic3', que é projetada para aumentar a legibilidade para usuários com dislexia, ao corpo do documento.

Se a variável for *'lexie'*, a ferramenta aplica a fonte 'Lexie Readable', outra fonte conhecida por ser amigável para leitores com dislexia.

## 5.13 OCULTAÇÃO DE IMAGENS

Esta funcionalidade permite aos usuários ocultar todas as imagens em uma página *web*. É controlada por uma variável que, quando ativada, aciona a ação de ocultar as imagens.

Quando a variável está definida como *'on'*, o código percorre todas as imagens na página e altera o estilo de visibilidade para *'hidden'*, efetivamente ocultando-as da visualização.

Se a variável não estiver definida como *'on'*, o código reverte a visibilidade das imagens para *'visible'*, fazendo com que elas reapareçam.

## 5.14 LINKS, BOTÕES E TÍTULOS EM DESTAQUE

Esta funcionalidade é projetada para destacar títulos, links e botões em uma página *web*, melhorando a visibilidade e a acessibilidade. A ativação ou desativação do destaque é controlada por um evento que, quando ativado, aplica um estilo especial aos elementos selecionados.

Para títulos (h1, h2, h3, h4, h5, h6), quando ativado, o código aplica um fundo claro, uma borda colorida, um pouco de arredondamento nos cantos, um ‘padding’ (enchimento) e altera a cor do texto. Esse destaque torna os títulos mais visíveis e fáceis de distinguir.

Para links e botões, o código aplica um estilo similar, mas com cores diferentes, destacando esses elementos de forma eficaz.

Quando a funcionalidade é desativada, o estilo aplicado é removido, retornando os elementos ao seu estado original.

# 6 FLUXO DE DISTRIBUIÇÃO DO WIDGET

## 6.1 *BUILD* VIA GITHUB ACTIONS

O processo de distribuição do *widget* inicia-se com cada *commit* no repositório GitHub, acionando o GitHub Actions para realizar um novo *build*. Esse *build* é executado utilizando o padrão Nx do Angular, uma extensão do Angular CLI que facilita o desenvolvimento e a construção de aplicações Angular em larga escala. Durante esse processo, o Nx realiza várias tarefas cruciais, incluindo a compilação de TypeScript para JavaScript, a otimização de *assets* e a geração de artefatos de produção, assegurando que o *widget* seja construído eficientemente e de acordo com as melhores práticas.

Após a conclusão do build pelo Nx, entra em cena um comando Webpack adicional. O Webpack, uma ferramenta de empacotamento de módulos, é utilizado para consolidar todos os arquivos e dependências gerados no *build* em um único arquivo JavaScript. Esse arquivo único, gerado como artefato final do processo de *build*, simplifica a distribuição e implantação do *widget*, facilitando seu carregamento e execução no lado do cliente.

Esse fluxo automatizado via GitHub Actions garante que cada *commit* resulte em um *widget* atualizado, otimizado e pronto para ser distribuído. Com isso, o *widget* mantém-se sempre atualizado com as últimas modificações e melhorias, contribuindo para uma entrega contínua e segura.

## 6.2 HOSPEDAGEM E CONFIGURAÇÕES NO AZURE

Após o processo de *build* e empacotamento via GitHub Actions, o artefato JavaScript final é enviado para o Azure Storage, um serviço de armazenamento de dados na nuvem oferecido pelo Microsoft Azure. Esse envio envolve a criação de dois arquivos principais no Azure Storage:

- Arquivo Nomeado pela Versão (SemVer): O primeiro arquivo é nomeado de acordo com a versão do *widget*, seguindo as convenções do Semantic Versioning (SemVer). O SemVer é um sistema de versionamento amplamente adotado que atribui a cada versão um identificador único, baseado em três

números principais: versão maior (*major*), menor (*minor*) e *patch*. Essa prática permite que os usuários e desenvolvedores identifiquem facilmente qual versão do *widget* está sendo utilizada, facilitando o gerenciamento de atualizações e dependências.

- Arquivo ‘widget-latest.js’: Além do arquivo versionado, é criado um segundo arquivo denominado 'widget-latest.js'. Esse arquivo serve como uma referência constante para a versão mais recente do widget. Independentemente de quantas atualizações sejam feitas, os usuários que apontam para 'widget-latest.js' sempre terão acesso à versão mais atual do *widget*, sem a necessidade de modificar o link para cada nova versão.

A combinação desses dois métodos de armazenamento no Azure Storage oferece flexibilidade e eficiência na gestão e distribuição do *widget*. Enquanto o arquivo versionado facilita o controle de versões específicas, o arquivo 'widget-latest.js' assegura que a versão mais recente e atualizada esteja sempre disponível para uso imediato. Adicionalmente, o Azure Storage garante segurança e escalabilidade, fornecendo uma plataforma confiável para o armazenamento dos arquivos do widget.

## 6.3 CDN PARA DISTRIBUIÇÃO RÁPIDA

Para aprimorar ainda mais a distribuição do *widget*, foi integrado um *Content Delivery Network* (CDN) ao Azure Storage. Essa adição traz melhorias significativas na velocidade e eficiência da entrega do *widget* aos usuários finais. O CDN, configurado especificamente para esse contexto, apresenta duas configurações-chave:

- Sem *Cache* para 'widget-latest.js': O arquivo 'widget-latest.js', que representa a versão mais recente do *widget*, é configurado no CDN para não manter cache. Isso significa que toda vez que uma requisição é feita para esse arquivo, a versão mais atualizada é fornecida diretamente do Azure Storage, garantindo que os usuários sempre tenham acesso à última versão do *widget* sem atrasos causados pelo cache.

- Compressão de Arquivos JS: Outra configuração importante do CDN é a compressão automática de arquivos JavaScript, incluindo o *widget*. Quando um arquivo JS é requisitado pelo navegador, ele é baixado na forma comprimida (gzip), o que reduz significativamente o tamanho do arquivo e, consequentemente, o tempo de *download*. Essa compressão é especialmente valiosa para melhorar a performance em conexões de internet mais lentas ou instáveis.

A combinação dessas configurações no CDN resulta em uma distribuição mais eficiente e rápida do *widget*. Os usuários se beneficiam de tempos de carregamento reduzidos e da garantia de sempre acessarem a versão mais atual do *widget*, enquanto a compressão dos arquivos ajuda a minimizar a largura de banda necessária e aperfeiçoa a experiência geral do usuário.

# 7 IMPLEMENTAÇÃO DO WIDGET

## 7.1 IMPLEMENTAÇÃO DO WIDGET EM UMA SOLUÇÃO JS

Para a implementação do *widget* em uma solução JavaScript, a primeira etapa é fazer referência ao arquivo provido pelo CDN. Esse arquivo já está otimizado e comprimido para uma entrega rápida e eficiente. Em vez de baixar e importar o *bundle* para a pasta de ativos (*assets*) do projeto, o arquivo é referenciado diretamente no arquivo principal do projeto, como o 'index.html'. Isso é feito adicionando o link do script que aponta para o CDN. Assim que o link é adicionado, o navegador dos usuários finais irá buscar o arquivo diretamente do CDN quando a página for carregada. Por fim, o elemento HTML correspondente à biblioteca é adicionado ao documento, o que permite a integração efetiva e a utilização do widget na solução.

A Figura 41 mostra um exemplo de implementação da biblioteca em uma solução Vanilla JS.

**Figura 41:** Implementação do Widget em Solução JS – Widget YesLMS

```
<yeslms-widget></yeslms-widget>
<script src="https://yeslms-libs.azureedge.net/yeslms-libs/accessibility-widget/widget-latest.js"></script>
```

Fonte: Exemplo capturado em ambiente local utilizando editor VS Code.

## 7.2 APIS OFERECIDAS PELA BIBLIOTECA

A biblioteca oferece diversas APIs em escopo global que podem ser acessadas diretamente via *tag* <script> dentro de um arquivo HTML.

A interface de programação (API) possibilita a abertura ou fechamento do menu do *widget*, além de permitir a gestão de algumas funcionalidades e atributos do *widget* de acessibilidade. Essa funcionalidade pode ser útil caso haja a necessidade de automatizar a utilização de certas funções do *widget*.

Para empregar a API YesLMS, é necessário ter o *widget* implementado no site.

As Figuras 42, 43, 44 e 45 apresentam uma lista de métodos disponíveis para interação com a API, cada método possui um comentário acima explicando sua função.

**Figura 42:** Interação com a API #1 - Widget YesLMS

```
<script>
    /* Mostra o Widget */
    YesLMS.showWidget();

    /* Esconde o Widget */
    YesLMS.hideWidget();

    /* Alterna entre mostrar e esconder o Widget */
    YesLMS.toggleWidget();

    /* Reseta as configurações do Widget para o padrão */
    YesLMS.resetSettings();

    /* Altera a linguagem padrão do Widget.
       Linguages suportadas: pt-BR, es-ES, en-US */
    YesLMS.changeDefaultWidgetLanguage('pt-BR');

    /* Altera a linguagem atual do Widget.
       Linguages suportadas: pt-BR, es-ES, en-US */
    YesLMS.changeWidgetLanguage('pt-BR');

    /* Possibilita passar um método para ser executado quando o usuário clicar
        em um dos botões de acessibilidade.
       Esse método ira sobrescrever o método padrão do Widget */
       YesLMS.setCustomAccessibilityOptions(
        {
            'high-contrast': [
                {
                    '1': () => {
                        console.log('high-contrast 1');
                    }
                },
                {
                    '2': () => {
                        console.log('high-contrast 2');
                    }
                }
            ],
        }
    );
</script>
```

Fonte: Exemplo capturado em ambiente local utilizando editor VS Code.

**Figura 43:** Interação com a API #2 - Widget YesLMS

```
<script>
    /* Ativa o cursor de acessibilidade
       Opções suportadas: large-white-cursor, large-black-cursor, reading-mask */
    YesLMS.activateCursor('large-white-cursor');

    /* Desativa o cursor de acessibilidade */
    YesLMS.deactivateCursor();

    /* Ativa o modo de auxílio para TDAH */
    YesLMS.activateAdhdHelper();

    /* Desativa o modo de auxílio para TDAH */
    YesLMS.deactivateAdhdHelper();

    /* Ativa o auxílio para dislexia
       Opções suportadas: friendly, lexie */
    YesLMS.activateDyslexiaHelper('friendly');

    /* Desativa o auxílio para dislexia */
    YesLMS.deactivateDyslexiaHelper();

    /* Oculta as imagens */
    YesLMS.activateHideImages();

    /* Mostra as imagens */
    YesLMS.deactivateHideImages();

    /* Ativa o alto contraste
       Opções suportadas: 1 (Preto/Branco), 2 (Preto/Amarelo), 3 (Azul/Amarelo) */
    YesLMS.activateHighContrast('1');

    /* Desativa o alto contraste */
    YesLMS.deactivateHighContrast();

    /* Destaca os títulos */
    YesLMS.activateHighlightTitles();

    /* Remove o destaque dos títulos */
    YesLMS.deactivateHighlightTitles();

    /* Destaca os links */
    YesLMS.activateHighlightLinks();

    /* Remove o destaque dos links */
    YesLMS.deactivateHighlightLinks();
</script>
```

Fonte: Exemplo capturado em ambiente local utilizando o editor VS Code.

**Figura 44:** Interação com a API #3 - Widget YesLMS

```
<script>
    /* Altera o espaçamento entre as letras
       Opções suportadas: wide, extra-wide */
    YesLMS.activateLetterSpacing('wide');

    /* Remove o espaçamento entre as letras */
    YesLMS.deactivateLetterSpacing();

    /* Altera o espaçamento entre as linhas
       Opções suportadas: medium, large */
    YesLMS.activateLineHeight('medium');

    /* Remove o espaçamento entre as linhas */
    YesLMS.deactivateLineHeight();

    /* Simplifica textos grandes */
    YesLMS.activatePlainText();

    /* Remove a simplificação de textos grandes */
    YesLMS.deactivatePlainText();

    /* Altera a saturação das cores
       Opções suportadas: desaturate, low, high */
    YesLMS.activateSaturation('high');

    /* Remove a saturação das cores */
    YesLMS.deactivateSaturation();

    /* Ativa o leitor de tela */
    YesLMS.activateScreenReader();

    /* Desativa o leitor de tela */
    YesLMS.deactivateScreenReader();

    /* Altera o align do texto
       Opções suportadas: right, center, justify */
    YesLMS.activateTextAlign('center');

    /* Remove o align do texto */
    YesLMS.deactivateTextAlign();
</script>
```

Fonte: Exemplo capturado em ambiente local utilizando o editor VS Code.

**Figura 45:** Interação com a API #4 - Widget YesLMS

```
<script>
    /* Altera o tamanho da fonte
       Opções suportadas: large, extra-large */
    YesLMS.activateTextSize('large');

    /* Remove o tamanho da fonte */
    YesLMS.deactivateTextSize();

    /* Ativa tooltips maiores */
    YesLMS.activateTooltips();

    /* Desativa tooltips maiores */
    YesLMS.deactivateTooltips();
</script>
```

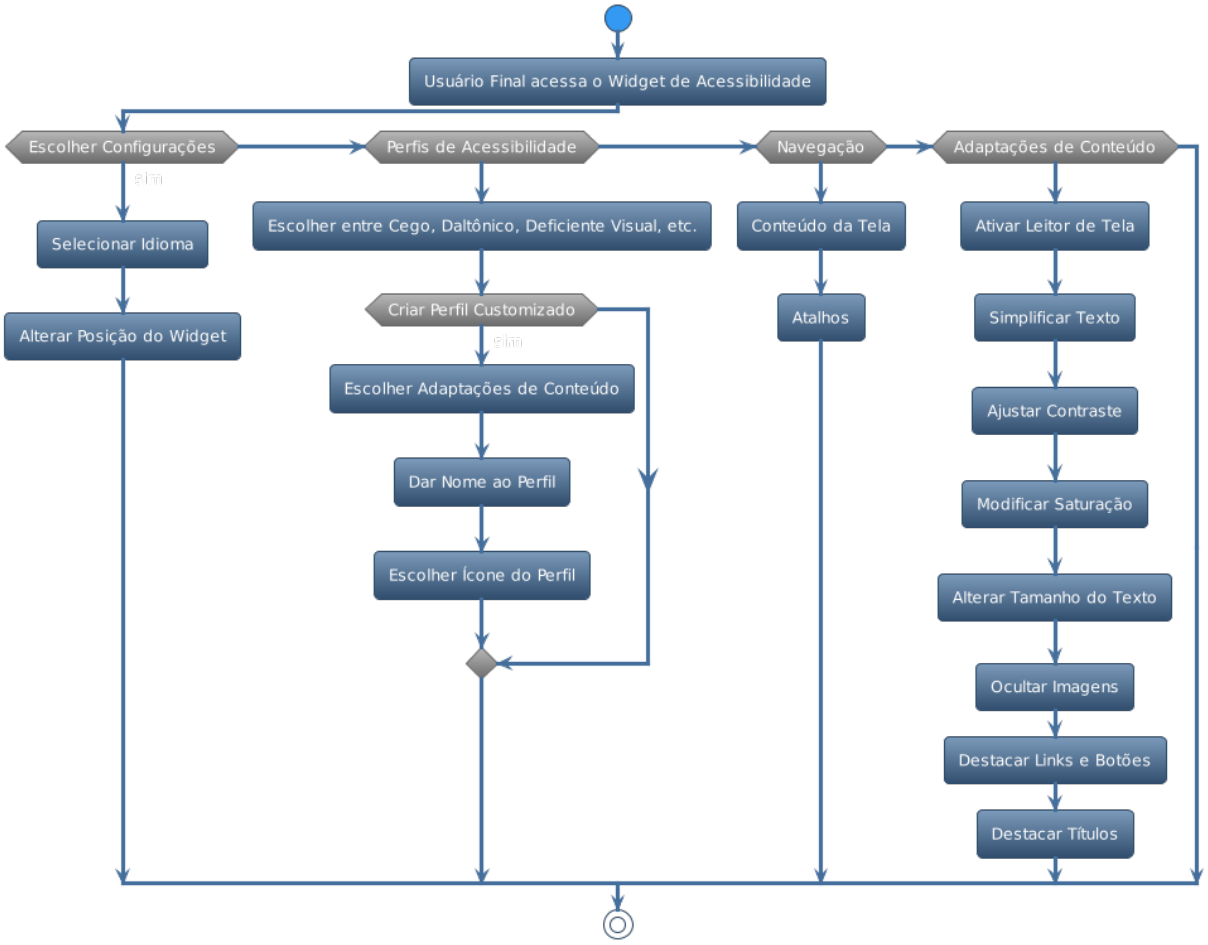
Fonte: Exemplo capturado em ambiente local utilizando o editor VS Code.

# 8 DIAGRAMA DE ATIVIDADES

Um diagrama de atividades é uma ferramenta visual usada para modelar o fluxo de um processo ou sistema, destacando as etapas, decisões e a sequência das ações.

A Figura 46 ilustra o fluxo de utilização do *widget* de acessibilidade por um usuário final.

**Figura 46:** Diagrama de Atividades




Fonte: Exemplo criado utilizando o software draw.io.

## 9 COMPARATIVO EM RELAÇÃO AOS CONCORRENTES

A Tabela 1 apresenta uma comparação direta entre as funcionalidades dos *widgets* de acessibilidade da UserWay e da AccessiBe, além de destacar as características do *widget* desenvolvido neste trabalho.

**Tabela 1:** Comparativo UserWay, AccessiBe e YesLMS

| Funcionalidade | UserWay | AccessiBe | YesLMS |
|---|---|---|---|
| Número de recursos | Mais de 20 recursos, incluindo contraste, fontes, cores, tamanhos de texto, navegação e acessibilidade de teclado | Mais de 15 recursos, incluindo contraste, fontes, cores, tamanhos de texto, navegação e acessibilidade de teclado | Mais de 20 recursos, incluindo ajustes de contraste, saturação, atalhos de teclado, leitor de tela e ferramentas exclusivas como a simplificação de texto e auxílio para pessoas com TDAH. |
| **Abordagem** | Processos Automatizados e Manuais | Tecnologia de IA Automatizada | Manipulação dinâmica do conteúdo da tela e ferramentas que utilizam tecnologia de IA. |
| **Base de Clientes** | Mais de 1 milhão de instalações em sites | Mais de 100.000 clientes, incluindo organizações legais e governamentais | Sendo testado em uma base com mais de 10.000 usuários. Foco será entrar no mercado através dos clientes da YesLMS que utilizam o LMS. |
| **Personalização** | Personalizável para combinar com o tema do site | Personalizável para combinar com o tema do site | Ainda não possuí personalização por cliente. Será implementado futuramente. |

| | | | |
|---|---|---|---|
| **Preço para Pequenos Sites** | A partir de $49/mês | A partir de $49/mês | Valores ainda serão definidos. |

Fonte: https://www.accessibilitychecker.org/blog/userway-vs-accessibe e funcionalidades do *widget* YesLMS.

Analisando o gráfico apresentado, nota-se que os concorrentes detêm uma ampla base de usuários e funcionalidades já consolidadas. O widget desenvolvido neste projeto é direcionado a um segmento de mercado no qual a empresa YesLMS possui um extenso número de clientes com deficiências. Isso proporciona uma excelente oportunidade para obter feedback valioso que será fundamental para o aperfeiçoamento contínuo do widget. Ademais, o widget oferece funcionalidades inovadoras, tais como a simplificação de textos e um assistente para usuários com TDAH, bem como a flexibilidade para configurar múltiplos atalhos de teclado, enriquecendo sua usabilidade. O objetivo é evoluir o widget com base no retorno dos clientes, introduzindo novas funcionalidades que sejam genuinamente úteis para pessoas com deficiência.

# 10 CONCLUSÃO

Este estudo teve como objetivo principal o desenvolvimento de um *widget* de acessibilidade visando a inclusão digital de indivíduos com deficiências e transtornos cognitivos na plataforma de cursos online da YesLMS. O projeto inovou ao integrar funcionalidades únicas, como a simplificação de textos e o suporte direcionado a usuários com TDAH, distinguindo-se de modelos anteriores.

A construção deste *widget* foi realizada utilizando o *framework* JavaScript Angular, em conjunto com a tecnologia Angular Elements. Além disso, houve contribuições significativas de outras tecnologias, como webpack e dotnet, enriquecendo o processo de desenvolvimento.

Embora os resultados iniciais sejam preliminares, eles indicam uma tendência positiva na facilitação do acesso e na melhoria da experiência de aprendizagem para esse público.

Durante a realização deste trabalho, tornou-se evidente a urgente necessidade de mais ferramentas como esta, bem como de outras soluções que tornem a internet um espaço mais acolhedor e inclusivo para pessoas com deficiências e transtornos cognitivos. Este projeto conseguiu uma contribuição modesta, mas significativa, para minimizar essa lacuna.

Olhando para o futuro, o widget de acessibilidade da YesLMS continuará em constante evolução, passando por revisões e aprimoramentos contínuos que focarão principalmente na expansão e refinamento de suas funcionalidades. É essencial manter um ciclo de melhoria contínua, especialmente para atender às novas necessidades que surgem constantemente, vindas de grupos de usuários com exigências específicas em termos de acessibilidade. Entre as diversas melhorias planejadas, a redução do tamanho do arquivo JavaScript final será também considerada, para assegurar uma performance superior e alinhada com a concorrência. Este trabalho, portanto, representa um passo importante, mas é apenas o começo de um caminho longo e necessário rumo a uma inclusão digital mais ampla e efetiva, onde a eficiência e a otimização do código caminham lado a lado com a acessibilidade.

## 11 REFERÊNCIAS

OMS, 2023. Disponível em: https://www.who.int/news-room/fact-sheets/detail/disability-and-health#:~:text=Key%20facts,1%20in%206%20of%20us
Último acesso em 9/12/2023.

IBGE, Pesquisa Nacional de Amostra de Domicílios Contínua (PNAD Contínua), 2022. Disponível em: https://www.ibge.gov.br/estatisticas/sociais/trabalho/17270-pnad-continua.html
Último acesso em 9/12/2023.

NUNES, Sérgio Sobral. A Acessibilidade na Internet no Contexto da Sociedade da Informação. Universidade do Porto, 2022. Disponível em: https://web.fe.up.pt/~mgi01016/is/acessibilidade.pdf
Último acesso em 9/12/2023.

The WebAIM Million, 2023. Disponível em: https://webaim.org/projects/million/
Último acesso em 9/12/2023.

Widget de Acessibilidade UserWay. Disponível em: https://userway.org/
Último acesso em 9/12/2023.

Widget de Acessibilidade AccessiBe. Disponível em: https://accessibe.com/
Último acesso em 9/12/2023.

Ambiente de Teste da YesLMS. Disponível em: https://test-vrdg.yeslms.dev/
Último acesso em 20/11/2023.

TRICHTER, Danny. UserWay vs AccessiBe: Comparing Two Top Accessibility Tools (2023). Disponível em: https://www.accessibilitychecker.org/blog/userway-vs-accessibe/
Último acesso em 9/12/2023.