Documentação de Projeto

para o sistema

Guia de Desenvolvimento de Interface com o usuário (UI Guide)

Versão 2.7

Projeto de sistema elaborado pelo(s) aluno(s) Ariel Santos Barcelos e Karolina Vaz Coelho e apresentado ao curso de **Engenharia de Software** da **PUC Minas** como parte do Trabalho de Conclusão de Curso (TCC), sob orientação de conteúdo do professor José Laerte Xavier, orientação acadêmica do professor Lesandro Ponciano e orientação de TCC II do professor Lesandro Ponciano.

27/11/2021

Tabela de Conteúdo

Histórico de Revisões

Nome	Data	Razões para Mudança	Versão
Criação	06/03/2021	Criação do documento	1.0
Modificações	14/03/2021	Adição de modelo de usuários	1.1
Modificações	21/03/2021	Adição de templates (modelos) de tela	1.2
Modificações	31/03/2021	Adição de DSS (Diagrama de sequência do sistema)e caso de uso	1.3
Modificações	03/04/2021	Adição de DS (Diagrama de sequência) e diagrama de comunicação	1.4
Modificações	18/04/2021	Adição de Diagrama de Componentes e Diagrama de Arquitetura	1.5

Modificações	24/04/2021	Adição de Diagrama de Implantação, MER e Glossário de Dados	1.6
Modificações	30/04/2021	Adição de Testes	1.7.1
Modificações	04/05/2021	Correções	1.7.2
Modificações	08/05/2021	Adição de cronograma e continuação correções	1.8
Modificações	21/05/2021	Correções gerais no documento	1.9
Modificações	23/05/2021	Revisão das correções gerais	2.0
Modificações	14/08/2021	Correção após avaliação subjetiva	2.1
Modificações	19/09/2021	Modificação da seção 7 para abordar o processo de desenvolvimento do TCC II	2.2
Modificações	26/09/2021	Inclusão das subseções 7.1, 7.2 e 7.3. Modificações na seção 7.	2.3
Modificações	07/10/2021	Modificação da Tabela 29, inserção na subseção 7.3	2.4
Modificações	24/10/2021	Correções gerais	2.5
Modificações	11/11/2021	Correções gerais	2.6
Modificações	27/11/2021	Correções após a banca de TCC II	2.7

1. Introdução

Este documento agrega: 1) a elaboração e revisão de modelos de domínio e 2) modelos de projeto para o sistema Guia de Interface do usuário (UI Guide, do inglês User Interface Guide). A referência principal para a descrição geral do problema, domínio e requisitos do sistema é o documento de especificação que descreve a visão de domínio do sistema. Tal especificação acompanha este documento. Anexo neste documento também se encontra o Glossário.

2. Modelos de Usuário e Requisitos

Nesta seção são apresentados os modelos de usuários e requisitos para o sistema. Tal apresentação é feita pela descrição dos atores (Seção 2.1), descrição dos modelos de usuários (Seção 2.2), descrição de casos de uso de histórias dos usuários (Seção 2.3) e descrição do diagrama de sequências e contrato de operações para o sistema (seção 2.4).

2.1 Descrição de Atores

Os atores identificados para a utilização do sistema são pessoas desenvolvedoras, pessoas não desenvolvedoras e a Interface de Programação de Aplicação (API, do inglês *Application Programming Interface*) que são descritos a seguir.

Desenvolvedor: pessoa com conhecimento em tecnologias *web*, que pode criar e customizar suas páginas a partir do sistema proposto.

Há também os atores que não são programadores. Um deles é a pessoa não desenvolvedora, que possui pouco ou nenhum conhecimento em tecnologias *web*. Ela pode criar suas telas a partir da aplicação com customização básica. O outro ator é, na verdade, uma API que permite a comunicação com o Ambiente de Desenvolvimento Integrado (IDE, do inglês *Integrated Development Environment*) *Visual Studio Code* (VSCode) através do editor "O que você vê é o que você obtém" (WYSIWYG, do inglês *What You See Is What You Get*). Essa interface é denominada *WebAPI custom editor*, e ela fornece *feedback* (comentários) sobre aspectos ergonômicos, acessíveis e acerca das leis de Gestalt.

2.2 Modelos de Usuários

O projeto modela seus usuários por meio de *personas*, seguindo a definição de Courage e Baxter (2005), abordando os seguintes itens: identidade, *status*, objetivos, habilidades e tarefas. As *personas* exemplificadas nas Tabelas 1, 2, 3 e 4 foram discorridas a partir de conversas com possíveis usuários/pessoas que tiveram interesse na proposta do sistema, e as características deles foram compiladas em *personas* fictícias, para melhor compreensão da abrangência do público deste trabalho.

 Fonte: freepik. Imagem livre de direitos autorais.	Ana Júlia, Designer, 26 anos. Trabalha numa empresa de tecnologia de médio porte (até 500 funcionários). Sempre gostou muito de *design*, mas essa paixão não se estende às tecnologias *web*. É uma pessoa proativa e inovadora, pertencente ao segmento de *marketing* da empresa na qual trabalha.
Status	Secundária
Objetivos	Desenvolver *layouts* criativos e que capturem a atenção do usuário. Deseja também transmitir claramente os objetivos e visões da empresa de onde trabalha para seu público alvo.
Habilidades:	Photoshop, CorelDraw, GIMP, pacote Office 365 intermediário. Possui bacharelado em Design.
Tarefas	Desenvolvimento de logotipos, banners, imagens publicitárias, entre outros. Algumas vezes, auxilia no *design* de páginas *web*, mesmo que não seja sua especialidade.

Tabela 1. Primeira *persona* identificada do sistema, nomeada Ana Júlia.

 Fonte: freepik. Imagem livre de direitos autorais.	Iara, 17 anos, cursa Ensino Médio Técnico em Informática. Ela está aprendendo a programar e tem dúvidas sobre qual área se especializar. A maioria de suas aplicações até o momento utilizaram o terminal do sistema operacional para serem executadas.
Status	Secundária
Objetivos	Aprender mais sobre a área de desenvolvimento de software, escolher uma área na qual se especializar, melhorar o desenvolvimento de suas aplicações
Habilidades:	Algoritmos, Python, pacote Office 365 básico, inglês básico, formatar computadores.
Tarefas	--

Tabela 2. Segunda *persona* identificada do sistema, nomeada Iara.

 Fonte: freepik. Imagem livre de direitos autorais.	Felipe Dias, 32 anos, desenvolvedor *full stack/back-end*. Seu foco maior é em *back-end* e foi contratado como tal, mas há casos em que precisa realizar tarefas relacionadas ao *front-end*, seja por falta de prazo ou alocação de mão de obra. Gosta de resolver problemas lógicos e está acostumado a trabalhar com metodologias ágeis.
Status	Secundário
Objetivos	Se especializar em sua área de atuação no mercado de trabalho, entregando sempre códigos limpos e eficientes, mas sem deixar de atender outras demandas que possam fugir dessa área.
Habilidades:	C#, ASP.NET framework, MongoDB, SQL(*Standard Query Language*), SQL Server, HTML (*HyperText Markup Language*), CSS (*Cascading Style Sheet*), JavaScript, SOAP (*Simple Object Access Protocol*)/REST (*Representational State Transfer*), *Serverless* (sem servidor, tradução literal), Docker
Tarefas	Desenvolver novas funcionalidades para as aplicações

	já existentes em seu trabalho, melhoria e *review* de códigos, desenvolver páginas *web* para novas verticais (projetos externos ao qual ele atua) da empresa em que trabalha, *pair programming*, auxiliar novos funcionários da empresa, sugerir soluções para as aplicações em que atua.

Tabela 3. Terceira *persona* identificada do sistema, nomeada Felipe Dias.

 Fonte: freepik. Imagem livre de direitos autorais.	Henrique Santos, Desenvolvedor *Front-end*, 30 anos. Ele trabalha em uma empresa de pequeno porte da área de tecnologia e foi designado a desenvolver páginas *web* focadas na experiência de usuário. Porém não possui conhecimento aprofundado em Interface de Usuário/Experiência de Usuário (UI/UX, do inglês *User Interface/User Experience*) e a empresa em que trabalha não pretende contratar um especialista dessa área.
Status	Primária
Objetivos	Desenvolver sistemas web que sejam amigáveis e acessíveis para os usuários.
Habilidades:	HTML, CSS, JavaScript, Jquery, React js, Vue js
Tarefas	Resoluções de bugs, integração dos sistemas com *web service* API, desenvolver páginas *web* e otimizar interfaces para melhorar a performance e garantir boa usabilidade.

Tabela 4. Quarta *persona* identificada do sistema, nomeada Henrique Santos.

2.3 Modelo de Casos de Uso e Histórias de Usuários

Nesta seção, é possível observar na Figura 1 como os casos de usos interagem com os usuários e com o sistema em si e a listagem das histórias do usuário definidas para o sistema. Elas são a materialização do escopo da aplicação. Para fins de organização, as histórias se encontram numeradas de 1 a 9 e rotuladas com a sigla US (História de Usuário, do inglês *User Story*), sendo essa ordem meramente organizacional e não significando uma ordem de prioridade.

1. US1: Maria, como desenvolvedora *front-end*, deseja se concentrar no *design* da página;

2. US2: Roberto, como desenvolvedor, deseja adicionar funcionalidades de acessibilidade de forma automatizada;
3. US3: Lediane, como desenvolvedora, deseja criar páginas amigáveis para o usuário;
4. US4: Henrique, como *designer*, deseja desenvolver uma página *web*;
5. US5: Joana, como desenvolvedora e utilizadora da extensão, deseja reutilizar um *layout* já criado pela mesma extensão;
6. US6: Raissa, como desenvolvedora, deseja *feedback* ergonômico sobre a estrutura de suas páginas;
7. US7: Felipe, como desenvolvedor, deseja auxílio ao criar o *front-end* de uma aplicação cujo público alvo possui algum tipo de deficiência visual, auditiva ou motora;
8. US8: Andreia, como *designer*, deseja criar um *layout* de página *web* que seja responsivo para vários tipos de dispositivos;
9. US9: Floriano, como desenvolvedor *back-end*, deseja uma maneira fácil de elaborar o *front-end* de suas páginas *web*;

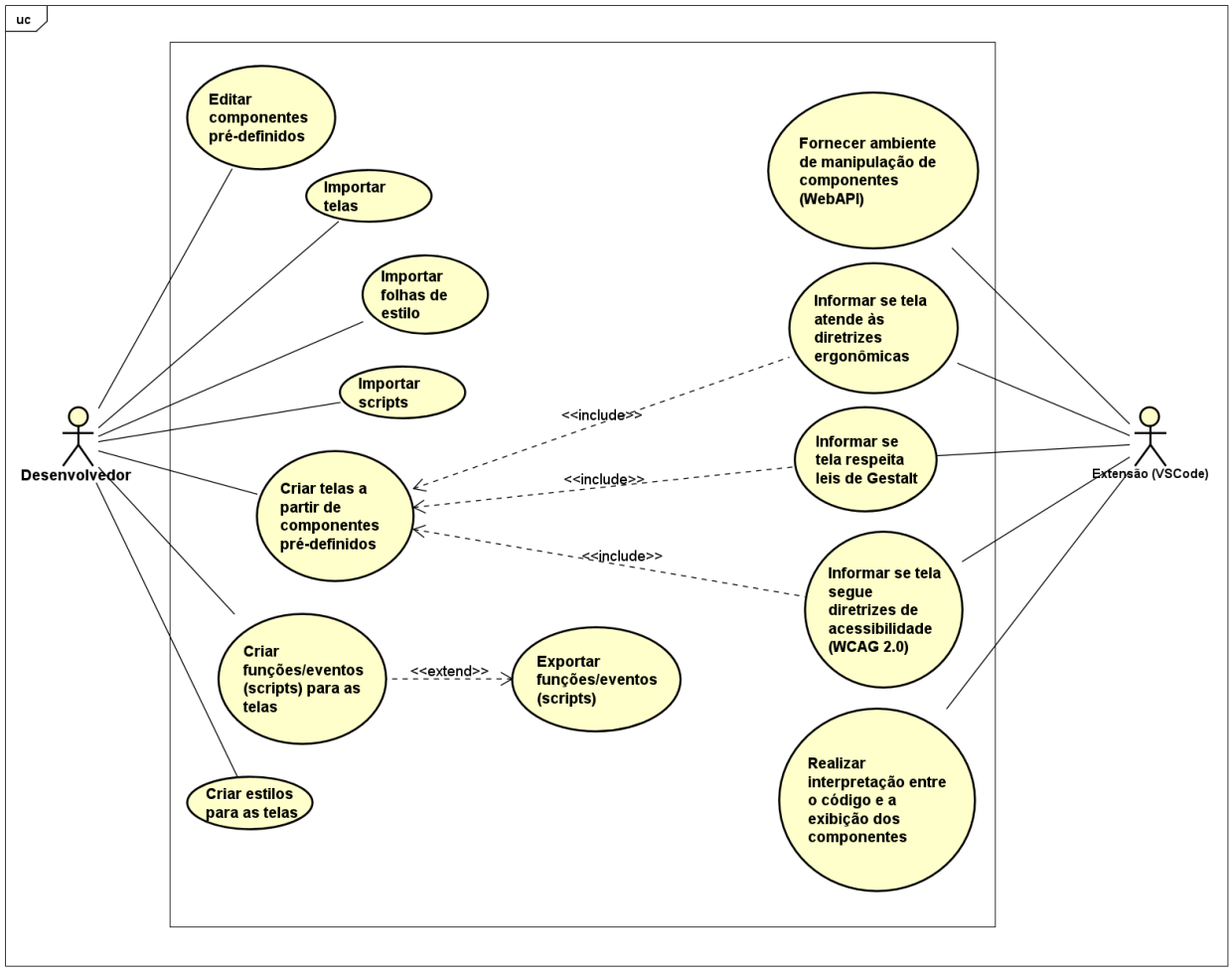

Figura 1. Diagrama de casos de uso do sistema.

2.4 Diagrama de Sequência do Sistema e Contratos de Operações

Nesta seção, apresentam-se os diagramas de sequência do sistema, baseados nos diagramas de casos de uso apresentados anteriormente. Foram feitos diagramas para: criação de páginas e selecionar componente (Figura 2) , abrir projeto e abrir arquivo (Figura 3), criar funções/eventos (Figura 4), criar estilo (Figura 5), exportar scripts (Figura 6), feedback sobre leis de Gestalt/diretrizes ergonômicas e feedback de acessibilidade (Figura 7). Para cada diagrama apresentado, há várias tabelas de seus contratos de operações.

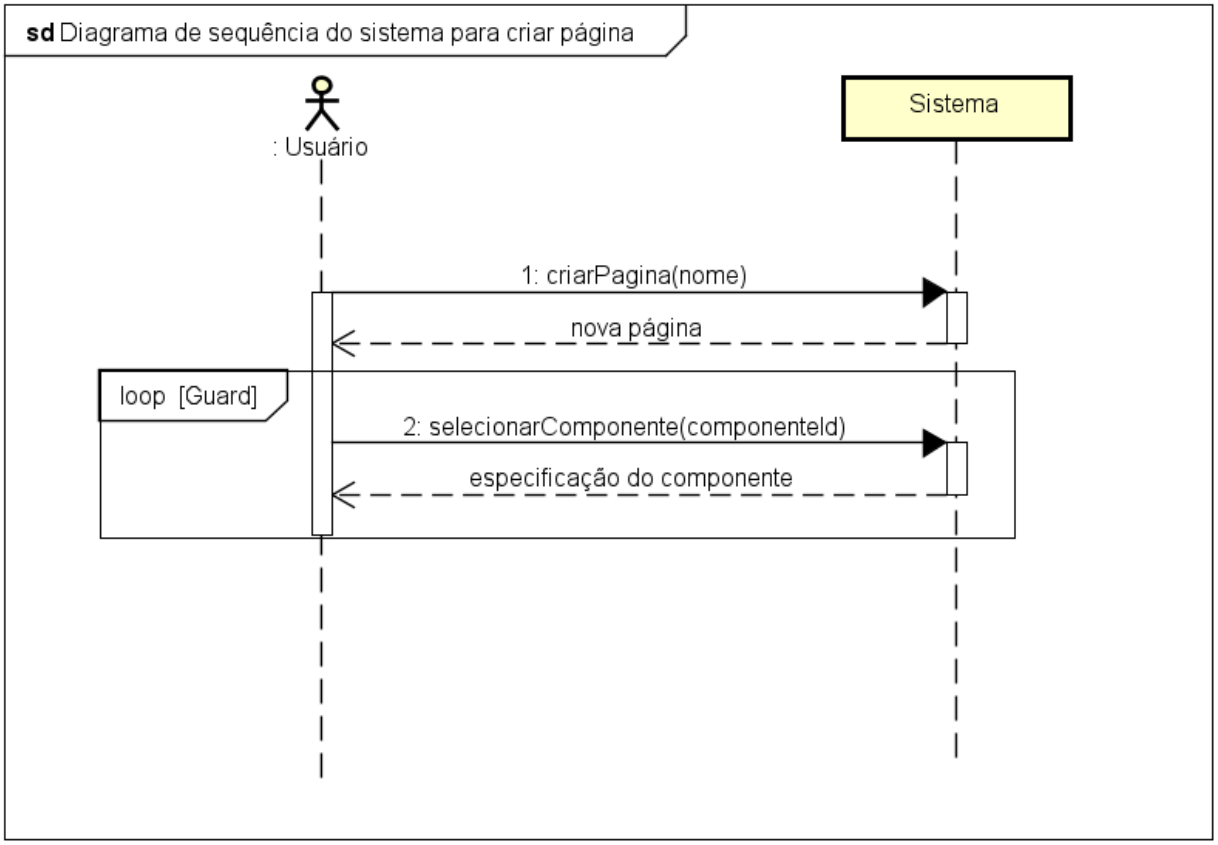

Figura 2. Diagrama de sequência do sistema para criar página.

Formato para cada contrato de operação mostrado na Figura 2 nas Tabelas 5 e 6.

Contrato	criarPagina()
Operação	criarPagina(nome)
Referências cruzadas	Criar página a partir de componentes pré-definidos
Pré-condições	
Pós-condições	• Uma instância de pagina foi criada

Tabela 5. Contrato de criar página.

Contrato	selecionarComponente()
Operação	selecionarComponente(componenteId)
Referências cruzadas	Criar página a partir de componentes pré-definidos
Pré-condições	Criar página (local de manipulação do componente)
Pós-condições	Componente foi associado a uma página

Tabela 6. Contrato de selecionar componente.

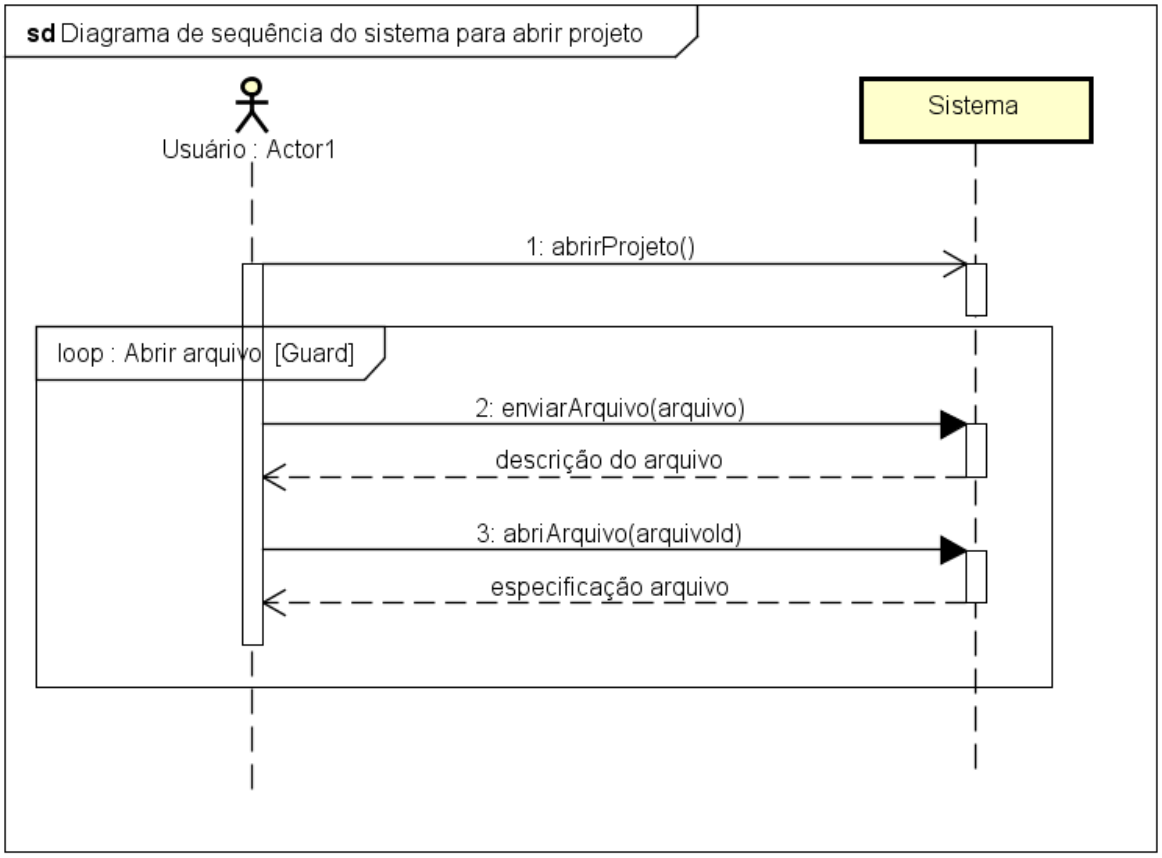

Figura 3. Diagrama de sequência do sistema para abrir projeto.

Formato para cada contrato de operação mostrado na Figura 3 nas Tabelas 7, 8 e 9.

Contrato	abrirProjeto()
Operação	abrirProjeto()
Referências cruzadas	• Importar folha de estilo • Importar páginas • Importar scripts
Pré-condições	

Pós-condições	Associação de arquivos ao projeto

Tabela 7 - Contrato de abrir projeto

Contrato	enviarArquivo()
Operação	enviarArquivo(arquivo)
Referências cruzadas	• Importar folha de estilo • Importar páginas • Importar scripts
Pré-condições	Projeto aberto
Pós-condições	Arquivo associado ao projeto

Tabela 8. Contrato de enviar arquivo.

Contrato	abrirArquivo()
Operação	abrirArquivo(arquivoId)
Referências cruzadas	• Importar folha de estilo • Importar páginas • Importar scripts
Pré-condições	Arquivo associado
Pós-condições	Visualização de dados de arquivo

Tabela 9. Contrato de abrir arquivo.

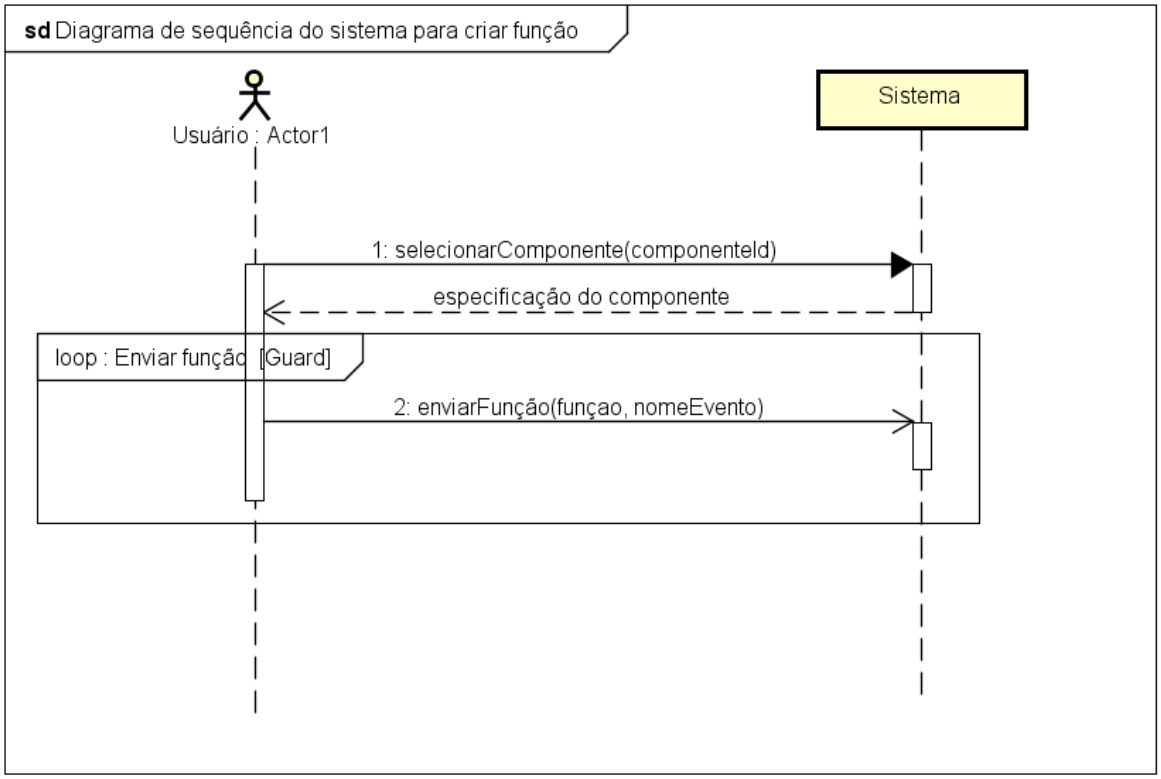

Figura 4. Diagrama de sequência do sistema para criar funções/eventos (*scripts*) para as páginas.

Formato para cada contrato de operação mostrado na Figura 4 nas Tabelas 10 e 11.

Contrato	selecionarComponente()
Operação	selecionarComponente(componenteId)
Referências cruzadas	Criar páginas a partir de componentes pré-definidos
Pré-condições	Criar páginas / Importar páginas
Pós-condições	Componente foi associado a uma página

Tabela 10. Contrato de selecionar componente.

Contrato	enviarFunção()
Operação	enviarFunção(função, nomeEvento)
Referências cruzadas	Criar funções/eventos (*scripts*) para as páginas
Pré-condições	• Criar páginas / Importar páginas • Seleção de componente pré-definido
Pós-condições	• Função foi associada a um evento • Evento foi associado ao componente

Tabela 11. Contrato de enviar função.

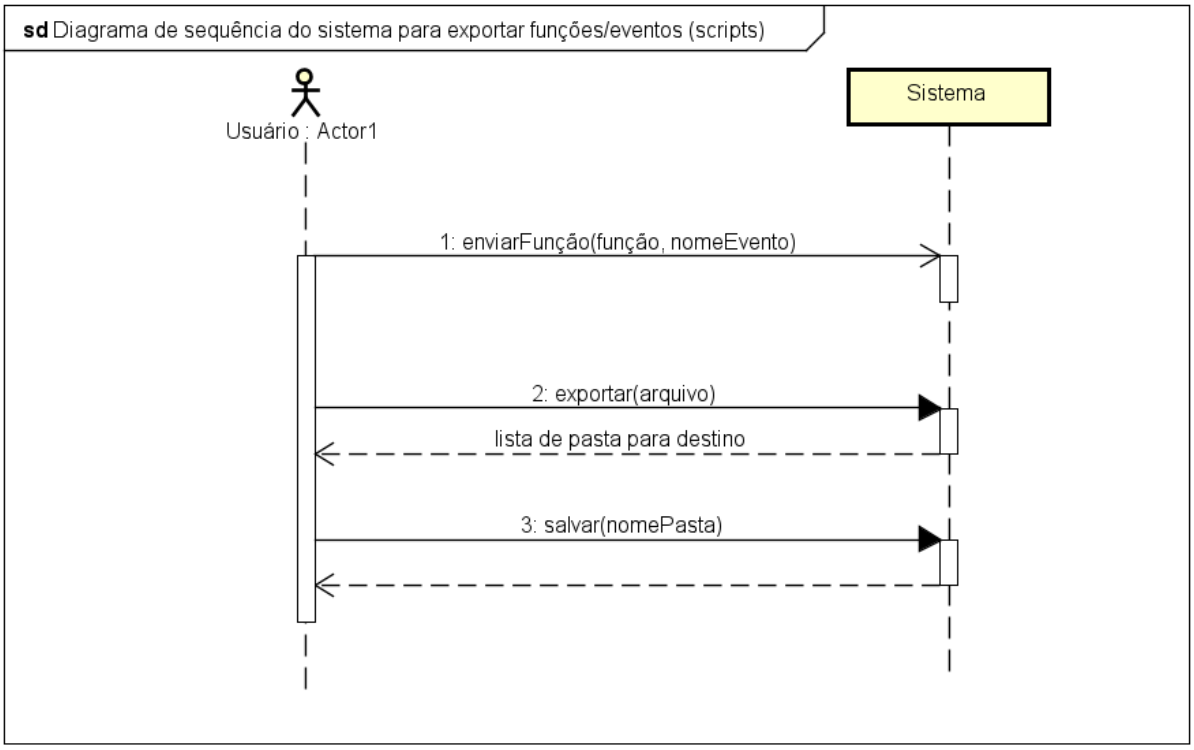

Figura 5. Diagrama de sequência do sistema para exportar funções/eventos (*scripts*).

Para o diagrama da Figura 5, o contrato da operação “enviarFunção” encontra-se na Tabela 11. Os contratos de exportar e salvar estão detalhados nas Tabelas 12 e 13.

Contrato	exportar()

Operação	exportar(arquivo)
Referências cruzadas	Exportar funções/eventos (scripts)
Pré-condições	Criar função (função JS para manipular componente)
Pós-condições	Arquivo (arquivo JS com as funções geradas)

Tabela 12. Contrato de exportar arquivo script.

Contrato	salvar()
Operação	salvar(nomePasta)
Referências cruzadas	Exportar funções/eventos (scripts)
Pré-condições	Criar função (função JS para manipular componente)
Pós-condições	Arquivo gerado (arquivo JS com as funções geradas)

Tabela 13. Contrato de salvar arquivo.

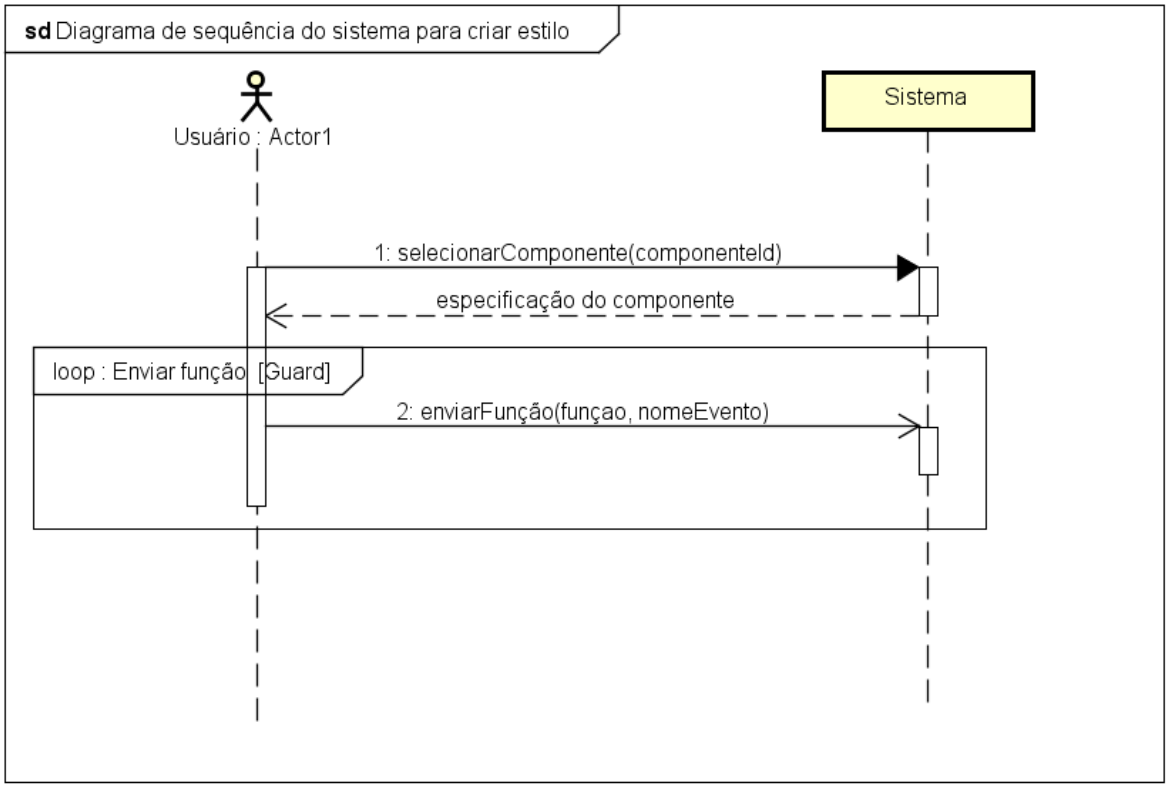

Figura 6. Diagrama de sequência do sistema para criar estilo para as páginas.

Formato para os contratos de operação de selecionar componente e alterar estilo exibidos nas Tabelas 14 e 15.

Contrato	selecionarComponente()
Operação	selecionarComponente(componenteId)
Referências cruzadas	Criar páginas a partir de componentes pré-definidos
Pré-condições	Criar páginas / Importar páginas

Pós-condições	Componente foi associado a uma página

Tabela 14 - Contrato de salvar componente

Contrato	alterarEstilo()
Operação	alterarEstilo(propriedadeCSS, valor, componenteId)
Referências cruzadas	Criar estilos para as páginas
Pré-condições	• Criar páginas / Importar páginas • Criar componente
Pós-condições	Componente foi alterado

Tabela 15. Contrato de alterar estilo.

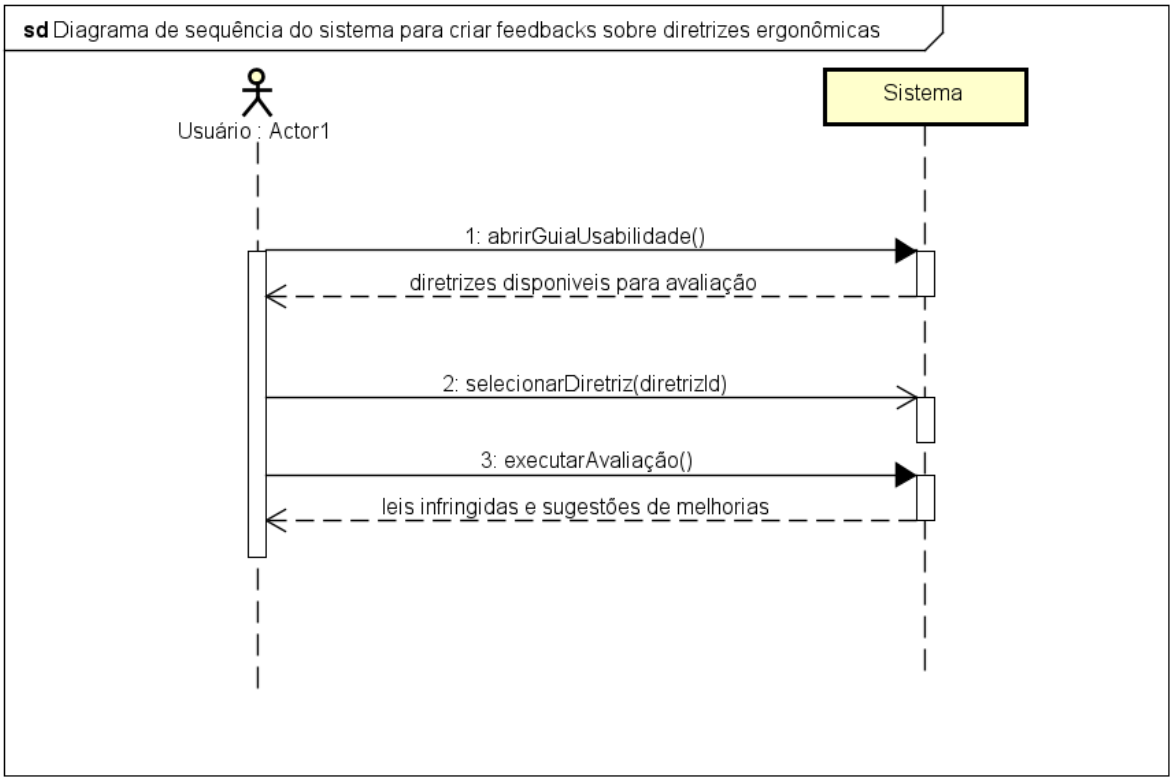

Figura 7. Diagrama de sequência do sistema para criar feedbacks sobre diretrizes ergonômicas para as páginas.

Formato para os contratos de abrir guia de usabilidade, selecionar diretriz e executar avaliação exibidos nas Tabelas 16, 17 e 18, respectivamente.

Contrato	abrirGuiaUsabilidade()
Operação	abrirGuiaUsabilidade()
Referências cruzadas	• Informar se página atende às diretrizes ergonômicas • Informar se página respeita leis de Gestalt

	• Informar se página segue diretrizes de acessibilidade (WCAG 2.0)
Pré-condições	• Criar páginas / Importar páginas
Pós-condições	Visualização de diretrizes

Tabela 16 - Contrato de abrir guia de usabilidade

Contrato	selecionarDiretriz()
Operação	selecionarDiretriz(diretrizId)
Referências cruzadas	• Informar se página atende às diretrizes ergonômicas • Informar se página respeita leis de Gestalt • Informar se página segue diretrizes de acessibilidade (WCAG 2.0)
Pré-condições	• Criar páginas/ Importar páginas
Pós-condições	Instância de diretriz

Tabela 17. Contrato de selecionar diretriz.

Contrato	executarAvaliação()
Operação	executarAvaliação()
Referências cruzadas	• Informar se a página atende às diretrizes ergonômicas • Informar se a página respeita leis de Gestalt • Informar se a página segue diretrizes de acessibilidade (WCAG 2.0)
Pré-condições	• Criar página / Importar página • Selecionar diretriz
Pós-condições	

Tabela 18. Contrato de executar avaliação.

3. Modelos de Projeto

Nesta seção são apresentados os modelos de projeto. Essa apresenção é composta pelo diagrama de Classes (Seção 3.1), diagramas de sequência (Seção 3.2), diagramas de comunicação (Seção 3.3), arquitetura lógica (Seção 3.4), diagramas de estado (Seção 3.5) e pelos diagramas de componentes e implantação (Seção 3.6).

3.1 Diagrama de Classes

O diagrama de classes apresentado na Figura 8 contém 7 classes: “Feedback”, “Página” e “Componentes”, cuja classe serve de base para os componentes pré-definidos criados na extensão. As classes “Paginação”, “Tabela”, “Stepper” e “Autocomplete” são para componentes que precisaram de atributos ou funções adicionais em comparação à classe “Componente”. Todos os componentes pré-definidos, tanto em estrutura quanto em layout e em funcionamento, podem ser vinculados a várias páginas quantas vezes o usuário precisar. Sendo assim, haverá um identificador único para cada instância de componente.

É importante destacar que esse identificador é diferente do atributo “id” de uma tag HTML, pois enquanto o identificador interno (“idComponente”) será usado para identificar a instância do componente, o “id” de uma tag (“idHTML”) é usado para manipulação de estilo ou funcionamento. Por exemplo: o “idComponente” pode ter valor “123_456”, que é gerado internamente pela aplicação; enquanto o “idHTML” é informado pelo usuário e pode ter um valor como “caixaDeTexto”.

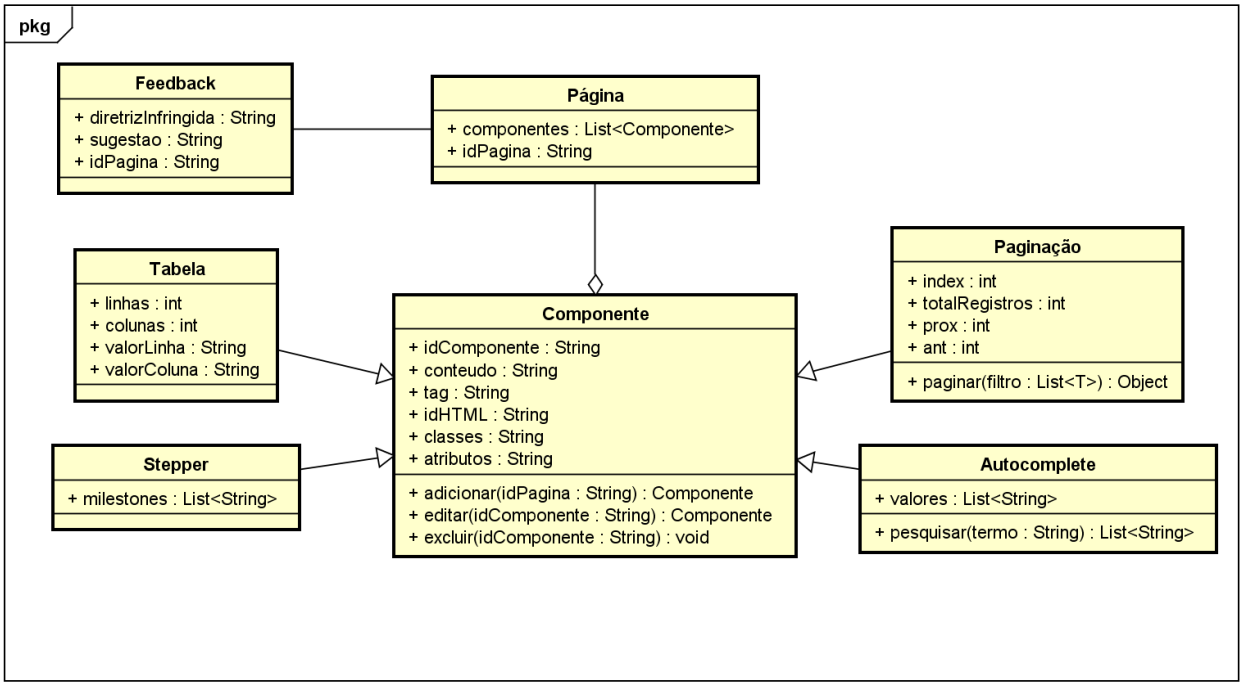

Figura 8. Diagrama de Classes do sistema.

3.2 Diagramas de Sequência

Nesta seção são apresentados os diagramas de sequência do sistema, sendo eles:

1. **Criar páginas, a partir de componentes pré-definidos:** é necessária interação com o editor WYSIWYG para exibir os componentes nas páginas e da própria página em si, já que ela recebe os componentes e os renderiza para o usuário. A "Interface UI Guide" é onde o editor WYSIWYG se encontra. A troca de mensagens entre "Usuário", "Interface UI Guide" e "Página" é exibida na Figura 9.

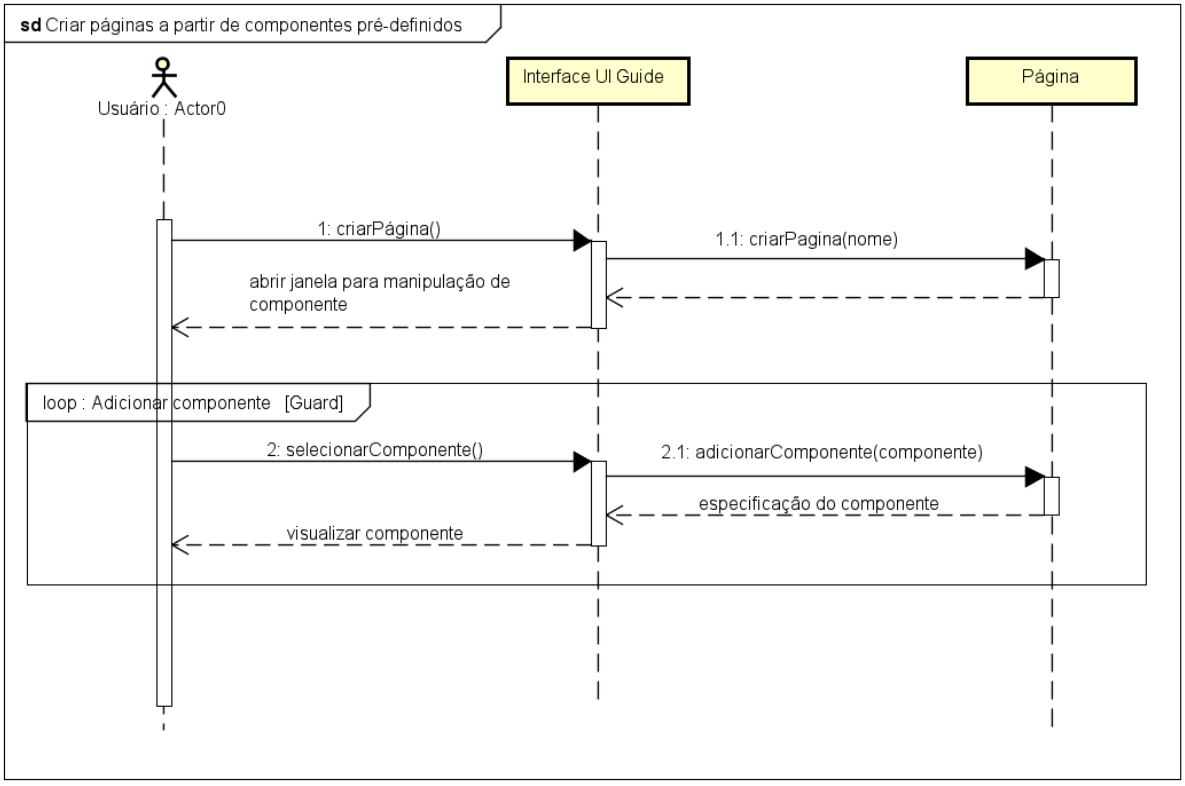

Figura 9. Diagrama de Sequência para criar páginas a partir de componentes pré-definidos.

2. **Criar funções/eventos (scripts) para as páginas:** para adicionar uma função ou evento, é necessário fazê-lo dentro de um componente. O usuário informa as funções/eventos em espaço disponibilizado pelo editor WYSIWYG, com o componente selecionado, e a página atualiza as mudanças, como mostrado na Figura 10. É possível adicionar vários eventos e funções, conforme necessidade do usuário.

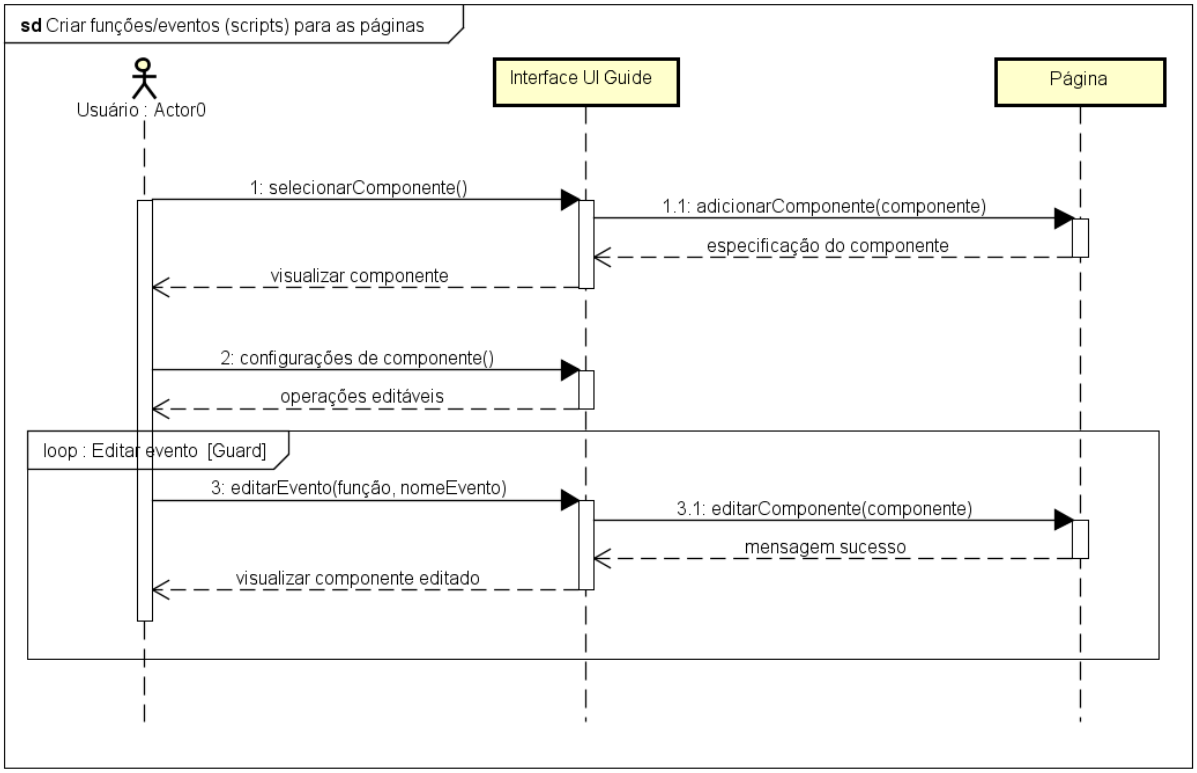

Figura 10. Diagrama de Sequência para criar funções/eventos (scripts) para as páginas.

3. **Criar estilos para as páginas:** semelhante ao funcionamento das funções e eventos. O usuário informa o estilo a ser aplicado em espaço fornecido pelo editor WYSIWYG, com o componente selecionado, e a página aplica as mudanças quando forem salvas, como pode ser observado na Figura 11.

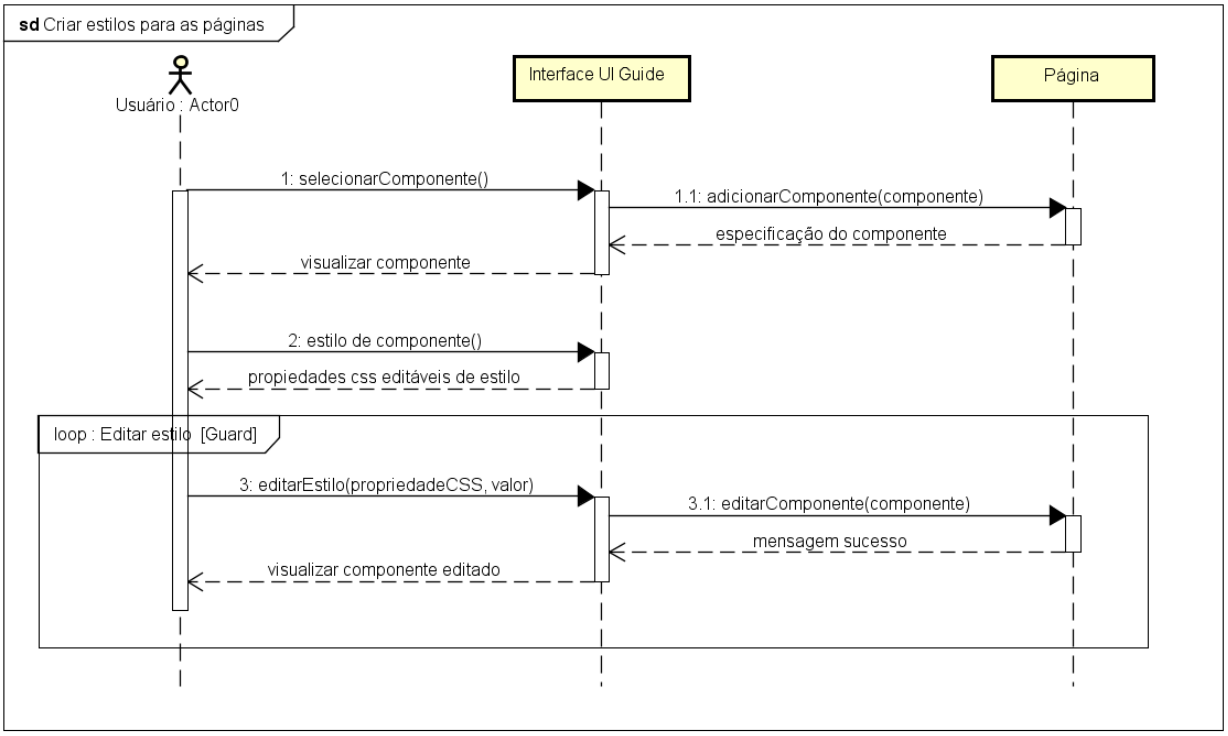

Figura 11. Diagrama de Sequência para criar estilos para as páginas.

4. **Informar se a página atende as diretrizes ergonômicas/leis de Gestalt:** quando a opção de fornecer feedback for selecionada, será realizada uma análise acerca dos quesitos mencionados e uma resposta será fornecida para o usuário no editor WYSIWYG, como demonstrado na Figura 12.

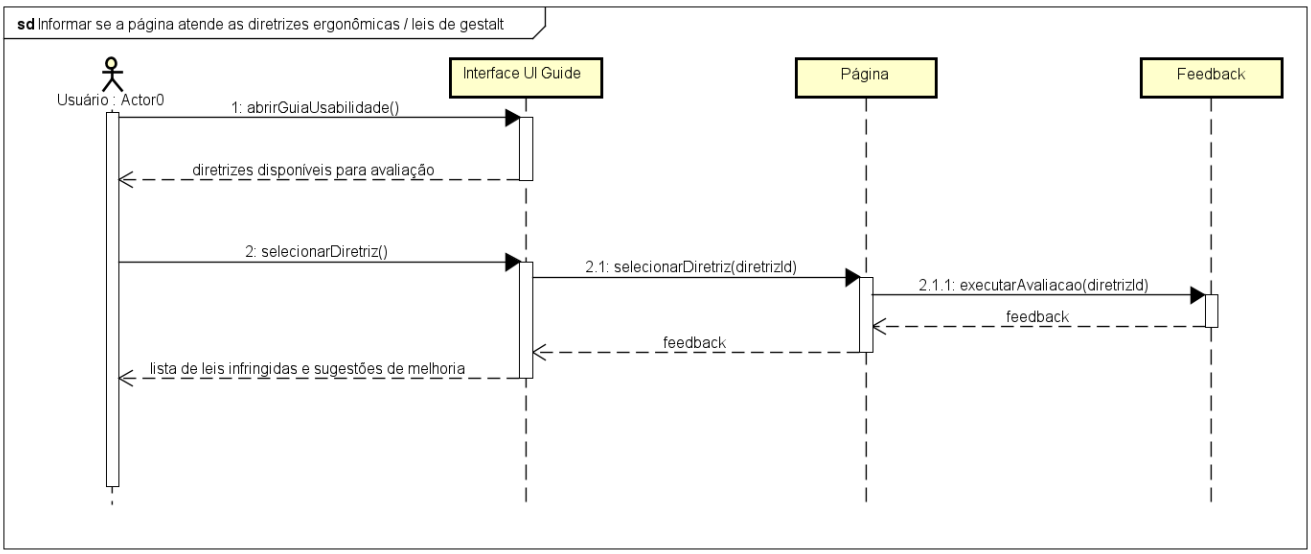

Figura 12. Diagrama de Sequência para informar se a página atende às diretrizes ergonômicas/ leis de gestalt.

5. **Informar se a página atende às diretrizes de acessibilidade:** a diferença do diagrama apresentado na Figura 13 para o diagrama da Figura 11 é o uso de uma API externa para verificar as diretrizes de acessibilidade. Esta API devolverá a análise da página, e a análise será formatada e exibida para o usuário no editor WYSIWYG.

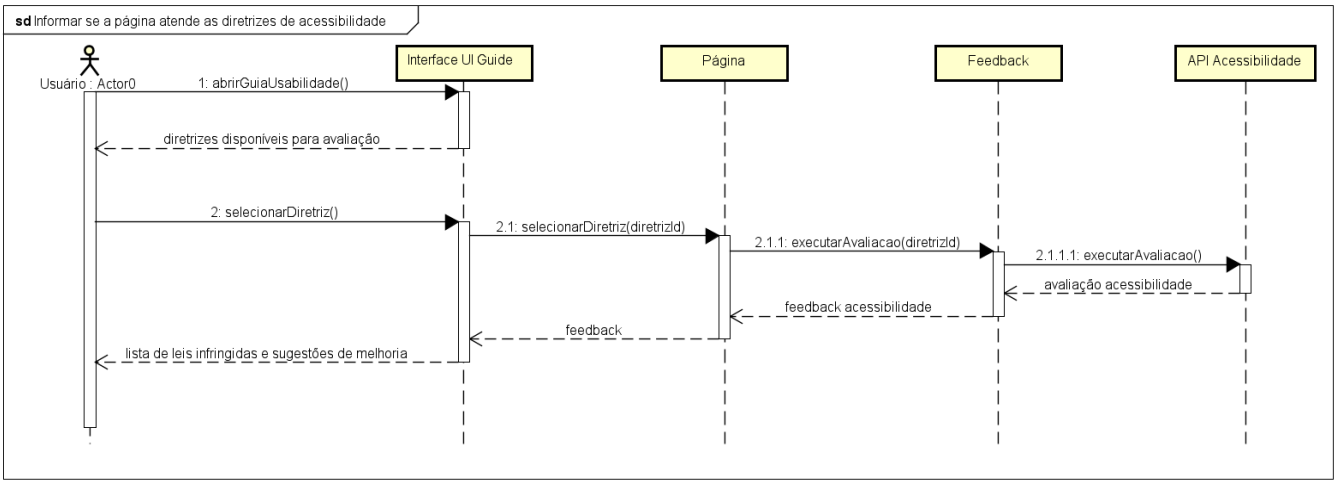

Figura 13. Diagrama de Sequência para informar se a página atende às diretrizes de acessibilidade.

6. **Importar páginas:** o usuário seleciona a pasta na qual exportou suas páginas anteriormente para importá-las de volta à aplicação, como é observado na Figura 14.

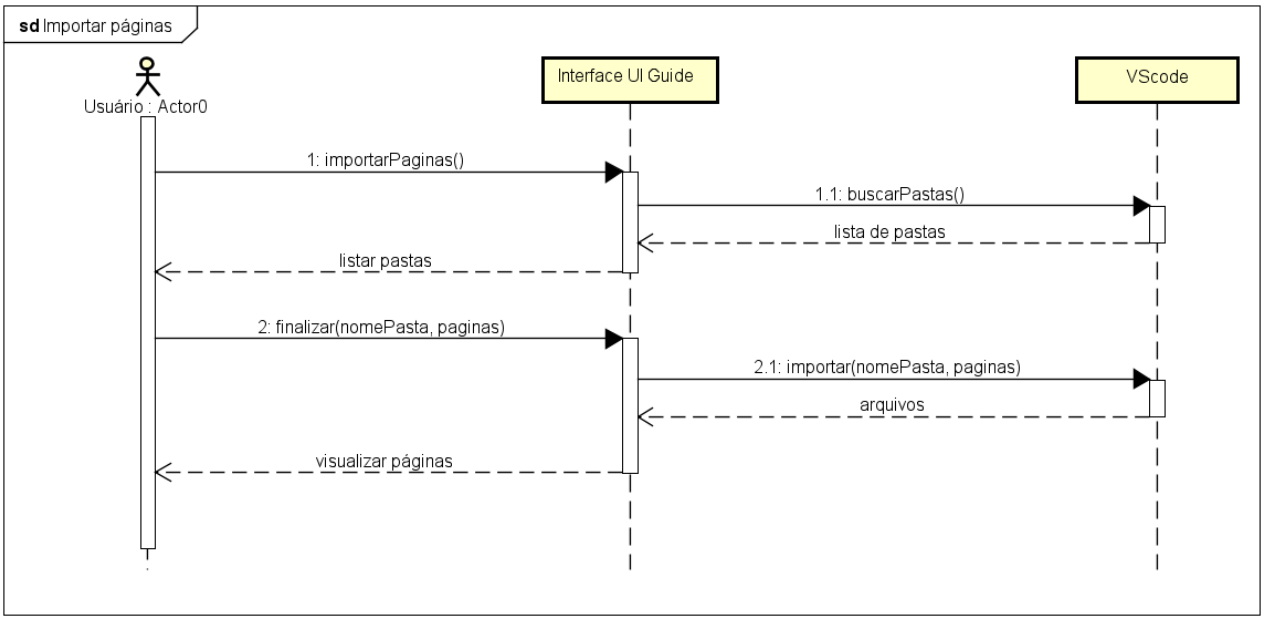

Figura 14. Diagrama de Sequência para importar uma página.

3.3 Diagramas de Comunicação

Nesta seção são apresentados os diagramas de comunicação, baseados nos diagramas de sequência realizados no tópico anterior. As mensagens são, inicialmente, encaminhadas para a "Interface UI Guide", fazendo o papel de fronteira do sistema. A página foi considerada como parte da linha de vida da comunicação, como é possível observar respectivamente nas Figuras 15, 16, 17, 18, 19 e 20.

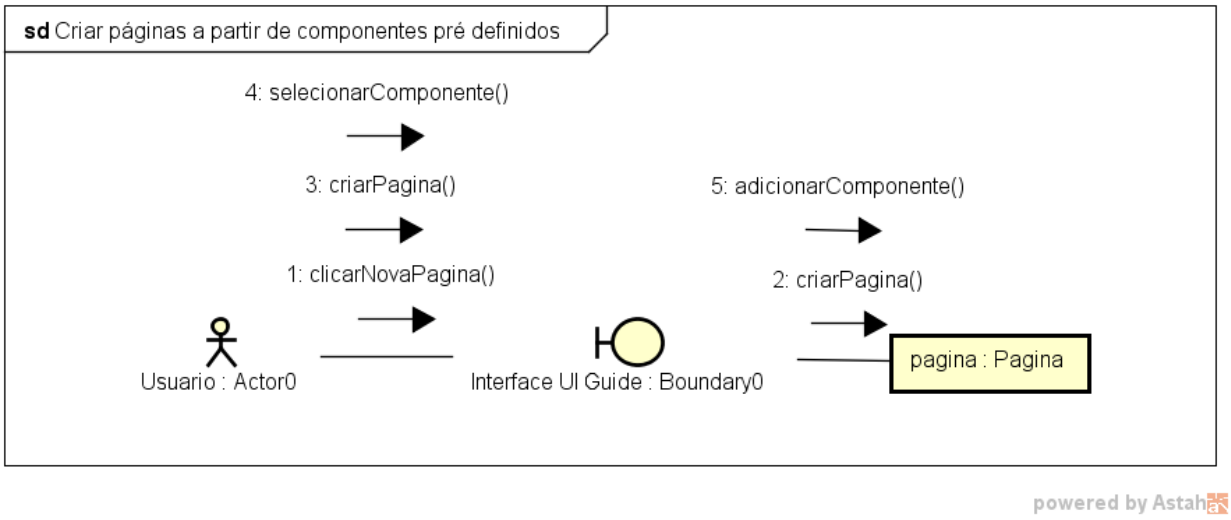

Figura 15. Diagrama de comunicação para criação de páginas.

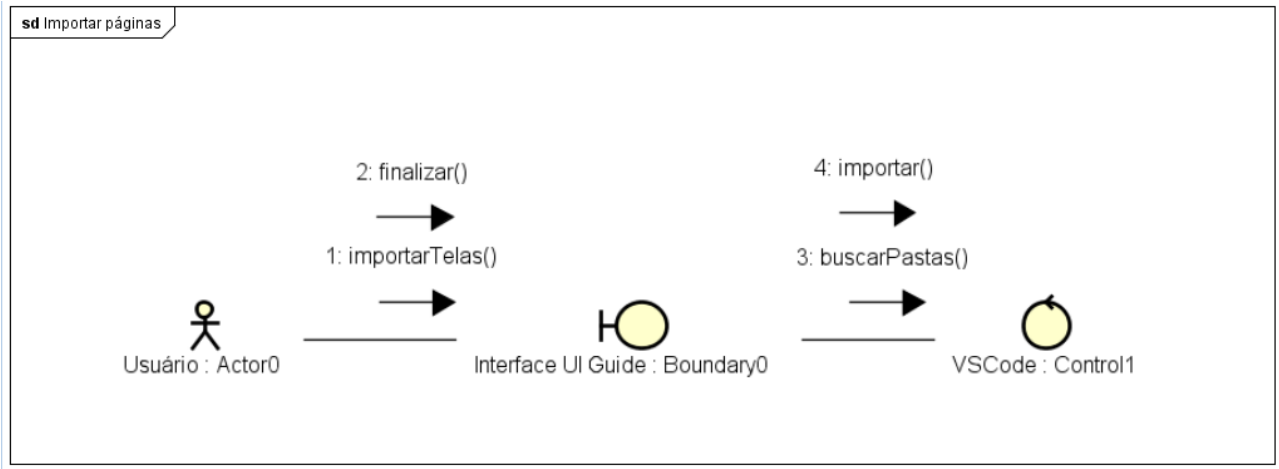

Figura 16. Diagrama de comunicação para importar páginas.

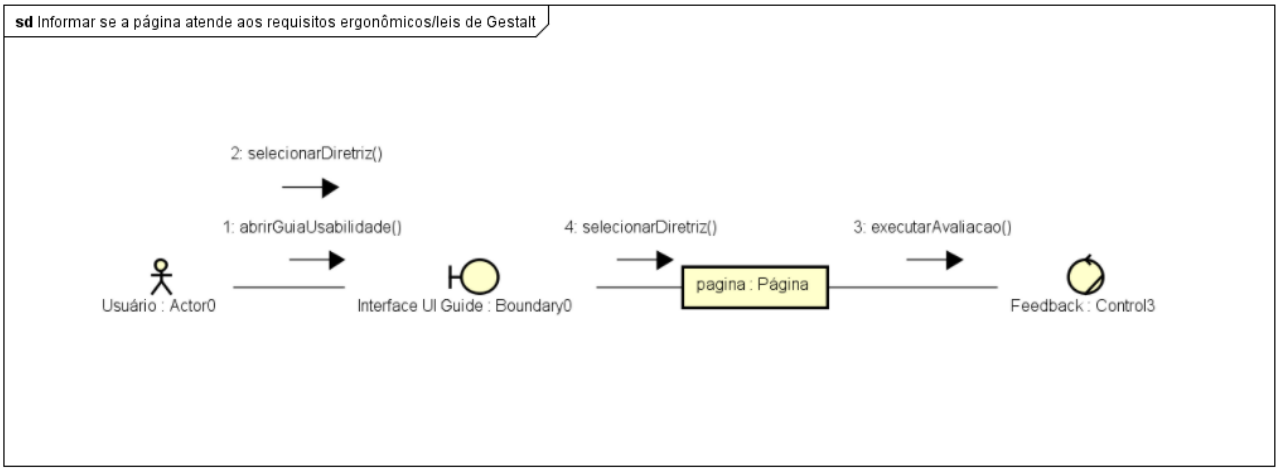

Figura 17. Diagrama de comunicação para quesitos ergonômicos e leis de Gestalt.

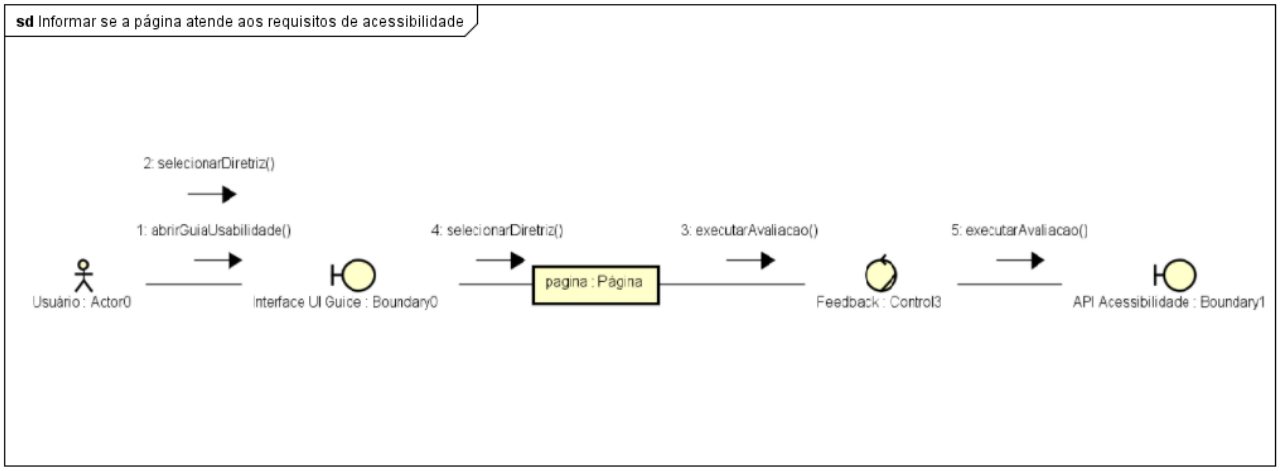

Figura 18. Diagrama de comunicação para quesitos de acessibilidade.

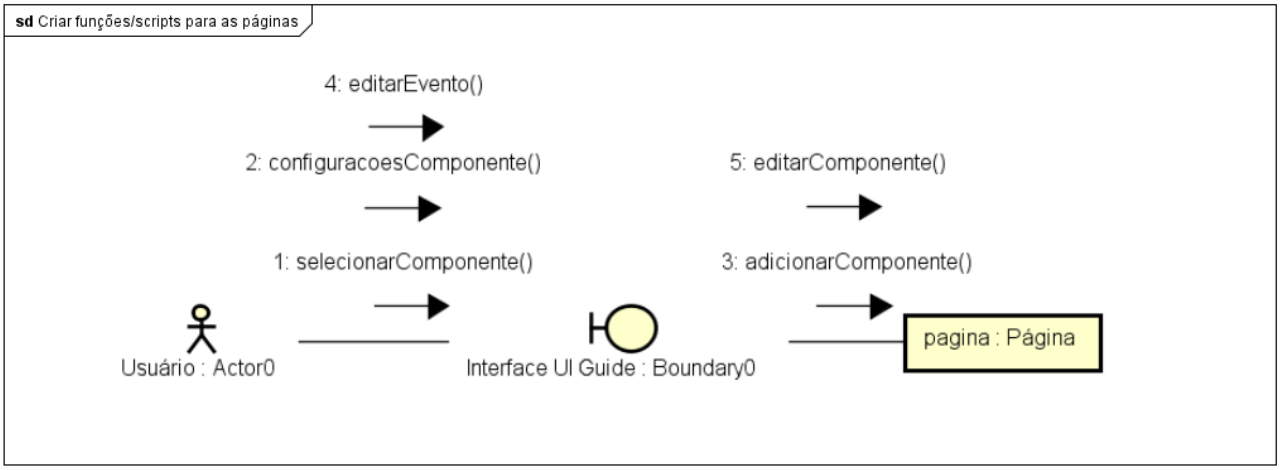

Figura 19. Diagrama de comunicação para criação de eventos/scripts para as páginas.

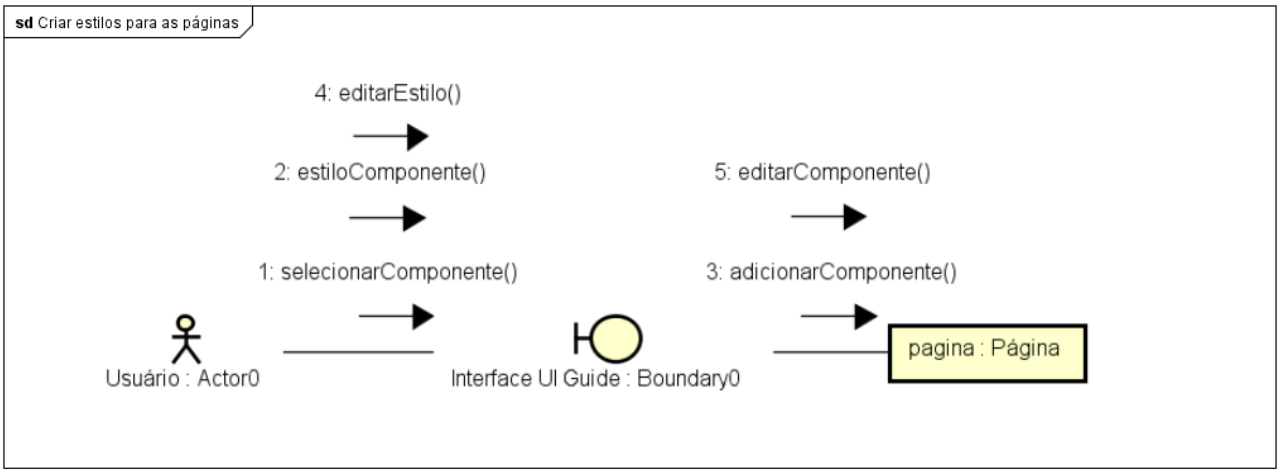

Figura 20. Diagrama de comunicação para criação de estilos para as páginas.

3.4 Arquitetura Lógica

A arquitetura escolhida para implementação do projeto foi a de padrão de microsserviços, tendo em vista que tanto o contexto de extensão do VSCode quanto a API de acessibilidade podem ser considerados microsserviços. Logo, os outros feedbacks necessários para a aplicação - sobre leis de Gestalt e diretrizes ergonômicas - também serão desenvolvidos dessa forma, facilitando seu uso em outras aplicações e agregando à abordagem de código aberto. As páginas, por mais que sejam representadas de forma externa na Figura 21, não serão desenvolvidas na forma de microsserviços, tendo em vista que, isoladas, as páginas não terão funcionalidade suficiente para serem consideradas serviços.

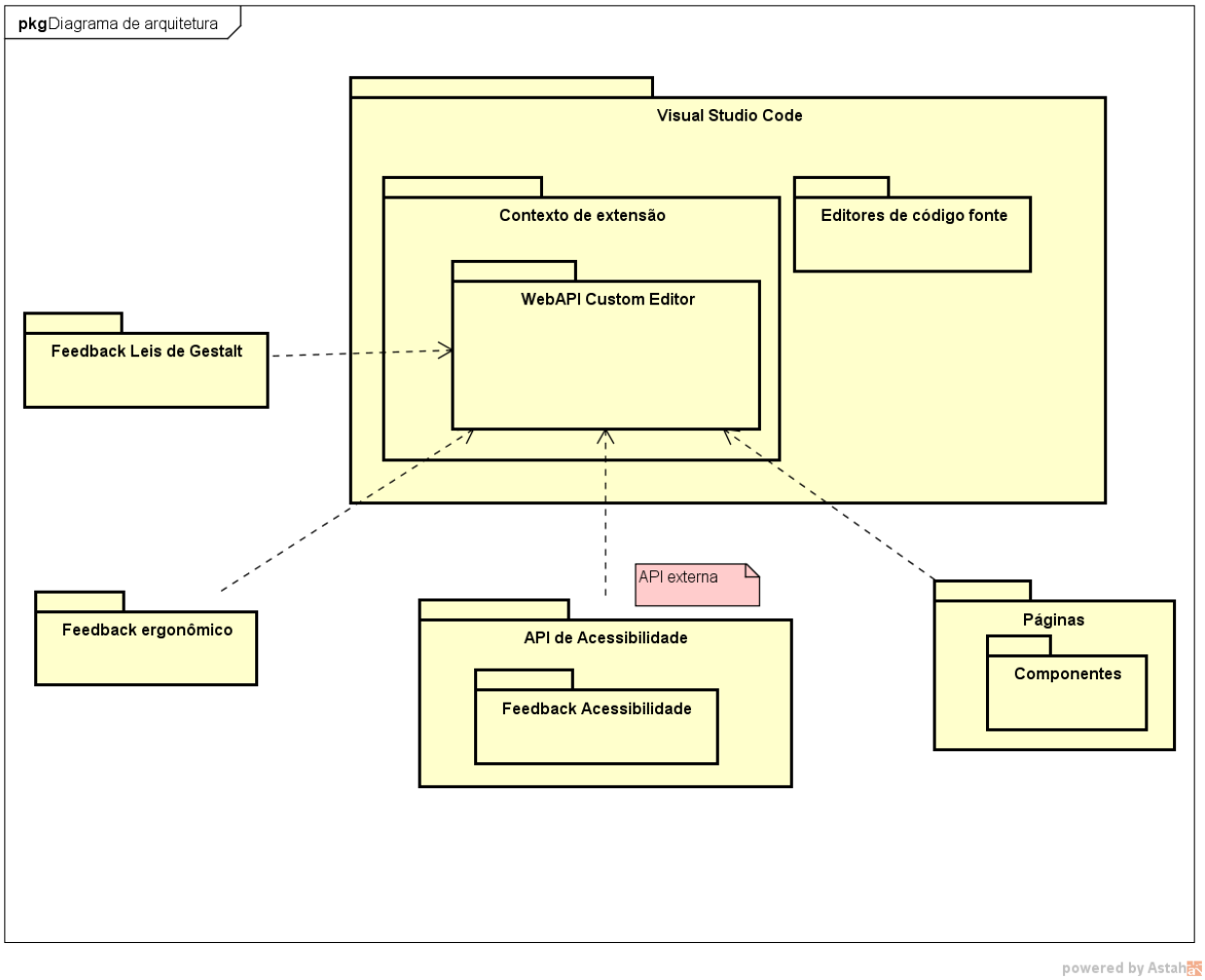

Figura 21. Arquitetura proposta para o sistema, seguindo o padrão de microsserviços.

3.5 Diagramas de Estados

Foram feitos dois diagramas de estados para aplicação, considerando os estados que uma página pode passar durante sua elaboração. Ele é demonstrado na Figura 22. Há também o diagrama de estado para o feedback, na Figura 23, pois ele será alterado conforme as alterações nas páginas forem realizadas. A página pode ter seus estilos e eventos/funções editados, além de passar pela análise mencionada do feedback; e pode ser também exportada. Já o feedback pode ser positivo ou negativo (conter ou não infrações), além de também ser vinculado com a parte de sugestões, em caso de resposta negativa.

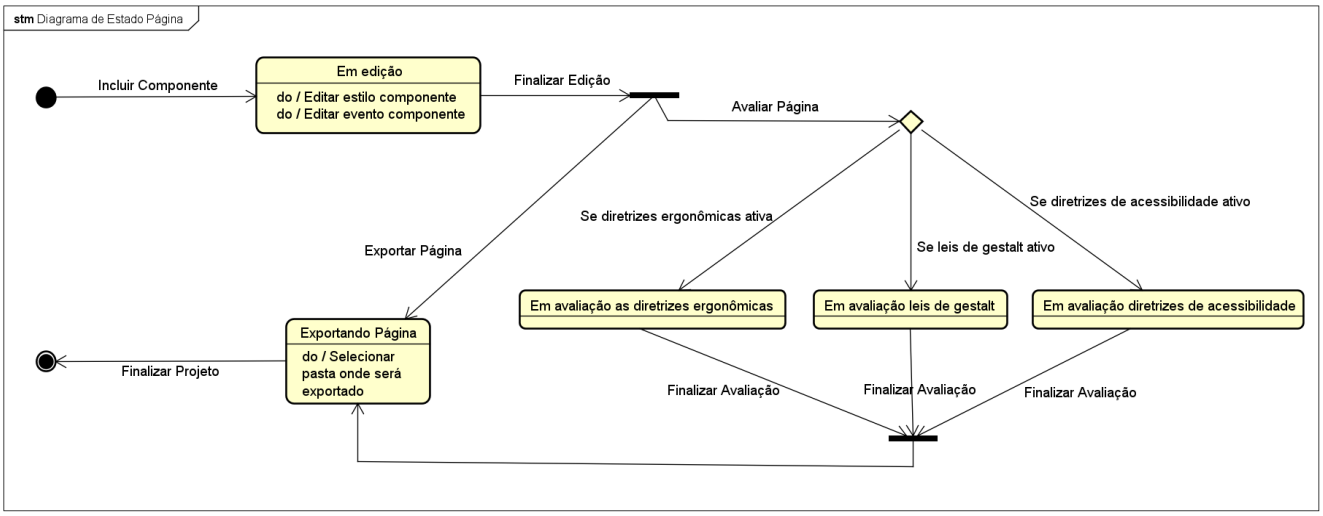

Figura 22. Diagrama de Estado da Página.

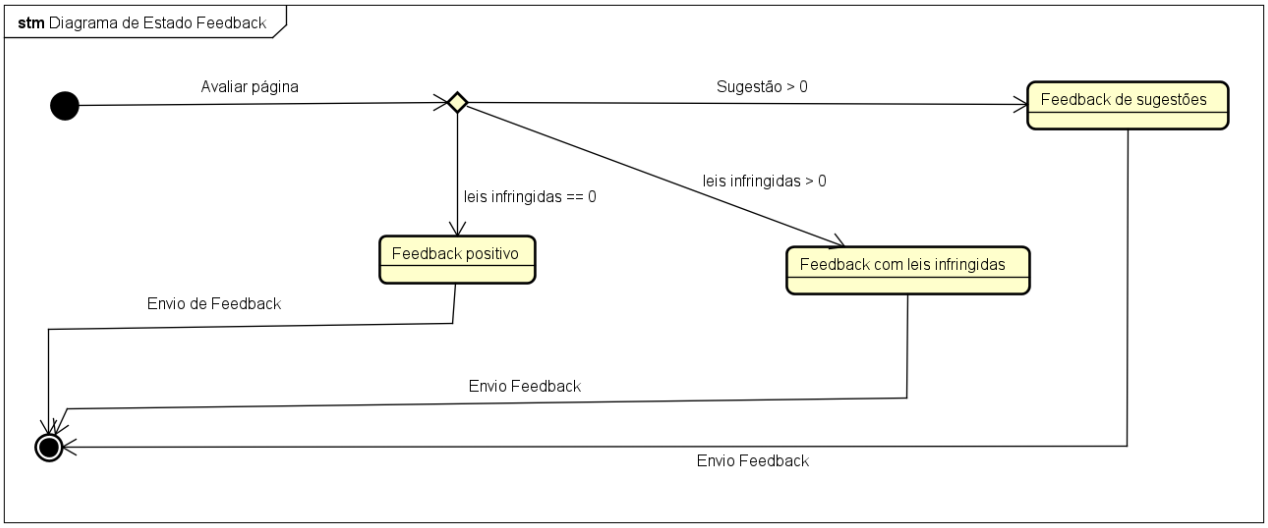

Figura 23. Diagrama de Estado do Feedback.

3.6 Diagrama de Componentes e Implantação

Foram considerados os seguintes aspectos na realização do diagrama de implantação: O sistema como um todo; as páginas que serão criadas dentro do sistema e os controladores necessários para que a manipulação das páginas seja realizada com sucesso. Este diagrama é exibido na Figura 24 e segue o padrão *Model View Controller* (MVC, do inglês Modelo, Visão e Controle).

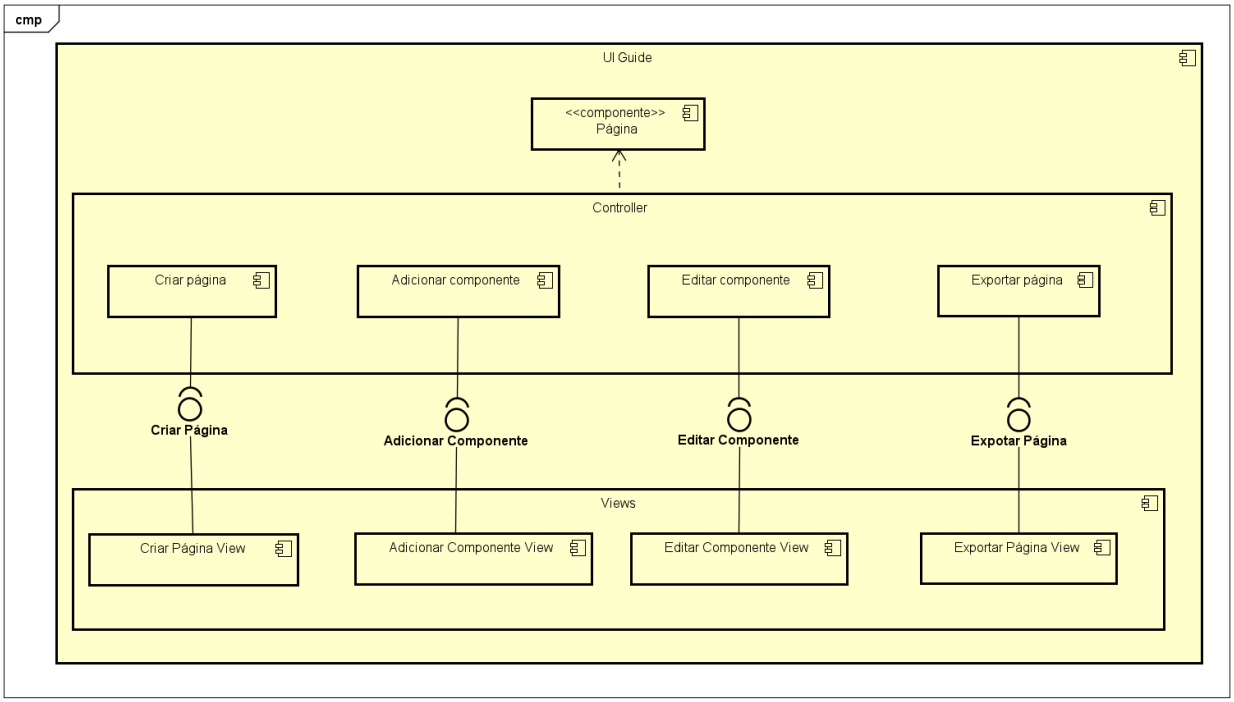

Figura 24. Diagrama de Componentes proposto para o sistema, seguindo o padrão MVC.

Para o diagrama de implantação apresentado na Figura 25, há a máquina do usuário na qual o VSCode está instalado. A extensão será instalada dentro do VSCode e fará a comunicação com a API de acessibilidade para gerar o feedback das páginas sobre diretrizes de acessibilidade.

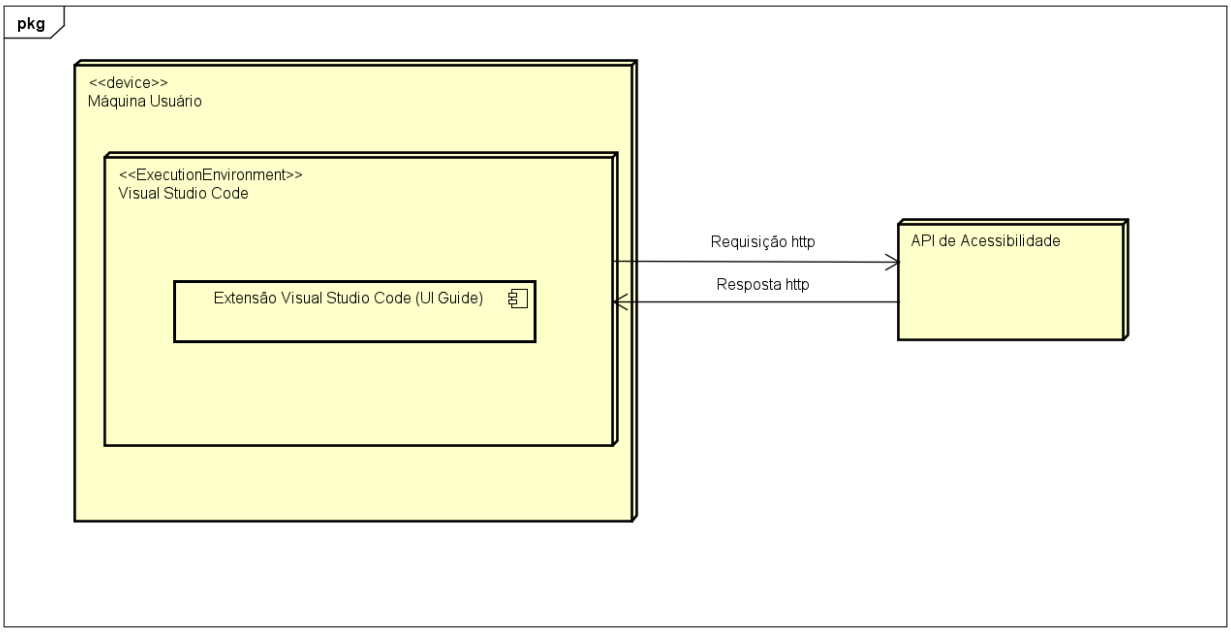

Figura 25. Diagrama de Implantação proposto para o sistema, com suas devidas relações.

4. Projeto de Interface com Usuário

Com o intuito de transmitir de maneira clara o formato do projeto desenvolvido, são apresentados nesta seção esboços das interfaces que permitem a interação com o usuário.

Na Figura 26 é possível visualizar o esboço da primeira página. Ao realizar a abertura do sistema, é possível visualizar uma barra de tarefas com suas principais funcionalidades, sendo elas: a ferramenta de manipulação do projeto, na qual é possível criar e importar arquivos; a ferramenta de componentes, na qual será possível selecionar componentes e arrastá-los para visualização na página; e o guia de *layout*, que exibe as opções de feedbacks sobre princípios de Gestalt, ergonomia e acessibilidade. Esta página também apresenta um quadro branco, no qual serão exibidas todas as edições realizadas e a possibilidade de redimensionar o tamanho da página.

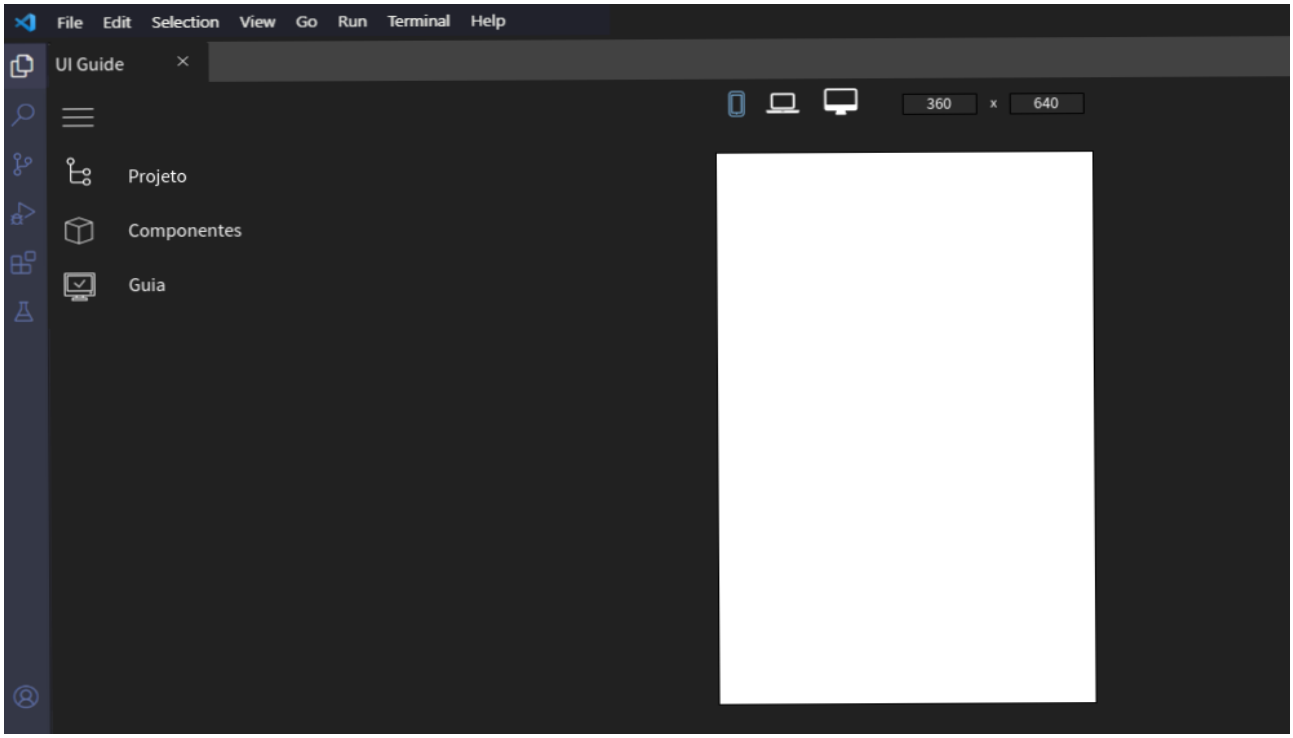

Figura 26. Tela inicial do sistema.

Ao selecionar o tópico “Projeto”, é possível visualizar a estrutura de arquivos com suas páginas, folhas de estilo, scripts e imagens. Além disso, há também a opção de elaborar um fluxo de uso para as páginas exportadas. Essas opções estão inclusas na Figura 27, e encontram-se por meio de um *dropdown* aberto para melhor visualização.

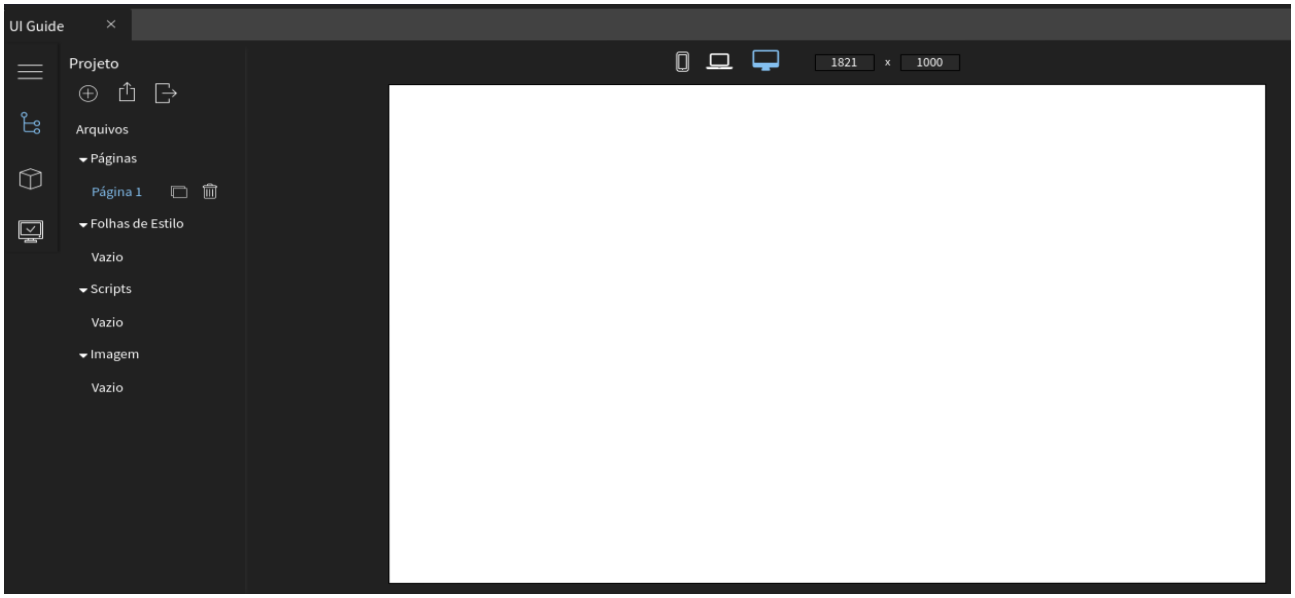

Figura 27. Tela inicial da aba projeto.

No topo esquerdo da Figura 28 há um botão para criar uma página. Ao clicar nele, é gerado um modal no qual é possível nomear a nova página.

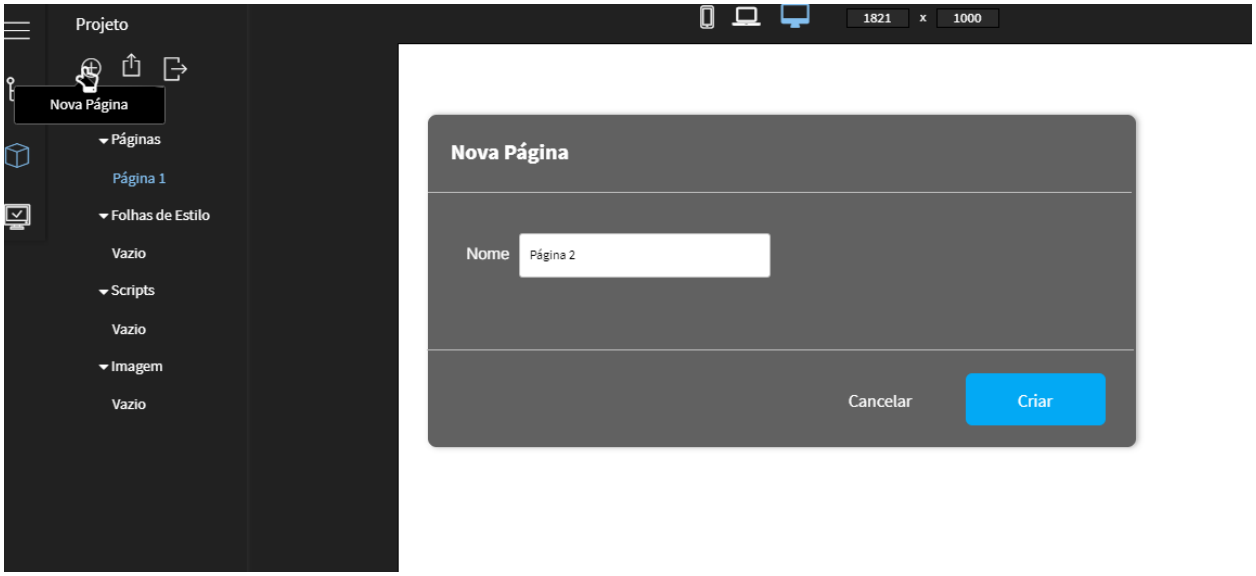

Figura 28. Tela criar nova página.

Ao lado do botão de criação de nova página, há o botão para realizar a importação de arquivos dentro da página criada. Esses arquivos podem ser folhas de estilo, imagens e arquivos de *scripts,* como mostrado na Figura 29. Para essa importação, o usuário terá duas opções: a de arrastar um arquivo para a tela e a de clicar no botão de "Selecionar um arquivo do seu dispositivo". Não há nenhuma divergência de ação para ambas as maneiras de importação.

Figura 29. Tela importar arquivos.

No menu componentes mostrado na Figura 30, são listados todos os componentes arrastáveis disponíveis no sistema. Para adicionar um componente à página, basta selecioná-lo no menu lateral de componentes e arrastá-lo para a página aberta. Para fins organizacionais, os componentes foram divididos em algumas seções, sendo elas “Formulário”, “Navegação” e “Visualização de dados”.

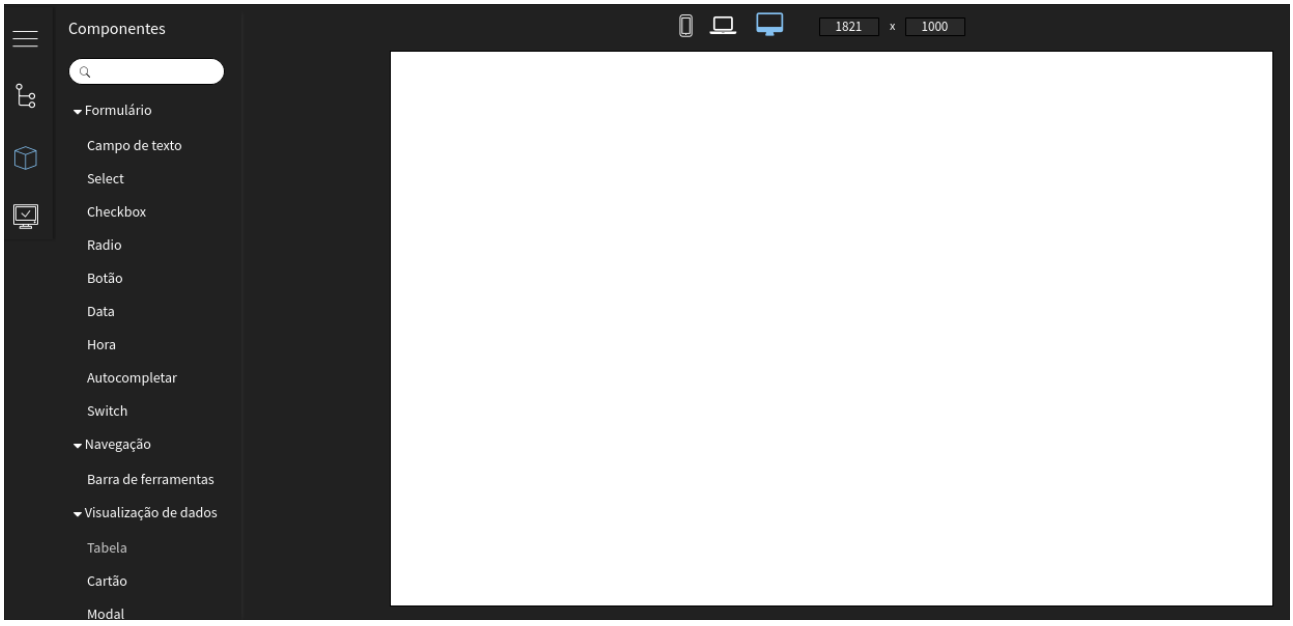

Figura 30. Tela onde são arrastados os Componentes.

Ao arrastar um componente para a página, surgem ao lado direito duas abas para personalização, sendo elas: aba de estilo, que engloba as alterações de arquivos CSS e a aba de configuração, na qual é possível preencher informações sobre o componente e adicionar eventos

(*click, change, focus, losefocus*). As abas também podem ser abertas em outra janela ou podem mudar de posição na página, de acordo com a escolha do usuário como é possível observar, respectivamente, nas Figuras 31 e 32.

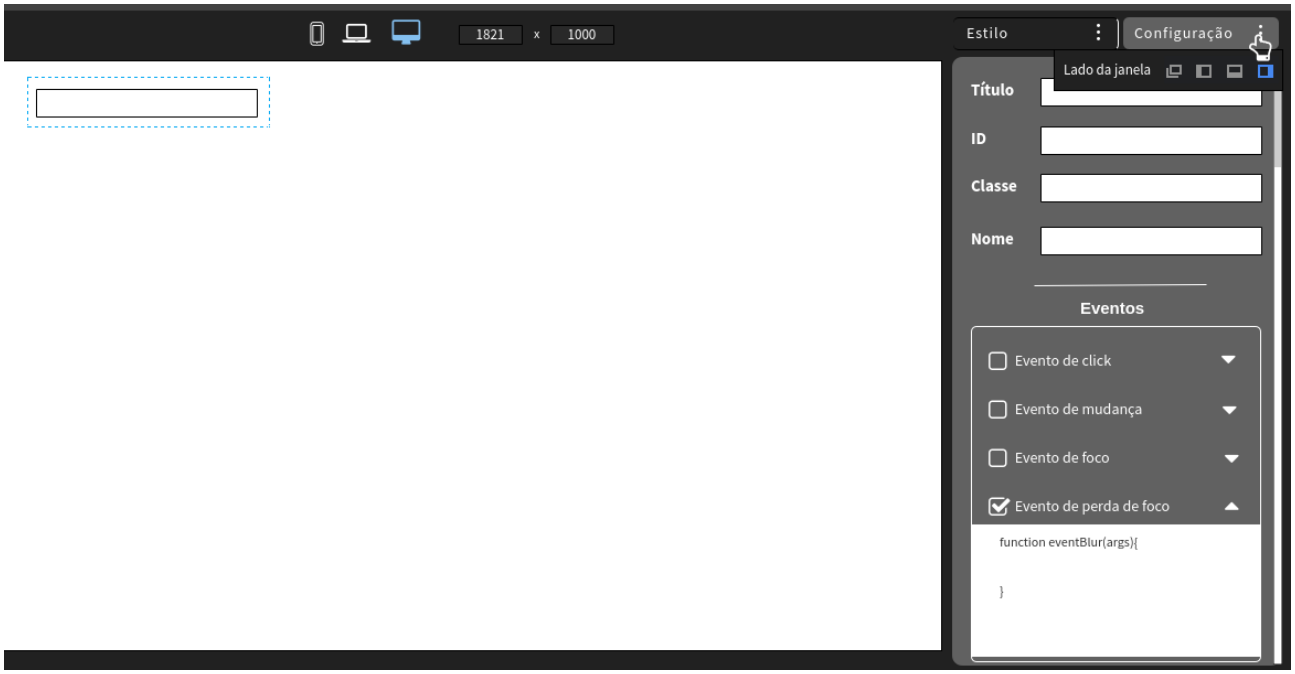

Figura 31 . Tela de configuração de componente com opção de movimentação da janela.

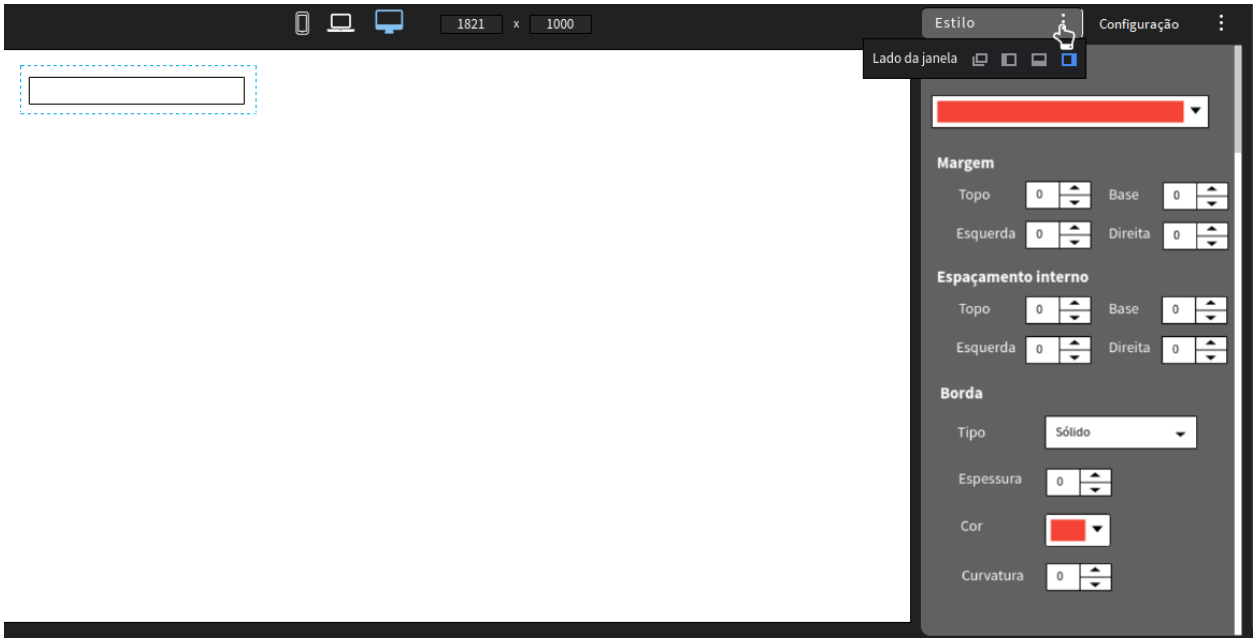

Figura 32. Tela de estilização de componente com opção de movimentação da janela.

O componente “Tabela” possui um formato diferente devido às suas características, como quantidade de linhas e colunas, possibilitando que algumas edições sejam realizadas diretamente

no componente. Isso é mostrado no esboço da Figura 33. As características comuns aos outros componentes são mantidas para a tabela.

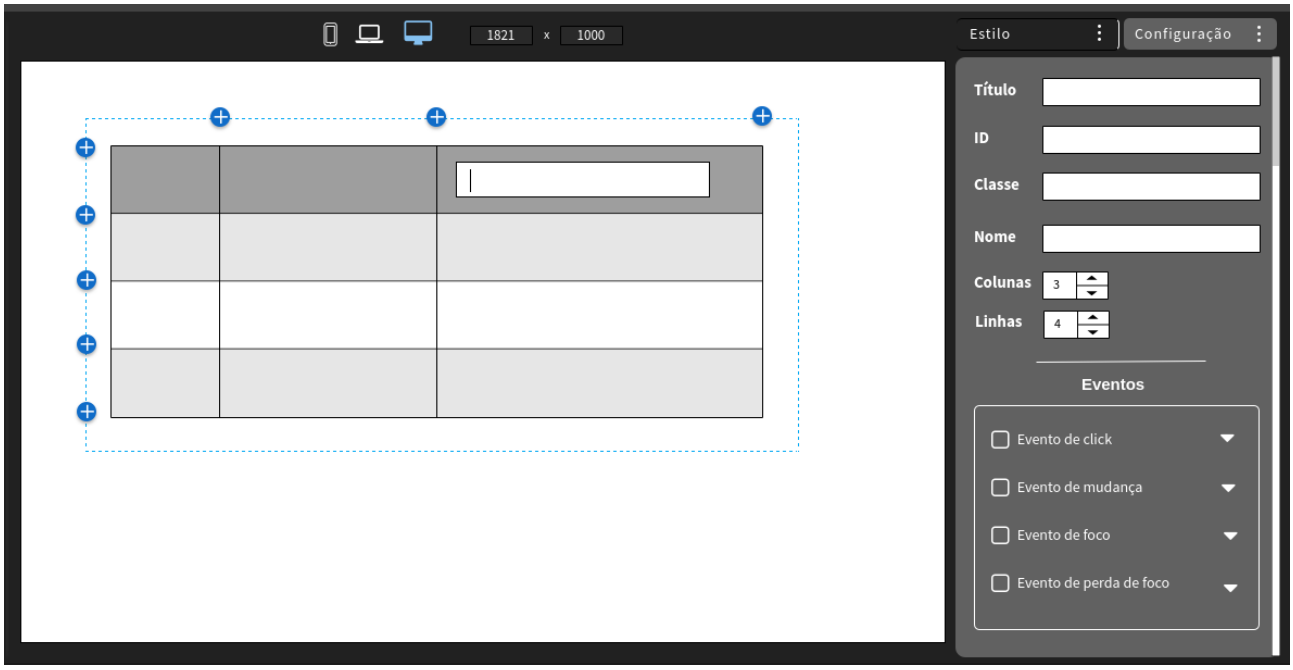

Figura 33. Tela de configuração do componente Tabela.

No menu “Guia” é possível selecionar qual diretriz o usuário deseja inspecionar em seu projeto. Ao realizar a inspeção, é exibido um menu lateral à direita com duas caixas com listagem. A primeira caixa cita as leis infringidas encontradas durante a verificação. Cada princípio/lei infringida é acompanhado da diretriz que foi violada, detalhando o que foi verificado. Por exemplo, se o princípio de “Boa continuidade” foi violado por elementos desalinhados horizontalmente em uma barra de navegação, é neste quadro que é exibido esse feedback. Na seção 6, é descrito com detalhes quais verificações são realizadas pela aplicação.

A segunda caixa propõe as respectivas sugestões de correções para as leis infringidas, citando o que e onde pode ser realizada a correção de acordo com os feedbacks informados anteriormente, como mostrado no esboço da Figura 34. Na seção 6 é especificado em detalhes como cada diretriz foi desenvolvida para identificar os princípios infringidos.

Como o feedback de acessibilidade é realizado por uma API externa, seu comportamento é diferente das demais verificações implementadas. Portanto, o feedback e as sugestões informadas por essa API estão disponíveis no *console* do *developer tools*, dentro do VSCode. Para acessá-lo, basta pressionar as teclas “ctrl + shift + i”.

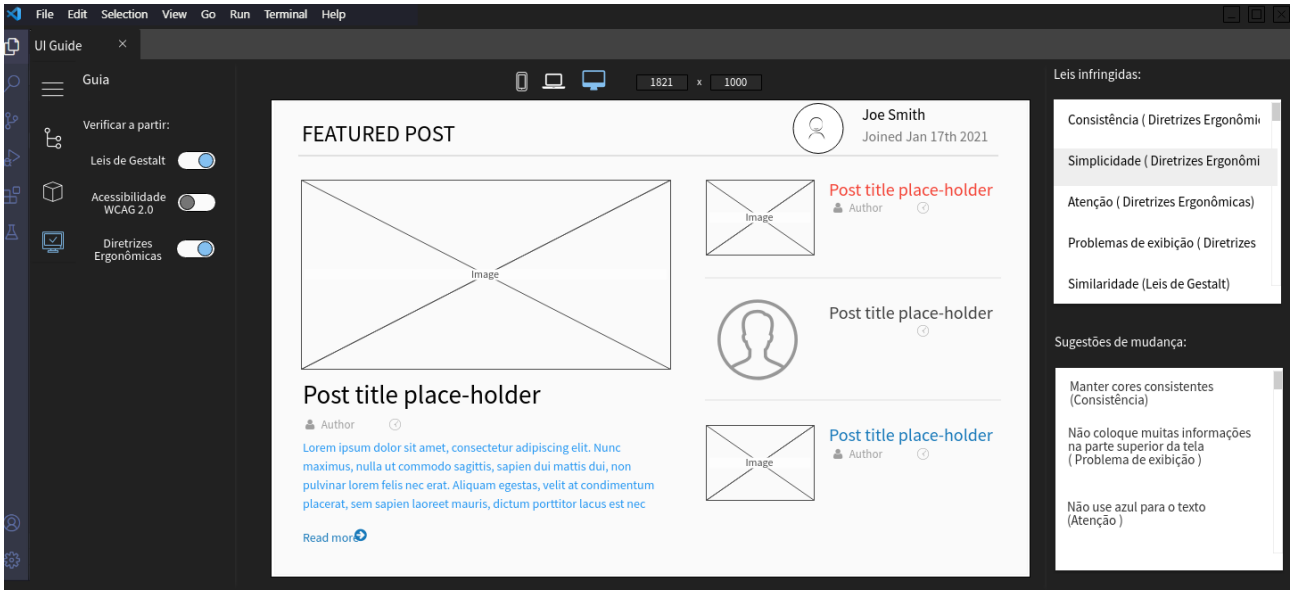

Figura 34. Tela de guia de usabilidade.

5. Glossário e Modelos de Dados

Neste tópico é apresentado o glossário do sistema, contemplado na Tabela 19. Nele, estão os atributos que podem ser visualizados pelo usuário e a sua devida descrição. Os valores booleanos marcados com o prefixo "[Flag]" representam que o campo servirá como alavanca para outro valor, isto é, permitirá que outro atributo seja alterado.

Atributo	**Formato**	**Descrição**
Página	Interface	Local onde os componentes podem ser manipulados, visualizadas todas as alterações dos componentes adicionados e renderizados arquivos abertos.
Nome (página)	*String*	Nome que a página criada terá.
Arquivo (importar)	Arquivo	O usuário adicionar um arquivo da sua máquina que está sendo utilizada no projeto que está sendo desenvolvido
Arquivo (exportar)	Arquivo	O usuário exportar um arquivo gerado pela aplicação após montar sua(s) página(s).
Componente	Interface	Atributos que compõem uma página. Podem ser os componentes pré-definidos pela aplicação, ou podem ser criados e customizados pelo usuário.
Cor (componente)	*String*	Cor de um componente ou de uma página.
Texto (conteúdo)	*String*	Conteúdo de um determinado componente em formato de texto.

Imagem (conteúdo)	Arquivo	Conteúdo de um determinado componente em formato de imagem, podendo ser nos formatos jpg, jpeg, png, svg, tiff, bmp, gif e webp.
[Flag] Leis de Gestalt	Booleano	Se verdadeiro, ativa o feedback de leis de Gestalt.
[Flag] Diretrizes Ergonômicas	Booleano	Se verdadeiro, ativa o feedback acerca das diretrizes ergonômicas.
[Flag] Acessibilidade	Booleano	Se verdadeiro, ativa o feedback acerca das diretrizes de acessibilidade.
Feedback (Gestalt)	Arquivo	Arquivo no qual serão gerados os feedbacks acerca das leis de Gestalt infringidas pelo usuário. Este arquivo pode ter extensão txt/JSON.
Feedback (Ergonomia)	Arquivo	Arquivo no qual serão gerados os feedbacks acerca das diretrizes ergonômicas infringidas pelo usuário. Este arquivo pode ter extensão txt/JSON.
Sugestões (Gestalt)	Arquivo	Arquivo no qual serão fornecidas sugestões de correção para o usuário acerca das leis de Gestalt infringidas. Este arquivo pode ter extensão txt/JSON
Sugestões (Ergonomia)	Arquivo	Arquivo no qual serão fornecidas sugestões de correção para o usuário acerca das diretrizes ergonômicas infringidas. Este arquivo pode ter extensão txt/JSON
Margem	Decimal	Tamanho da margem que um componente/página terá. O valor informado será decimal e será convertido para pixels.
Espaçamento interno	Decimal	Tamanho do espaçamento interno (*padding*) que um componente/página terá. O valor informado será decimal e será convertido posteriormente para pixels.
Título	*String*	Título que um componente em específico terá. É o texto que irá aparecer quando o cursor do mouse estiver sobre o componente.
Id	*String*	Id que um componente em específico terá. O Id deve ser único.
Classe	*String*	Classe que um componente em específico terá. Vários componentes podem ter a mesma classe.
Nome	*String*	Nome que determinado componente terá. Usado majoritariamente em componentes do tipo *input*.
Tipo (*input*)	*String*	Tipo de um componente *input*. Exemplos: *submit, text, number...*
Aria	*String*	Descrição de um campo *Aria*.
Linhas	Inteiro	Quantidade de linhas que uma tabela terá. Específico do componente tabela.
Colunas	Inteiro	Quantidade de colunas que uma tabela terá. Específico do componente tabela.
Folha de estilo	Arquivo	Arquivo CSS no qual serão registradas as formatações de estilo.

Script	Arquivo	Arquivo JS no qual serão registradas funções/eventos relativos ao comportamento de páginas/componentes.
Função	*String*	Função JS respectiva a uma página/componente que será salvada dentro de um Script.
Evento	*String*	Evento JS respectivo a uma página/componente que será salvo dentro de um Script.
Tipo (Evento)	Lista	Lista dos eventos que podem ser utilizados para manipular páginas/componentes. Exemplos: *click, blur, focus*...
Cor (borda)	*String*	Cor que a borda de uma página/componente terá.
Espessura (borda)	Decimal	Espessura que a borda de uma página/componente terá.
Tipo (borda)	String	Tipo que a borda de uma página/componente terá.
Curvatura (borda)	Decimal	Raio que a curvatura da borda de uma página/componente terá.
Altura (resolução)	Inteiro	Altura que o editor WYSIWYG terá. Valor pode ser editado a qualquer momento.
Largura (resolução)	Inteiro	Largura que o editor WYSIWYG terá. Valor pode ser editado a qualquer momento.
Feedback - Consistência (Ergonomia)	*String*	Avalia se todos os elementos da página estão consistentes entre si.
Feedback - Simplicidade (Ergonomia)	*String*	Avalia se o conteúdo da página está distribuído de uma forma complexa.
Feedback - Atenção (Ergonomia)	*String*	Avalia se o conteúdo da página não utiliza de forma exagerada elementos para chamar a atenção do usuário. (Ex.: Componentes de texto que aparecem mais de quatro vezes em tamanhos diferentes, ou em cores diferentes)
Feedback - Problemas de Exibição (Ergonomia)	*String*	Avalia se o *layout* da página está equilibrado.
Feedback - Similaridade (Gestalt)	*String*	Avalia se elementos de um mesmo grupo da página possuem estilos diferentes.
Sugestão: “Manter cores consistentes” (Consistência -Ergonomia)	*String*	Sugere manter cores iguais em funcionalidades semelhantes.
Sugestão: “Não coloque muitas	*String*	Sugere manter um *layout* equilibrado, evitando manter muitas informações em apenas um quadrante da página.

informações na parte superior da página" (Problemas de Exibição -Ergonomia)		
Sugestão: "Não use azul para o texto" (Atenção-Ergonomia)	*String*	Sugere não utilizar a cor azul em textos, por não ser uma cor que facilite a leitura.

Tabela 19. Glossário do sistema.

5.1 Modelo de Dados

O sistema não fará uso de banco de dados, visto que suas estruturas (páginas, componentes, folhas de estilo...) serão salvas em forma de arquivos na máquina do usuário. Ao analisar essa estruturação, no entanto, decidiu-se representar a maneira que os conjuntos de dados se relacionam por meio de um Diagrama Entidade Relacionamento simplificado, na Figura 35, para expressar suas relações e cardinalidade.

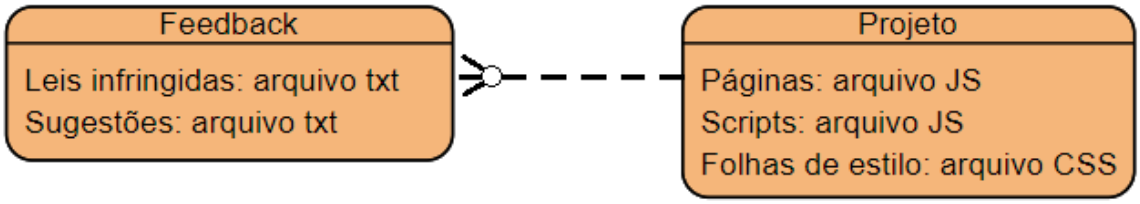

Figura 35. Representação de dados por meio de DER.

O "Projeto" será armazenado dentro de uma pasta na máquina do usuário, à sua escolha. Dentro dele, haverão as "Páginas" criadas, além de scripts e folhas de estilo, que serão gerados conforme edição do usuário. Na Figura 10, percebe-se que uma página se relaciona com componentes, e isso realmente acontece em tempo de implementação.

Cada projeto pode ter vários arquivos de "Feedback". O "Feedback" se encaixa nas partes acerca dos princípios de Gestalt e diretrizes ergonômicas implementados no projeto. Na Figura 36 há um exemplo de como será a estruturação dos arquivos.

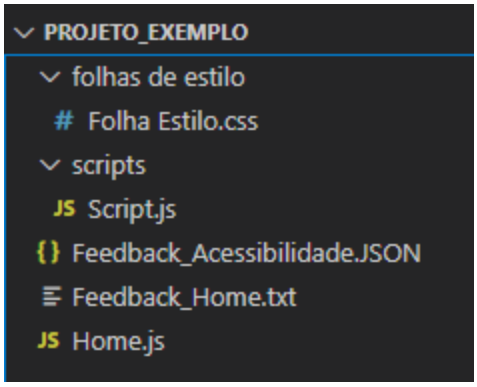

Figura 36. Exemplo de estruturação de arquivos.

6. Estratégia de Verificação de Acessibilidade, Ergonomia e Gestalt

Como cerne do trabalho, há a necessidade de maior detalhamento sobre as tarefas relacionadas à feedbacks e sugestões sobre princípios de gestalt, diretrizes ergonômicas e de acessibilidade. É importante enfatizar que o sistema não conseguirá abordar todos os aspectos dessas diretrizes e princípios, tanto por motivos de complexidade e tempo, tanto por alguns quesitos serem extremamente visuais (ou seja, é necessária uma análise visual humana para validar o quesito). Portanto, registra-se nesta seção quais aspectos serão englobados e a qual grupo de diretrizes eles pertencem.

6.1. Diretrizes de Acessibilidade

Anteriormente, decidiu-se utilizar a ferramenta Lighthouse para realizar a análise de acessibilidade. No entanto, não é possível utilizar essa ferramenta para fazer análise de arquivos locais, sendo necessário que a página a ser analisada esteja sob o protocolo HTTP ou HTTPS. Com base nesse obstáculo encontrado e ainda apoiando-se nas diretrizes WCAG 2.0, outra ferramenta foi escolhida, a a11y-checker. Utilizando análise feita com esta ferramenta em páginas web e com apoio da documentação disponível da ferramenta, segue as diretrizes de acessibilidade que são validadas.

1. **Títulos de páginas**: páginas *web* devem possuir títulos que descrevam seu tópico ou propósito;
   Idioma do documento: a língua humana padrão de cada página *web* pode ser programaticamente determinada;
2. **Codificação do documento**: Declarar codificação de caracteres;
3. **Escalação do documento**: redimensionamento de texto;
4. **Imagens**: todas as imagens devem possuir um atributo alt;
5. **Links**: links devem possuir contexto;
6. **Botões**: botões devem possuir texto descritivo;

7. **Formulários**: formulários devem haver rótulos descritivos;
8. **Iframes**: iframes devem possuir títulos descritivos;
9. **Áudios e vídeos**: deve ser provida uma alternativa para mídia baseada em tempo e conteúdo apenas em formato de áudio;
10. **Valor positivo para tabindex**: deve haver ordem de foco;
11. **IDs únicos**: os valores dos atributos de IDs devem ser únicos.

6.2. Diretrizes Ergonômicas

Para desenvolver a avaliação das diretrizes ergonômicas foi utilizada como base a documentação da Cornell University Ergonomics Web, que lista as diretrizes ergonômicas para projetos de interface. Para a avaliação realizada pela extensão desenvolvida, implementamos pelo menos uma diretriz de cada princípio listado, com exceção dos princípios de Direcionamento Cognitivo e Antropomorfização por não identificarmos maneiras lógicas para desenvolver a validação. Não são verificadas todas as diretrizes citadas na documentação por serem em grande quantidade podendo impactar na qualidade da validação considerando o tempo disponível para sua implementação. As diretrizes implementadas foram selecionadas levando em consideração diretrizes que não necessitem ter um conhecimento da regra de negócio do projeto implementado, como por exemplo a diretriz “não mostre informações importantes na tela por breves períodos de tempo” do princípio “Limitações de memória humana”, com base nas informações que podem ser obtidas no projeto não é possível avaliar o que poderia ser uma informação importante do projeto. Na lista abaixo, é possível visualizar quais diretrizes são verificadas e qual é o procedimento de desenvolvimento.

1. **Consistência ("Princípio da menor surpresa"):** Para as diretrizes “Terminologia consistente entre telas” e “Cor consistente entre telas com funções semelhantes”, se existe mais de uma página criada no projeto, é verificado se entre elas existem componentes com atributos semelhantes, mas com conteúdo de texto diferente para a validação da terminologia e cor diferente para consistência de cores.
2. **Simplicidade**: Para a diretriz “Quebrar longas sequências em etapas separadas”, é verificado primeiramente se o componente não é um container que cobre todos os componentes da página, se não for é verificado a existência de componentes com mais de 7 (sete) elementos filhos. Esse número vem da Lei de Miller, que define que as pessoas processam o número 7 (podendo variar em até dois números) porções de informação de cada vez.
3. **Limitações de memória humana**: Tem como diretriz “Organizar campos de dados para corresponder às expectativas do usuário ou para organizar a entrada do usuário (por exemplo, autoformatação de números de telefone)”. É verificado se uma entrada de dados possui atributos ou label que sejam semelhantes a algumas das palavras chaves separadas para a validação. As palavras chaves estão relacionadas as tipagens disponíveis para a tag HTML *input,* e os atributos *pattern* e *required.* Como exemplos de palavras chaves considerando o tipo *date,* as palavras adicionadas foram “data, *date*, data inicial, data final, data de nascimento e data de aniversário”. Se forem semelhantes, é verificado se o atributo tipo da entrada de texto corresponde ao relacionado a chave ou se possui uma

máscara do valor preenchido. Como segunda diretriz analisada para este princípio "Minimize as cargas de memória de trabalho limitando a duração das sequências e a quantidade de informações - evite a mania de ícones!", é verificado se na página existem mais de 20 ícones (o número 20 foi definido de acordo com o *Checklist* para validação de ícones do William Horton), verificando a quantidade de *tags svg* e *i* presentes na página.

4. **Comentários**: Tem como diretriz "Fornecer feedback semântico apropriado - feedback que confirma a intenção de uma ação (por exemplo, destacando um item sendo escolhido de uma lista)". É verificado se componentes que têm eventos possuem alguma alteração no estilo ao ativar o evento, comparando a listagem de estilo sem especificação com a com *hover* ou *active*.
5. **Mensagens do sistema**: Possui como diretriz "Evite o uso de mensagens ameaçadoras ou alarmantes (por exemplo, erro fatal, execução abortada, *kill job*, erro catastrófico)". São percorridos todos os componentes do projeto e comparado se o texto do conteúdo possui alguma das palavras chaves separadas para essa validação, como os citados na diretriz.
6. **Modalidade**: Para a diretriz "Permitir rotas de fuga das operações", é verificado se existem componentes com atributos e nomenclatura de ação como "próximo" ou "salvar", e se para cada um deles existem componentes de ação para sair dessa ação, como "cancelar" ou "voltar".
7. **Atenção**: Para a diretriz "Não use mais de 4 tamanhos de fonte diferentes por tela", são percorridos todos os componentes do projeto que possuam texto e salva em um *array* o tamanho da fonte sem que adicione tamanhos repetidos. Ao final, é verificado se o *array* ultrapassou o tamanho de 4 elementos.
8. **Problemas de exibição**: Tem como diretriz "Use bastante 'espaço em branco' ao redor dos blocos de texto - use pelo menos 50% de espaço em branco para telas de texto". São percorridos os componentes que possuam texto com mais de 96 caracteres, e verifica se possuem *padding* ou *margin* maiores que 10%.
9. **Diferenças individuais**: Tem como diretriz "Permitir formas alternativas para comandos (por exemplo, combinações de teclas por meio de seleções de menu)". São verificados se componentes que possuem atributos de evento preenchido, possui mais de uma chamada para a mesma função.

6.3. Princípios de Gestalt

Para o desenvolvimento de verificação dos princípios de gestalt, são considerados os princípios mencionados no livro "Interação Humano-Computador", de autoria de Simone Diniz Junqueira Barbosa e Bruno Santana da Silva. Ainda que estes princípios dependam amplamente de validação visual de um ser humano, é possível validar alguns de seus aspectos de forma lógica. As verificações destes princípios passam por todos os elementos de uma página, começando pela tag <body> e percorrendo todos os seus elementos filhos. Portanto, cabe ao usuário avaliar o feedback informado pela ferramenta. A seguir há informações sobre quais aspectos são utilizados na validação dos princípios.

1. **Boa continuidade:** é verificada no que tange ao alinhamento de elementos. Em barras de navegação e abas, é verificado se seus links (*anchors*) de navegação estão na mesma coordenada X, pela obtenção da propriedade *bottom* de sua posição relativa na página. Se

o valor dessa propriedade for diferente para qualquer um dos links, é informado para o usuário que houve infração do princípio de boa continuidade.

2. **Simetria**: é verificada no que tange ao equilíbrio dos elementos em uma página. Ou seja, é comparada a quantidade de elementos que estão no topo e no final de uma página, assim como os que estão do lado esquerdo e direito. Isso é feito ao obter a posição central da página, dividindo sua largura por dois, e calculando os elementos que estão de cada lado por sua posição relativa na página. Para calcular os elementos que estão do lado esquerdo, por exemplo, é usada sua posição *left* e a largura do próprio elemento, além da largura da página. Caso a maior parte deste elemento esteja do lado esquerdo da página, ele é adicionado em um *array* de elementos que estão à esquerda. Se o elemento estiver igualmente dividido entre esquerda e direita da página, ele é adicionado em ambos *arrays* de elementos à esquerda e à direita. Caso haja mais elementos em um dos *arrays* comparados, é possível afirmar que o conteúdo não está simétrico. São comparados entre si os *arrays* de elementos à esquerda e direita e, separadamente, os de topo e fundo.

7. Casos de Teste

Neste tópico, é explicado como os testes de aceitação e integração serão realizados conforme as funcionalidades desenvolvidas do projeto. Para ambos os casos de teste, são consideradas pré-condições, realizadas pelas funcionalidades; entradas, que são os comportamentos executados para se testar os cenários de teste e saída ou estado, que são os comportamentos ou resultados esperados à partir da entrada.

7.1 Testes de Aceitação

Nesta seção é apresentado um plano de teste de aceitação e um plano de teste de integração. O plano de teste de aceitação é apresentado na Tabela 20, e foi planejado a partir das necessidades descritas no documento de visão.

Necessidade	Pré-condição	Entrada	Saída/Estado
Automação/facilitação do processo de desenvolvimento front-end	Criar página	Arrastar componente botão	Renderizar botão na página
		Arrastar componente botão dentro de componente modal	Componente botão inserido no componente modal, e ambos os componentes renderizados na página

Customização de componentes	Criar componente	Inserir número de linhas ao editar componente tabela	Renderizar tabela com a quantidade de linhas digitadas
		Adicionar imagem da máquina do usuário no componente imagem	Renderizar imagem selecionada
		Importar folha de estilo	Renderizar componentes com o estilo do arquivo
Exportar scripts e outros *assets*	Criar página, script e folha de estilo	Exportar componente selecionado	Arquivo com código equivalente ao componentes selecionado
		Exportar script	Arquivo JS com código equivalente aos scripts da página
		Importar projeto dentro da extensão	Páginas com seus componentes disponíveis para serem arrastados, scripts e folhas de estilo disponíveis para serem editados dentro de cada componente e demais arquivos carregados (imagens, arquivos de texto)
Auxílio em criar *layout* amigável para o usuário	Ter uma página selecionada	Feedback sobre lei de gestalt simetria	Alerta dentro do editor WYSIWYG com a descrição de qual(is) componente(s) violaram a lei
		Sugestões de correção sobre a lei de gestalt agrupamento (quando	Alerta dentro do editor WYSIWYG com a descrição da correção a ser

		violada)	realizada
		Feedback sobre lei de gestalt boa continuidade	Alerta dentro do editor WYSIWYG com a descrição de qual(is) componente(s) violaram a lei
Facilidade ao adaptar o *layout* a telas.	Abrir a extensão	Redimensionar altura da tela	Editor WYSIWYG com altura redimensionada de acordo com o parâmetro informado
		Redimensionar largura da tela	Editor WYSIWYG com largura redimensionada de acordo com o parâmetro informado
		Informar tamanhos de tela baseados em displays de dispositivos (*smartphone, tablet, laptop*)	Editor WYSIWYG com altura e largura redimensionados de acordo com o dispositivo informado

Tabela 20. Testes de Aceitação.

7.2 Testes de Integração

Para os casos de teste de integração contemplados na Tabela 21, há comunicação com a ferramenta a11y-checker, devido ao seu uso para validação de diretrizes de acessibilidade; e o próprio contexto de extensão do VSCode, tendo em vista que esse contexto irá disponibilizar o editor WYSIWYG dentro do VSCode. Para cada umas dessas interfaces, será necessária uma abordagem diferente, baseada nos casos de teste tabela abaixo:

Interface/API	Pré-condição	Entrada	Saída/Estado
a11y-checker	Ter uma página selecionada	Ativar avaliação de acessibilidade	Itens de acessibilidade infringidos na página

			avaliada e sugestões de melhoria disponíveis no *developer tools*.
Contexto de extensão VSCode	Abrir a extensão	Permitir a renderização dos componentes em tempo de edição	Componente com posição, estilos e eventos aplicados assim que for realizada sua manipulação

Tabela 21. Testes de Integração.

Os testes de integração são realizados conforme seus pré-requisitos são desenvolvidos. Por exemplo, não é possível renderizar os componentes no editor WYSIWYG se os componentes não estiverem prontos. Portanto, é viável realizar os testes de integração na seguinte ordem: Contexto de extensão VSCode, visto que engloba a manipulação dos componentes quando estes estiverem prontos e a11y-checker, que pode ser testado após a validação acerca da manipulação dos componentes para criação de páginas.

8. Cronograma, Processo de Desenvolvimento e Artefatos de Implementação e Testes

Nesta seção é apresentado o processo realizado para a implementação do projeto. Com o intuito de organizar em tarefas as funcionalidades definidas no documento de visão, foram criadas *issues* no repositório do projeto para cada funcionalidade do sistema. Com exceção das funcionalidades de componentes que foram agrupadas em duas *issues* com a lista dos componentes definidos para criação e edição, foram divididos em dois grupos para facilitar a divisão de tarefas entre as integrantes do grupo. Considerando que cada grupo de componentes foi designado a uma integrante que ficou responsável pelo desenvolvimento de todas as tarefas relacionadas àqueles componentes listados. As outras tarefas foram divididas de acordo com as entregas, de maneira que cada uma fique com a mesma quantidade ou com uma diferença mínima de tarefas desenvolvidas a cada versão.

Considerando o formato de desenvolvimento ágil, o projeto foi dividido em cinco entregas incrementais, que podem ser visualizadas com os *milestones* criados no repositório do projeto. Cada *milestone* possui um agrupamento de *issues* que irão gerar uma nova versão funcional do sistema. Ao final de cada *milestone* é gerado um vídeo demonstrando todas as funcionalidades da nova versão do sistema. Além disso, o link para o vídeo é adicionado nos comentários da *issue* criada e também ao final desta seção.

As tarefas, conforme nomeadas na aba de *issues* do repositório do projeto, foram divididas em 4 entregas como mostrado na Figura 37.

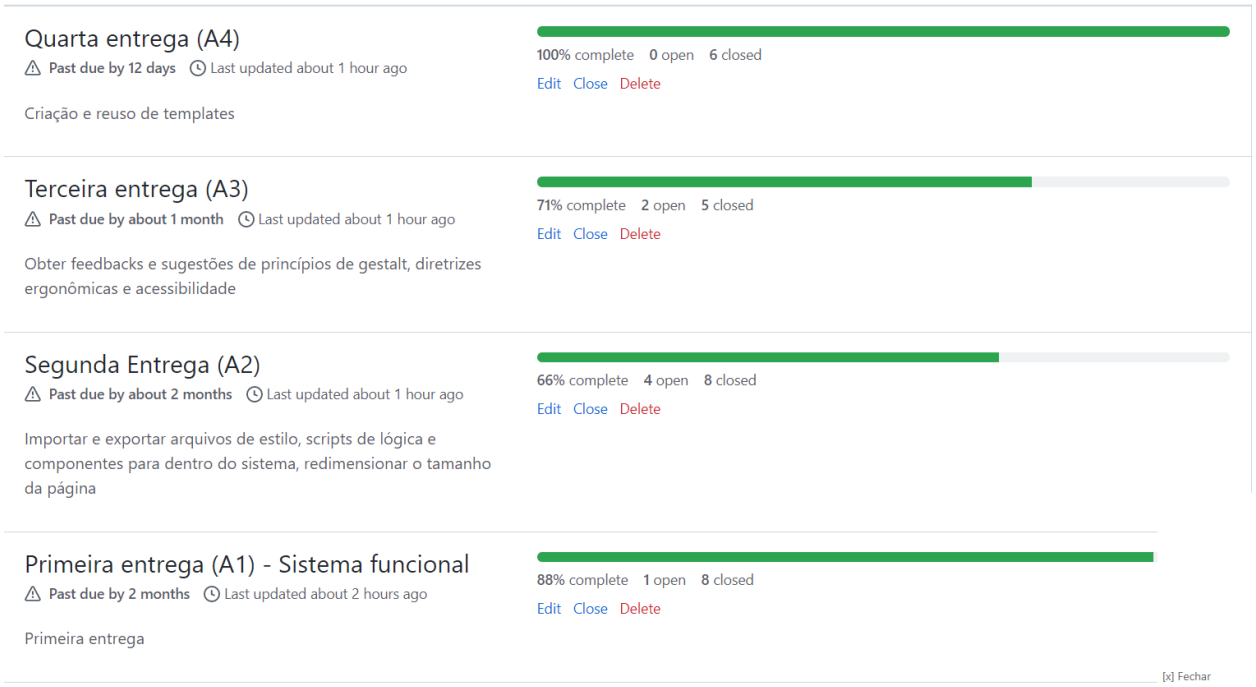

Figura 37. *Milestones* definidos para entrega do projeto.

Conforme as tarefas são finalizadas, a integrante responsável pela tarefa faz um comentário dentro da *issue* relacionada com algum artefato (imagem, descrição, documento, dentre outros) que comprove a realização desta. Às segundas-feiras, durante o segundo semestre de 2021, dentro do prazo vigente deste projeto, são realizados encontros com o orientador do projeto com a finalidade de acompanhar a ocorrência das atividades. Após averiguação de que as tarefas foram cumpridas, é responsabilidade do orientador fechar as *issues* correspondentes às tarefas no repositório do projeto. Durante os encontros semanais, são escritas atas para registrar os combinados, e estas são persistidas em formas de *issues* no repositório do projeto, com a *label* "AtaDeReuniao".

Para a primeira entrega, foram definidas todas as funcionalidades base do sistema. Ao final dela, é esperado que o usuário consiga criar e editar páginas, bem como componentes. Para a segunda entrega é previsto que o usuário consiga exportar e importar arquivos no projeto, sendo eles arquivos script e componentes, além de poder redimensionar o tamanho da página. Na terceira entrega foram agrupadas as funcionalidades de feedbacks dos princípios de gestalt, diretrizes ergonômicas e acessibilidade. Na quarta e penúltima entrega, são feitas as funcionalidades de sugestões dos princípios de gestalt, diretrizes ergonômicas e acessibilidade. E por fim, na última entrega, são realizadas correções e melhorias de funcionalidades que tiveram algum defeito durante a etapa de testes. Ao final de cada uma das entregas é gravado um vídeo demonstrativo das funcionalidades esperadas para as respectivas entregas. Estes vídeos estão disponíveis nos seguintes links:

- Primeira entrega: Foi entregue a documentação do projeto, tendo em vista que as funcionalidades base ainda não estavam completas.
- Segunda entrega: https://youtu.be/0wm4TA1zFaE
- Terceira entrega: https://youtu.be/hxDhpUBhiNU

- Quarta entrega: https://youtu.be/n6FHl7dqJZ0
- Quinta entrega: https://youtu.be/QYIWEH4tGO8

Para a quarta entrega foram realizados os testes seguindo os casos de teste apresentados na Seção 7. Para os casos de teste de aceitação, a renderização da quantidade correta de linhas na tabela, o feedback dos princípios de Gestalt e o redimensionamento de tela não tiveram os resultados esperados, sendo necessárias pequenas correções logo em sequência dos testes. Já nos casos de testes de integração apresentados na Tabela 25, todos os casos de teste chegaram aos resultados esperados, não sendo necessárias alterações.

O software desenvolvido é *open source,* ou seja, o código fonte é disponibilizado para que qualquer usuário possa acessá-lo . Ele pode ser acessado pelo link a seguir:

https://github.com/ICEI-PUC-Minas-PPLES-TI/plf-es-2021-1-tcc1-5308100-ariel-barcelos-e-karolina-vaz

9. Recomendações de Evoluções

O software proposto e implementado traz uma solução útil para que usuários possam desenvolver o layout de um projeto, em conjunto de um guia para interfaces de usuário que verifica a acessibilidade, ergonomia e princípios de Gestalt. Além disso, o software oferece sugestões para essas verificações. Esses aspectos fornecem uma maior segurança para o usuário sobre acessibilidade em geral durante o desenvolvimento do projeto, além de agregar em aprendizado e economizar tempo no desenvolvimento com os componentes pré definidos na aplicação.

O sistema ainda possui espaço para ser evoluído, podendo incluir mais verificações no projeto, principalmente no que tange às verificações sobre princípios de Gestalt e ergonomia. Poderiam ser adicionadas formas para que os usuários compartilhassem seu projeto com outros usuários da extensão, por exemplo. Também seria viável criar mais componentes, como carrosel e *banner*, para aumentar o número de possibilidades e facilidades para os usuários ao criarem suas páginas.