5- Criação de uma Skin para o Sistema de Interface de uma Aplicação de eLearning

Simão Monteiro[PG47677], Júlio Alves[PG47390], and Nuno Silva[PG42645]

Universidade do Minho

Abstract. Em Engenharia Informática uma pele (skin) é, basicamente, uma apresentação gráfica de um produto de software, que é previamente definida para ser interpretada posteriormente por um sistema de interfaces gráfico (GUI). Neste trabalho pretende-se desenvolver uma skin específica para uma aplicação de eLearning, que permita ser configurada de acordo com um conjunto de requisitos específicos de um utilizador.

Keywords: Skin · Frontend · Framework · Vue · HTML · CSS · JS

1 Introdução

Em desenvolvimento Web, uma "skin" é um conjunto de regras de estilo que modificam a aparência de uma página web ou de uma aplicação. Essas regras são feitas para alterar elementos específicos e estéticos, como a cor de fundo, fonte da letra, tipo de *layout*, entre outros. No entanto, estas alterações não afetam a funcionalidade da página web ou da aplicação. As "skins" podem ser aplicadas em certos níveis de páginas, secções ou elementos específicos, de forma a permitir aos desenvolvedores ou utilizadores a criação de designs personalizados e adaptáveis.

A motivação por trás deste projeto é criar uma "skin" flexível e configurável, que possa atender aos requisitos específicos de um ou mais utilizadores. O objetivo final é desenvolver uma "skin" que possa ser facilmente adaptada às necessidades dos utilizadores, melhorando a sua experiência e a eficiência da aplicação.

Este projeto está inserido no domínio de desenvolvimento de sistemas de interfaces gráficos e aplicações educacionais.

2 Conceção e Implementação de Skins

2.1 O que é uma Skin?

Um sistema de "skins" é uma ferramenta essencial para fornecer uma experiência personalizada e atraente para os utilizadores. Uma "skin" é uma camada de apresentação gráfica que se sobrepõe ao conteúdo da aplicação, e permite aos utilizadores definir a aparência, consoante os seus gostos pessoais, levando a que

a utilização da aplicação ofereça uma maior satisfação aos mesmos.

Isto significa que, com um sistema de "skins", a aparência da aplicação pode ser alterada sem afetar o conteúdo ou a funcionalidade da aplicação.

2.2 Sistemas de Desenvolvimento de Skins

Nesta secção vai ser feita uma breve explicação e enumeração dos vários sistemas de desenvolvimento de skins usados nos dias de hoje. Atualmente existem vários sistemas de desenvolvimento de "skins" usados pelos desenvolvedores de *front-ends*:

- **Sistemas de gestão de temas**- Estas aplicações permitem criar e gerir vários temas(skins) para um website ou aplicação. Geralmente oferecem ferramentas prontas para personalizar um *layout*, alterar cores, alterar tamanhos e outros aspetos visuais da aplicação.
- ***Frameworks* de *front-end*** - Muitos *frameworks* de *front-end*, como o Bootstrap, Foundation ou o Vue, incluem suportes para a criação de "skins". Estes *frameworks* oferecem componentes reutilizáveis e recursos de *layout* que podem ser personalizados de forma a criar designs únicos.
- **Sistemas de gestão de conteúdo**- Existem aplicações já desenvolvidas que permitem a gestão de conteúdo, como o Wordpress, estas aplicações também permitem aos seus utilizadores instalarem e personalizarem "skins" para usarem nos seus websites.
- **Ferramentas de desenvolvimento**-De forma a ajudar no desenvolvimento destas "skins" existem também ferramentas criadas para este âmbito, como o Less, Sass, Stylus. A particularidade destas skins é que permitem aos desenvolvedores a criação de regras de estilo de forma mais eficiente.
- **Ferramentas de design**- Existem também algumas ferramentas de design, como o Adobe XD ou Figma, que permitem aos designers a criação de protótipos interativos. Estes protótipos interativos podem depois ser exportados como arquivos de estilo e usados por desenvolvedores de *front-end*.

2.3 *Frameworks*

Nesta secção vão ser apresentados e descritos algumns *frameworks* usados com frequência pelos desenvolvedores de *front-end* nos dias de hoje.

- **Vue.js** - é um framework JavaScript progressivo que permite criar aplicações web reativas e interfaces de usuário. Fornece uma sintaxe simples e intuitiva para trabalhar com componentes, e oferecem diversos recursos para se trabalhar com estilo, como *scoped styles*, e suporte a pré-processadores CSS como Sass, Less e Stylus.
- **Bootstrap** - É um dos frameworks de *front-end* mais populares, fornece uma variedade de componentes reutilizáveis e recursos de layout que podem ser personalizados para criar designs únicos.

- **Foundation** - Outro framework de *front-end* popular, oferece componentes reutilizáveis, recursos de layout e ferramentas de desenvolvimento para criar designs adaptáveis.
- **Material-UI** - É um framework de *front-end* baseado no Material Design, um conjunto de diretrizes criadas pelo Google para projetar aplicativos e websites modernos. Fornece recursos para personalizar a aparência dos componentes e adaptá-los a diferentes plataformas, como web, dispositivos móveis e desktops.
- **Tailwind CSS**- É um conjunto de ferramentas de estilo pré-construídas que permite que os desenvolvedores criem designs personalizados sem escrever muito código CSS. Foi construída de forma a que a construção destes estilos seja mais eficiente a nível de código.
- **Semantic UI** - Framework de *front-end* que fornece componentes reutilizáveis e recursos de layout, além de ferramentas para ajudar a construir interfaces de usuário intuitivas.
- **Bulma** - É um framework de *front-end* minimalista , *open-source*, baseado em flexbox, fornece uma variedade de componentes e recursos para ajudar os desenvolvedores a criar designs modernos e responsivos. Fornece uma série de classes pré-definidas que podem ser facilmente aplicadas aos elementos HTML para controlar o layout e estilo da página.

3 O sistema de skins desenvolvido

3.1 Utilização do sistema de skin

Na imagem seguinte é possível observar o configurador deenvolvido que permite ao utilizador personalizar a sua própria skin.

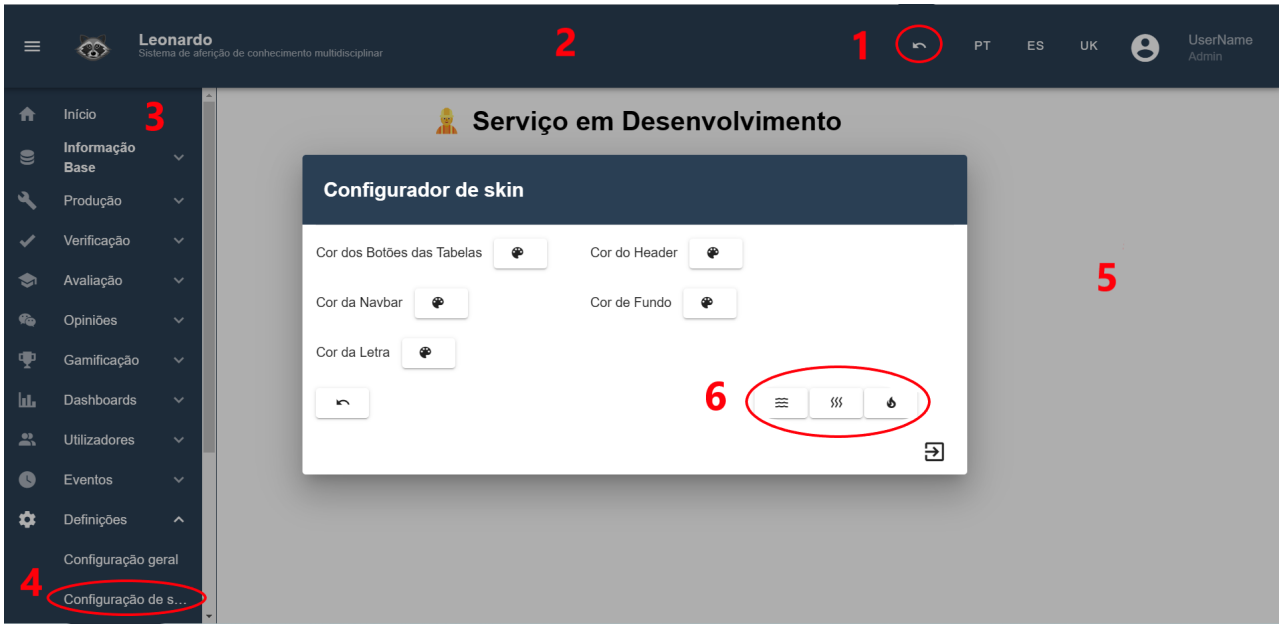

Fig. 1. Configurador

Como se pode ver na figura, existem 5 elementos diferentes que podem ser personalizados.

O primeiro elemento que pode ser personalizado é a "Cor dos botões das tabelas" que como o próprio nome diz, permite ao utilizador definir uma cor para todos os botões presentes nas tabelas (p.e. Utilizadores-Gestão). Outros elementos que também é possível personalizar é a cor do Header, da Navbar e do fundo, representados na imagem por 2,3 e 5 respetivamente. Ainda é possível também o utilizador escolher uma cor para as letras.

Os 3 botões presentes no canto inferior direito do configurador, representados na imagem por 6, servem para aplicar "skins" predefinidas.

Para prevenir a situação em que o utilizador escolhe uma cor de letra e de fundo igual, ficando assim sem conseguir ver as letras, foi criado o botão representado por 1 que aplica a skin default.

Para aceder a este configurador, basta ir à Navbar a Definições -> Configurador de skin, como se pode ver na imagem em 4.

3.2 Diagrama de relação de componentes

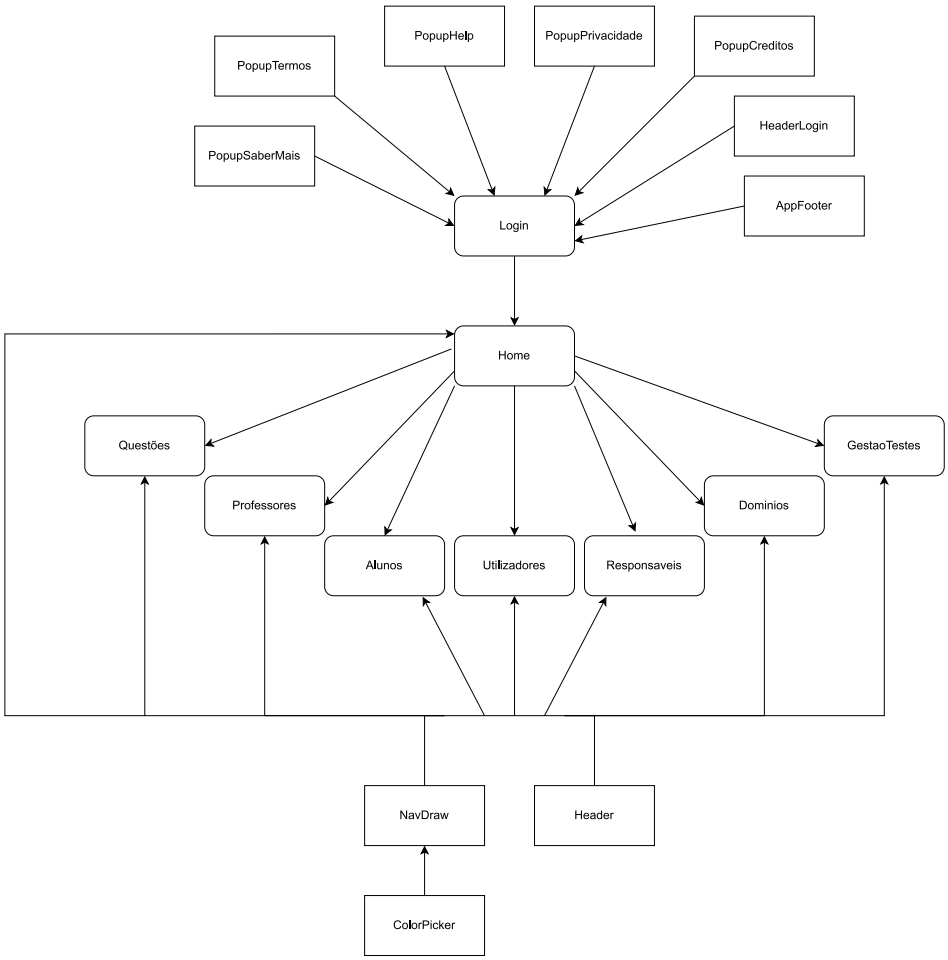

Fig. 2. Árvore de relação

No diagrama 2 pode-se visualizar a uma árvore de relação entre as vistas (representadas pelos retângulos com os cantos arredondados) e os componentes utilizados por cada uma (representados pelos retângulos com os cantos retos).

Como é vísivel, a vista do *Login* utiliza os componentes PopupSaberMais, PopupTermos, PopupHelp, PopupPrivacidade, PopupCreditos, HeaderLogin e AppFooter.

Num cenário de utilização real, após o utilizador fazer o login, a vista que lhe é apresentada é a vista do Home, daí a ligação entre as vistas do Login e Home.

A vista do Home dá acesso às vistas das Questões, Professores, Alunos, Utilizadores, Responsáveis, Domínios e GestaoTestes. Todas estas vistas, incluindo a vista da Home, utilizam os componentes Header e NavDraw, sendo que esta última também utiliza o componente ColorPicker.

4 Aspetos de Implementação

Este trabalho foi desenvolvido em Vue.js e está disponível no seguinte repositório github Leonardovue.

4.1 Vuex

Para se conseguir guardar as opções do utilizador no que toca à cor dos componentes, teve de ser usado esta tecnologia chamada Vuex.

O Vuex é um controlador de estados centralizados para aplicações Vue.js, oferecendo uma estrutura de armazenamento compartilhado para componentes Vue e facilitando a gestão do estado global da aplicação. Segue o padrão de fluxo de dados unidirecional, onde os dados só podem ser modificados através de "mutações" específicas e registadas em "logs", chamados de "mutation logs" para ajudar na depuração.

Também permite que os componentes despoletem "ações" para realizar operações assíncronas antes de efetuarem mudanças no estado.

4.2 Variáveis Sessão

Criaram-se então 5 variáveis de sessão, onde, em cada uma, será alocado o valor em código hexadecimal da cor pretendida. Cada variável está associada a um componente:

- ***topnavColor*** - Nesta variável, é guardado o valor da cor em "hex" correspondente ao *header*, isto é , à barra de cima da aplicação.
- ***bttColor*** - Nesta variável, é guardado o valor da cor em "hex" correspondente a alguns botões na aplicação, uma vez que nem todos os botões são personalizaveis. Os botões configuráveis são, somente, os botões associados às tabelas.
- ***navBarColor*** - Nesta variável, é guardado o valor da cor em "hex" correspondente à cor da barra lateral, onde são apresentados os diferentes menus das funcionalidades do Leonardo.
- ***backgroundColor*** -Nesta variável, é guardado o valor da cor em "hex" correspondente à cor do fundo.
- ***letterColor*** -Nesta variável, é guardado o valor da cor em "hex" correspondente à cor das letras. É importante referir que nem todas as letras são personalizaveis. Somente as letras consideradas título, e presentes no corpo das *views* oferecem a opção de personalização.

4.3 Modificação do valor das variáveis sessão

Para modificar o valor da variável sessão, usou-se a função *set* associada a cada variável. No entanto, para modificar as mesmas em tempo real, foi necessário usar uma opção específica do Vue.js. Esta opção tem o nome de *Watchers* e será explicada na secção seguinte. Sempre que o utilizador usa a palete de cores e modifica uma cor, é ativada uma função que guarda o valor da cor escolhida em "hex" na variável sessão.

Watchers

Watchers são mecanismos presentes no Vue.js que permitem a execução de uma determinada função sempre que uma propriedade específica dos dados estiver a ser observada. Estes mecanismos são úteis quando se pretende executar lógicas específicas sempre que uma determinada propriedade é alterada, sem a necessidade de recriar todo o componente. Os *Watchers* são adicionados a um componente através do método "watch", especificando a propriedade a ser observada e a função a ser executada sempre que a propriedade for alterada. É possível definir múltiplos *Watchers* para várias propriedades diferentes no mesmo componente.

4.4 Usar variáveis de sessão

Após modificar o valor dessas variáveis, era necessário usar as mesmas nos respetivos componentes para que se alterasse a cor. Esse valor era obtido através das funções *get* definidas anteriormente. Essas funções *get* e *set* estão contidas na opção *computed* que será explicada em seguida, permitindo ver aos utilizadores, os resultados da mudança de cor em tempo real.

4.5 *Computed*

As propriedades *Computed* são propriedades presentes no Vue.js que são automaticamente recalculadas sempre que uma ou mais das suas dependências sofrem modificações. São similares aos *Watchers*, mas em vez de se especificar uma função para ser executada quando uma propriedade é alterada, especifica-se uma função para ser executada. Após executada, esta função calcula o valor da propriedade *Computed*. As propriedades *Computed* são declaradas no objeto de opções do componente Vue, e as funções que as calculam são especificadas no objeto *Computed*. Como as propriedades *Computed* usam a cache,só serão executadas novamente se as suas dependências forem alteradas.

5 Conclusão e trabalho futuro

Em conclusão, o desenvolvimento do configurador de skins para uma aplicação eLearning foi um sucesso, permitindo aos utilizadores personalizar as cores de

6 componentes diferentes. Isso contribuiu para aumentar a satisfação dos utilizadores e melhorar a sua experiência de utilização. Além disso, a capacidade de personalizar a aparência da aplicação pode ajudar a aumentar a motivação dos utilizadores.

Como trabalho futuro, seria interessante expandir as opções de personalização, incluindo novos componentes e opções de design. Por exemplo, adicionar a opção de personalizar o layout da página, ou permitir aos utilizadores importar as suas próprias imagens como plano de fundo. Isso poderia ajudar a tornar a aplicação mais atraente e única para cada um.

Além disso, poderiam ser adicionadas opções de personalização baseadas em preferências de acessibilidade para atender a necessidades específicas de utilizadores com dificuldades visuais. Estas opções podem incluir fontes de maior tamanho, contraste elevado, e opções de leitor de tela. Isso poderia ajudar a garantir que a aplicação seja acessível para qualquer pessoa e possa ser usada com conforto e eficiência.

References

[1] Pictogrammers. *Material Design Icons*. Acessado em: 27 de Janeiro de 2023. URL: https://pictogrammers.com/library/mdi/.

[2] Thomas Sigdestad. *Front-end frameworks: What is important right now?* Acessado em: 27 de Janeiro de 2023. URL: https://enonic.com/blog/front-end-frameworks-what-is-important.

[3] shrey vijayvargiya. *Develop a Theme Management System*. Acessado em: 27 de Janeiro de 2023. URL: https://javascript.plainenglish.io/developing-theme-management-system-c55e5bb996a0.

[4] Vue.js. *Vue.js - The Progressive JavaScript Framework*. Acessado em: 27 de Janeiro de 2023. 2023. URL: https://vuejs.org/.

[5] Vuetify. *Vuetify - Material Design Component Framework*. Acessado em: 27 de Janeiro de 2023. 2023. URL: https://next.vuetifyjs.com/.

[6] *Vuex*. Acessado em: 27 de Janeiro de 2023. URL: https://vuex.vuejs.org/.