

1. Contexto e Objetivo

Precisamos criar um componente de **Guidebar** (barra de navegação auxiliar ou barra de CTA) que fique fixada (sticky) no rodapé ou topo da página. O objetivo é aumentar a conversão para "Agendar Test Drive" ou "Solicitar Cotação" enquanto o usuário navega pelos detalhes dos veículos.

Este componente deve ser editável pelo autor (dialog) e deve permitir configurações de cores e links.

2. Requisitos Funcionais (Authoring)

- **Texto do CTA:** O autor deve poder alterar o texto (ex: "Garanta seu Velocity agora").
- **Link de Destino:** Campo para selecionar a página interna ou link externo.
- **Posição:** Opção para fixar no Topo ou no Rodapé (Dropdown).
- **Visibilidade:** Checkbox para exibir/ocultar a barra sem precisar remover o componente.
- **Ícone:** Opção de selecionar um ícone simples (opcional).

3. Requisitos Técnicos

- **Resource Type:** `practice/components/guidebar`
 - **Group:** `Practice - Structure` ou `Practice - Content`
 - **Category:** Deve ser responsivo (Mobile/Desktop).
 - **Performance:** Deve usar CSS puro para o efeito `sticky` sempre que possível, evitando JavaScript pesado de scroll listeners.
-

4. Proposta de Arquitetura Técnica

4.1 Estrutura do JCR (Node Structure)

A estrutura seguirá o padrão Core Components Proxy.

- `guidebar/`
 - `.content.xml` (Definição do componente)
 - `_cq_dialog/` (Definição dos campos de autoria)
 - `guidebar.html` (HTL Render script)
 - `clientlibs/` (CSS/JS isolado)

4.2 Definição do Diálogo (Dialog XML)

Precisamos de um Multifield? **Não**. Como é uma barra única, usaremos campos simples dentro de um container.

Campos Propostos:

1. `./text` (Textfield): Texto principal.
2. `./linkURL` (Pathfield): Link do botão.
3. `./btnLabel` (Textfield): Texto do botão.
4. `./position` (Select): Valores `top` ou `bottom`.
5. `./variant` (Select): `primary` (Escuro) ou `secondary` (Claro/Cyan).

4.3 Sling Model (Java)

Criação da classe `GuidebarModel.java`.

Interface:

Java

```
None
public interface Guidebar {
    String getText();
    String getLinkURL();
    String getBtnLabel();
    String getPositionClass(); // Retorna "guidebar--fixed-top"
    ou "guidebar--fixed-bottom"
    String getThemeClass();    // Retorna a classe de cor
    boolean isEmpty();
}
```

5. Prova de Conceito (PoC) - Implementação Rápida

5.1 HTL (`guidebar.html`)

Esboço da renderização HTML. Note o uso de `data-sly-test` para não renderizar uma barra vazia.

HTML

None

```
<sly
data-sly-use.guidebar="com.adobe.aem.tutorial.core.models.Guideba
rModel"/>
<sly
data-sly-use.templates="core/wcm/components/commons/v1/templates.
html"/>

<sly data-sly-call="${templates.placeholder @
isEmpty=guidebar.empty}"></sly>

<sly data-sly-test="${!guidebar.empty}">
    <div class="cmp-guidebar ${guidebar.positionClass}
${guidebar.themeClass}">
        <div class="cmp-guidebar__container">
            <span
class="cmp-guidebar__text">${guidebar.text}</span>

            <a class="cmp-guidebar__action"
href="${guidebar.linkURL}.html">
                ${guidebar.btnLabel}
            </a>
        </div>
    </div>
</sly>
```

5.2 CSS (Sticky Behavior)

O segredo do Guidebar está no CSS.

CSS

None

```
.cmp-guidebar {
    width: 100%;
    background-color: #0f1111; /* Cor padrão A4X */
    color: white;
```

```
padding: 15px 0;
z-index: 9999; /* Garante que fique acima do carousel */
box-shadow: 0 -2px 10px rgba(0,0,0,0.3);
}

/* Variação: Fixado no Topo */
.guidebar--fixed-top {
    position: sticky;
    top: 0;
}

/* Variação: Fixado no Rodapé */
.guidebar--fixed-bottom {
    position: fixed;
    bottom: 0;
    left: 0;
}

/* Layout Interno */
.cmp-guidebar__container {
    max-width: 1200px;
    margin: 0 auto;
    display: flex;
    justify-content: space-between;
    align-items: center;
    padding: 0 20px;
}
```