

Intercept Brasil Clone: Guia Técnico de Desenvolvimento

Visão Geral do Projeto

O objetivo deste projeto é recriar **visual e funcionalmente** o site do Intercept Brasil (<https://www.intercept.com.br/>) usando uma stack moderna (por exemplo, Next.js 13 com Tailwind CSS e componentes do **shadcn/ui**). A réplica deve **seguir fielmente o layout, componentes e comportamento responsivo** do site original – entretanto, **nenhum código, imagem ou conteúdo protegido por direitos autorais pode ser reutilizado**. Isso significa que o desenvolvedor deve implementar tudo com código próprio e usar conteúdos alternativos (texto placeholder, imagens livres de copyright ou geradas) mantendo a aparência e usabilidade idênticas. Em suma, vamos replicar a *experiência do usuário* e o *design* do Intercept Brasil, mas com nossos próprios recursos.

Resumo das características do site original: Trata-se de um site de jornalismo investigativo, com diversas seções editoriais. O design é limpo e focado em leitura, com ênfase em tipografia forte, fundo claro e destaques em texto e imagens. A navegação oferece categorias como política, direitos, meio ambiente etc., além de séries especiais, colunas de opinião (“Vozes”) e vídeos. O site é responsivo – adaptando menu, grade de notícias e tipografia conforme o tamanho da tela. Também há funcionalidades como busca, inscrição em newsletter (inclusive um *gate* que solicita cadastro após certo número de artigos lidos) e chamada para doações. A seguir detalhamos cada aspecto e, na sequência, apresentamos instruções passo a passo de implementação.

Arquitetura de Páginas

O site organiza seu conteúdo em várias páginas e seções principais, que precisam ser mapeadas e reconstruídas:

- **Homepage (Página inicial):** exibe as notícias mais recentes e destaques. No topo, há um ou dois **destaques principais** (notícias em evidência com imagem grande e título em destaque) ¹. Abaixo, segue uma lista cronológica de matérias recentes em formato de cartões contendo título, data, autor e um breve resumo ². Após listar um certo número de notícias, a homepage oferece um botão/link “Acesse mais matérias”, que leva ao arquivo completo de notícias. Essa página dá a visão geral do conteúdo novo do site.
- **Páginas de Categoria:** para seções editoriais como **Poder, Direitos, Meio Ambiente, Segurança, Tecnologia**, etc. Cada categoria possui uma página listando somente as matérias daquela seção, em ordem cronológica, geralmente com um título da categoria no topo ³. A listagem de cards é similar à da homepage. Por exemplo, a página da categoria “Poder” apresenta o título “Poder” seguido de uma lista de artigos relacionados ⁴. Itens que fazem parte de alguma série especial exibem o nome da série como uma tarja antes do título – por exemplo, “**Cartas Marcadas** Hugo Motta usa até seguranças...” aparece com a série *Cartas Marcadas* destacada ⁵. As páginas de categoria incluem paginação quando há muitas entradas (por exemplo, botões ou links numerados para acessar “página 2”, “página 3” etc. no fim da listagem) ⁶.

- **Página “Matérias” (Arquivo Geral):** um índice geral de todas as matérias publicadas, semelhante a uma categoria “todas”. No rodapé do site original há um link “Matérias” que leva a essa página ⁷. Deve ser estruturada como as páginas de categoria, listando todas as notícias sem filtro específico.
- **Especiais (Séries Especiais):** o Intercept Brasil produz séries especiais de reportagens sobre um tema específico. Há uma página índice de **Todos os Especiais**, listando cada série com nome e descrição ⁸. Cada série especial tem sua própria página dedicada, com um layout especial:
- **Página de Série:** exibe o título da série, possivelmente um selo ou tarja indicando se é “Especial” ou “Série”, seguido de uma imagem de capa ou ilustração temática e uma descrição introdutória da série ⁹. Por exemplo, a série “*Israel: estado genocida*” mostra a palavra “Especial” no topo, o título da série como um grande heading, e um texto descritivo contextualizando o tema ¹⁰. Abaixo disso, lista-se todas as matérias que fazem parte da série, em ordem cronológica (as mais recentes primeiro), similares a cards de notícia mas normalmente incluindo o nome da série antes do título (p. ex. “Ver mais **Israel: estado genocida** Trégua de Israel no Líbano...” etc.) ¹¹. Ou seja, dentro da página da série, não é necessário repetir o selo “Especial”, mas é comum que todos os títulos sejam precedidos pelo nome da série para reforço visual ¹¹. Essas páginas também podem ser paginadas se a série tiver muitas entradas.
- **Vozes (Colunas de Opinião):** há uma seção específica para colunistas. A página “**Vozes**” (também referida no menu como “Todas as vozes”) lista os colunistas principais e suas contribuições:
- No topo, apresenta uma lista dos **principais colunistas** com sua foto (no site original, há fotos dos autores) e nome, possivelmente com um destaque de sua última coluna. Na versão texto do site, isso aparece como itens do tipo “Ver mais [Nome do colunista] [título da última coluna] [data]” agrupados sob a rubrica “Colunistas” ¹². Ex.: “Ver mais Fabiana Moraes *Cristão, pai, marido, patriota. E assassino confesso* – 19 de agosto” ¹².
- Em seguida, há uma seção “**Mais Vozes**” listando as colunas recentes de todos os autores, em formato de lista similar às categorias (título da coluna, data e autor) ¹³. Muitas vezes, várias entradas sucessivas podem ser do mesmo autor (p.ex., João Filho aparece em múltiplas entradas seguidas), já que é uma lista cronológica de todas as colunas de opinião recentes ¹⁴.
- Assim, a página Vozes combina um destaque por autor (cada colunista em evidência com seu último artigo) e a listagem completa das colunas.
- **Vídeos:** há uma seção de vídeos com sua página própria. A **página “Vídeos”** exibe as produções em vídeo do Intercept. No topo, parece haver um destaque de vídeo principal (por exemplo, a matéria mais recente em vídeo) com formato diferenciado ¹⁵. No HTML extraído, notamos um primeiro item isolado seguido por uma linha divisória e então uma lista de vídeos ¹⁶ ¹⁷. Provavelmente, o layout exibe o vídeo mais recente em destaque (com player incorporado ou thumbnail maior) e abaixo uma lista estilo grade ou lista simples com outros vídeos. Os títulos de vídeos muitas vezes começam com a marca “VÍDEO:” no título (p. ex. *VÍDEO: Perseguido por vereador bolsonarista...*) ¹⁵, o que no front-end pode ser estilizado como um rótulo. A listagem de vídeos também inclui data e autor(es) e é paginada se necessário ¹⁸ ¹⁹.
- **Página de Post (Artigo individual):** é a página de leitura de uma matéria específica, com o conteúdo completo. Cada artigo segue um template com os seguintes elementos:

- **Cabeçalho do artigo:** pode incluir um **kicker** ou categoria do artigo em texto pequeno no topo. Por exemplo, no artigo de Hugo Motta, aparece “Censura oficial” acima do título ²⁰ – isso funciona como um subtítulo curto ou seção temática, destacando o contexto da notícia (nesse caso, indicando o assunto geral da matéria). Deve ser opcional no design (nem todos os artigos terão kicker).
- **Título** do artigo em destaque (HTML `<h1>`), geralmente bem visível ²⁰.
- Logo abaixo do título, há um **conjunto de ícones de compartilhamento** (WhatsApp, Facebook, Bluesky, etc.) e possivelmente um botão de apoio/doação. ²¹. Esses botões permitem compartilhar o artigo nas redes sociais; no site original eles aparecem como ícones clicáveis integrados ao cabeçalho do post (tanto no topo da página quanto eventualmente repetidos no fim, conforme o layout responsivo) ²¹. Precisamos implementar esses ícones com links de compartilhamento (ex.: abrir o link “Compartilhar no Twitter” com a URL do artigo, etc.) ou usar APIs de compartilhamento web.
- Em seguida pode vir uma imagem principal (capa da matéria) ou um vídeo incorporado, dependendo do artigo. No HTML extraído, o primeiro item após o título foi uma imagem com legenda relacionada ao artigo do Hugo Motta ²². Essa imagem de destaque deve ser apresentada em tamanho grande (full-width ou quase isso) para dar ênfase visual.
- **Créditos do autor:** normalmente aparecem logo no início do artigo, mostrando o nome do autor (ou autores) com link para perfil, e a data/hora de publicação ²³. No site original, vemos “Paulo Motoryn e Thalys Alcântara – 9 de set de 2025, 07h58” logo após o título e imagem ²³. Vamos posicionar isso adequadamente, provavelmente estilizado em itálico ou cor neutra, para contexto.
- Se a matéria fizer parte de uma **série especial ou newsletter**, isso é destacado no início do conteúdo. No exemplo, o artigo do Hugo Motta é parte da série *Cartas Marcadas*, então logo depois dos autores aparece um indicativo da série: “Cartas Marcadas – Parte 12” acompanhado de uma breve descrição da série e um link “Confira o especial completo” ²⁴. No front-end, podemos implementar isso como um *bloco de destaque* (talvez um componente) no início do artigo, com uma tarja ou box informando o nome da série, o número/parte e a descrição, junto a um botão/link para a página principal da série ²⁴.
- **Corpo do texto:** a matéria em si com parágrafos de texto, intertítulos e imagens. O texto é formatado para fácil leitura (fontes adequadas, tamanho confortável, espaçamento entre linhas e parágrafos). Pode haver subtítulos (subheadings) no meio do artigo para organizar o conteúdo (no HTML vimos, por exemplo, `### O cercadinho do silêncio` dentro do texto) ²⁵. Esses subtítulos devem ser estilizados de forma hierárquica (menor que o título principal, talvez em bold).
- **Imagens e legendas:** no meio do texto podem aparecer imagens ilustrativas ou infográficos. Devem ser centralizados, em tamanho adequado (talvez largura total em mobile e uma coluna centralizada em desktop). Cada imagem costuma vir acompanhada de uma legenda descritiva e/ou crédito. No exemplo, após o subtítulo “O cercadinho do silêncio”, há uma imagem (ou conjunto de imagens) seguida de um texto que serve como legenda explicativa sobre as ações de Hugo Motta, etc. ²⁶. É importante implementar um estilo para legendas (texto menor e talvez em itálico ou cor secundária) imediatamente abaixo das imagens.
- **Links externos e ênfases:** o texto pode conter hiperlinks (para fontes externas ou referências) embutidos. Estes devem ter estilo diferenciado (ex.: sublinhado ou cor diferente) e abrir em nova aba, já que são fontes externas. Também podem haver trechos em itálico ou aspas. Certifique-se de incluir o estilo para `<blockquote>` caso haja citações em blocos, e para itálico/strong.
- **Sessão final do artigo:** muitos artigos do Intercept terminam com seções padronizadas:
 - Um **bloco de chamada à ação** para doação/apoio: geralmente um texto enfatizando que o Intercept é financiado pelo público, seguido de um botão “Apoie o Intercept” ou

similar ²⁷ ²⁸ . Este bloco deve ter destaque (talvez um fundo levemente diferenciado ou borda) e um botão estilizado (ex: cor chamativa) para doação. *Obs:* O botão leva a uma página externa de doação, podemos linkar para um URL placeholder ou criar uma página fictícia de agradecimento.

- **Tags ou temas relacionados:** uma lista de categorias ou assuntos relacionados à matéria. Por exemplo, no fim do artigo do Hugo Motta aparece “TEMAS RELACIONADOS: Poder” ²⁹ – indicando que aquela matéria se relaciona ao tema “Poder” (categoria). Podemos implementar isso como links para as páginas de categoria correspondentes.
 - **Informações de contato do(s) autor(es):** o Intercept exhibe ao final de cada artigo um bloco “Entre em contato” com o nome do autor, suas redes (Twitter) e e-mail de contato ³⁰ . No site original isso aparece possivelmente com a foto do autor ou apenas texto. Na nossa implementação, podemos incluir foto se disponível (placeholder) e ícones/link para o Twitter/e-mail. Esse é um componente para **perfil do autor resumido** no rodapé do artigo.
 - **Outras matérias recentes ou da mesma série:** frequentemente há recomendações de leitura adicional. No exemplo, após o artigo aparecem títulos de outras edições da mesma série *Cartas Marcadas* ³¹ e também uma lista de “Artigos recentes” gerais ³² . Precisaremos decidir quais blocos incluir – talvez mostrar “Leia também” com alguns posts relacionados (podem ser fictícios ou repetir alguns últimos posts).
 - **Formulário de newsletter:** Importante, o Intercept Brasil implementou recentemente um *paywall invertido*, onde após ler um ou alguns artigos, o usuário é convidado a **inscrever-se na newsletter para continuar lendo** (mas sem cobrar pagamento). No HTML coletado, vemos um prompt inserido no final do artigo: “Inscreva-se na newsletter para continuar lendo. É grátis!” com um campo de e-mail e botão enviar ³³ . Há também um texto “Você possui 1 artigo para ler sem se cadastrar” seguido de um formulário e um botão “Continuar Lendo” que aparece caso o usuário já tenha se inscrito ³⁴ . Para fins da nossa cópia, devemos implementar essa funcionalidade de maneira não intrusiva: podemos simular que após, por exemplo, 1 ou 2 artigos vistos, exiba-se um modal ou overlay pedindo o e-mail para continuar. Como detalhes: deve ter um campo de e-mail, checkbox de consentimento com termos e política, e botões de enviar/continuar. Essa é uma funcionalidade importante a replicar para fidelidade, mas **não precisamos conectá-la a um backend real** – basta simular o fluxo (armazenar em localStorage se o usuário “inscreveu” para não mostrar de novo, etc., se possível).
 - **Seção de comentários:** O Intercept Brasil **não possui seção de comentários aberta** nos artigos (não há nada sobre comentar no site). Assim, **não implementaremos comentários**.
- **Página “Quem Somos” (Sobre):** esta página institucional apresenta informações sobre o Intercept Brasil e sua equipe. No topo há um título “QUEM SOMOS” e uma descrição da missão do veículo ³⁵ . Depois, geralmente vem uma lista de membros da equipe e colaboradores, cada um com foto, nome, cargo e mini-bio ³⁶ ³⁷ . No site original, cada nome de membro da equipe é um link que leva para a página de perfil individual daquela pessoa ³⁸ . Precisamos recriar algo similar:
- Podemos estruturar essa página com um componente de **cartão de membro da equipe**, incluindo foto circular, nome, título (ex: “Editor(a)”, “Repórter”, etc) e um resumo. Poderia ser um grid responsivo (2 ou 3 colunas em desktop, 1 por linha no mobile). No original, por exemplo, vemos *Andrew Fishman – Presidente e cofundador*, *Cecília Olliveira – Editora contribuinte*, etc., com um trecho da bio e um link “→” para ler mais ³⁹ ⁴⁰ .
- Esses “ler mais” links apontam para **páginas de perfil de autor** dentro de `/equipe/nomedoautor` . Vamos implementá-las também (descritas adiante).

- Após listar a equipe, pode haver uma seção para **colaboradores ou autores de colunas** (no original possivelmente listados juntos, dependendo do tamanho da equipe).
- **Perfil do Autor (Página de autor/colunista):** para cada jornalista ou colunista com perfil público, há uma página dedicada contendo sua bio completa e lista de artigos escritos por ele/ela. Por exemplo, a página de Andrew Fishman contém o nome como título, a biografia detalhada e depois uma lista de links das suas últimas matérias ⁴¹ ⁴². Devemos criar um template para esses perfis:
 - Incluir a foto grande do autor (pode ser banner ou ao lado da bio), nome em destaque, bio formatada (vários parágrafos se necessário) ⁴¹.
 - Uma seção de **Contatos** (como email, Twitter, etc.) se disponível ⁴³.
 - Uma lista (possivelmente paginada) de artigos escritos por esse autor, em formato similar à listagem de categoria (títulos clicáveis com data) ⁴⁴. No original, esses títulos vêm com o texto “Ver mais [título] [data] [Nome Autor] ...” ⁴⁵, mas podemos simplificar para apenas mostrar título e data, já que o contexto do autor já se sabe. Se o autor participou de co-autoria, pode aparecer outros nomes também – isso é um detalhe de conteúdo mais do que de design, mas podemos mostrar todos autores de cada post listado.
 - Essa página de perfil pode ser roteada como `/equipe/[slug]` ou similar.
- **Página “Seja nossa fonte” (Contato para denúncias):** é uma página de **contato seguro para whistleblowers**. No original tem o título “O Intercept Brasil quer receber suas denúncias” e instrui como entrar em contato com a equipe de forma segura (Signal, e-mail, correio) ⁴⁶ ⁴⁷. Para recriar:
 - Estruturar com um texto informativo (várias seções com subtítulos “Como fazer uma boa denúncia?”, “É seguro entrar em contato?”, etc.) ⁴⁸ ⁴⁷. São basicamente FAQ orientando o usuário sobre procedimentos de envio de informação confidencial.
 - Podemos usar componentes de **Accordion** aqui (por exemplo, do shadcn/ui) para cada pergunta/resposta, tornando a página mais interativa – isso seria até uma melhoria sobre o original (que parece ser texto estático). Por exemplo, cabeçalhos como “##### É seguro entrar em contato conosco?” seguidos de texto ⁴⁷ podem virar itens de accordion expandíveis.
 - Incluir destaque para dados de contato seguros: número de telefone para Signal/WhatsApp ⁴⁹, e-mail de contato (ex.: ) ⁵⁰ e endereço postal (caixa postal) ⁵¹. Esses elementos devem ser enfatizados (talvez em um cartão ou com ícones adequados).
 - Garantir que nenhum dado sensível real seja exposto – podemos usar placeholders (ex.: um número de telefone fictício ou aquele do Intercept se não for problemático, mas melhor usar fictício para evitar uso real).
 - Esta página não tem um formulário – é informativa. Porém, podemos incluir um **formulário de contato simples** (nome, email, mensagem) caso queiramos simular uma forma direta de contato – não presente no original, mas útil se o cliente desejar interação. Mas atenção: se não for pedido, mantenha como apenas informativa.
- **Página de Newsletter:** o Intercept tem uma página convidando à inscrição na newsletter gratuita. O original (em português) tem o título chamativo “*Jornalismo que Investiga o que o poder quer esconder.*” e explica os benefícios de assinar a newsletter ⁵². Elementos importantes:
 - Um campo de inscrição (email) e botão, com confirmação de aceitação de termos e privacidade (checkbox) ⁵².

- Destaques do porquê assinar: no site há vários bullets com ícones e texto, ex: **“Investigações exclusivas – Conteúdo que incomoda os poderosos.”**, **“Saiba antes – você fica sabendo em primeira mão...”**, **“Análises profundas – contexto e crítica...”**, **“Sem paywall – acesso gratuito...”**, **“Jornalismo independente e corajoso – sem rabo preso...”**, **“Atualizações semanais...”** ⁵³ ⁵⁴ . Cada item parece ter um ícone ilustrativo ao lado ⁵⁵ . Precisamos replicar isso provavelmente como uma grade ou lista estilizada com ícones (podemos usar ícones de biblioteca ou criar SVGs simples representativos, já que não podemos copiar os originais).
- Possivelmente um elemento visual com texto animado repetido (no HTML havia linhas repetindo **“Cadastre-se e receba gratuitamente...”** como um marquee ou um destaque animado) ⁵⁶ ⁵⁷ . Poderíamos recriar com um pequeno efeito de texto rolante ou simplesmente um texto destacado repetido como design element.
- Depoimentos de leitores: no original, há citações de pessoas (nomes como Isabel, Ana, Verônica, Rafael) elogiando o trabalho do Intercept e encorajando apoio ⁵⁸ ⁵⁹ . Isso pode ser implementado como um slider de testemunhos ou um simples conjunto de citações estilizadas (por exemplo, texto entre aspas ou em itálico com um travessão e nome da pessoa). Para fidelidade, podemos inserir 2-4 depoimentos fictícios ou inspirados nesses (sem copiar textualmente).
- Por fim, repetir o formulário de inscrição no fim da página para enfatizar conversão.
- Toda essa página deve ser altamente responsiva e atraente, possivelmente usando bastante elementos de design (cores de fundo em seções alternadas, ícones, etc.).
- **Página de Busca (Resultados):** O site original suporta busca (há indícios de páginas “Você pesquisou por X”). Precisaremos implementar uma funcionalidade de busca para o conteúdo:
 - No header do site, incluir um ícone de **lupa**. Ao clicar, pode expandir uma barra de busca (pode ser um input que desliza do topo ou ocupa o centro da tela).
 - A página de resultados de busca mostrará uma listagem de posts que correspondem ao termo. No WordPress, isso seria uma página `/?s=termo`. Vamos implementar como um **route** `/buscar?query=` ou `/search/[query]`. Escolha: podemos fazer um **client-side search** filtrando uma lista de posts carregados, ou simular com páginas estáticas pré-filtradas (já que conteúdo é estático e fictício). Para simplicidade, talvez implementemos um componente de busca client-side que filtra os dados das matérias já existentes (podemos ter um JSON com títulos e conteúdo para busca).
 - O layout do resultado de busca será similar a uma categoria: lista de cards de matéria correspondentes, talvez com um título tipo “Resultados da busca por: *termo*”. Exemplo (fictício): “Você pesquisou por ‘aborto’...” no topo, depois lista de matérias contendo a palavra ⁶⁰ . Não copiar conteúdo, apenas reproduzir o estilo.
 - Indique se nenhum resultado for encontrado (mensagem “Nenhum resultado para...”).
- **Páginas de FAQ, Privacidade e Termos:** no rodapé há links para **Perguntas Frequentes**, **Política de Privacidade** e **Termos de Uso** ⁶¹ . Precisamos criar páginas para elas:
 - **Perguntas Frequentes:** O Intercept Brasil possui uma FAQ focada nos doadores. No conteúdo extraído, vemos perguntas como “POR QUE DEVO APOIAR O INTERCEPT?” e respostas detalhadas ⁶² ⁶³ . Devemos formatar isso de forma agradável – possivelmente também usando um componente Accordion para cada pergunta (melhor UX para FAQ). As perguntas são títulos (p.ex. `<h4>` ou `<h5>`) e as respostas em parágrafos ⁶⁴ ⁶⁵ . Inclua todas principais (por ex: Como doar?, Já sou doador, preciso pagar assinatura?, doação pequena faz diferença?,

dados e privacidade do doador, etc., conforme conteúdo original). Novamente, escrever com nossas palavras, mas mantendo a essência.

- **Política de Privacidade e Termos de Uso:** Podemos criar páginas estáticas com texto genérico (ou escrever resumos) – o importante é ter os links ativos e o layout consistente. O conteúdo legal é extenso; talvez usar um texto falso ou se possível reutilizar um texto de política CC0 adaptado. O estilo deve ser simples (subtítulos, listas se necessário para cláusulas). Como o foco do prompt é replicar visual e não conteúdo, podemos inserir placeholders aqui (ex.: “<p>Política de Privacidade do site Intercept Brasil clone.</p><p>[Conteúdo da política...]</p>”). O fundamental é lembrar de linkar essas páginas no rodapé e manter consistência.

- **Outros elementos globais:**

- **Cabeçalho (Header) e Menu:** presente em todas as páginas no topo. Discutido em detalhes abaixo, mas mencionando aqui: contém o logotipo, menu de navegação principal e botões de ação (doar, buscar, etc.).
- **Rodapé (Footer):** presente em todas as páginas no final. Contém links para seções principais, links institucionais (sobre, FAQ, privacidade, termos) e ícones de redes sociais ⁶¹. Discutiremos adiante.

Com esse mapeamento, cobrimos a estrutura geral. Em seguida, definiremos os componentes de interface necessários para construir essas páginas conforme o layout original.

Componentes Visuais Principais

Para reconstruir a interface do Intercept Brasil de forma modular e reutilizável, identificamos os seguintes componentes-chave a serem desenvolvidos:

- **Header e Menu de Navegação:** O cabeçalho do site é fixo no topo e contém:
- **Logo do Intercept Brasil:** no site original é um logotipo textual em preto (“Intercept” com uma tipografia sans serif pesada). Teremos que criar um logo alternativo ou usar apenas texto estilizado (mesma fonte/peso, já que não podemos usar o SVG original). Este logo deve linkar para a homepage.
- **Botão “Quero Apoiar” (Doação):** um call-to-action destacado, presente no topo à esquerda ou direita. No HTML original ele aparece antes dos itens do menu ⁶⁶. Suponho que visualmente fique alinhado à direita como um botão. Iremos implementá-lo como um botão de destaque (cor diferenciada, possivelmente um verde ou amarelo bem chamativo – o Intercept usa verde neon em alguns casos – ou vermelho) para chamar atenção. Ele deve permanecer visível em desktop; em mobile pode ser reduzido a um ícone de coração ou doação ou movido para dentro do menu hambúrguer devido a espaço limitado.
- **Itens de menu principais:** uma lista horizontal de seções: *Newsletter, Poder, Direitos, Meio Ambiente, Segurança, Tecnologia, Especiais, Vozes, Vídeos, Seja nossa fonte, Quem Somos*. No HTML, esses aparecem como uma lista com subitens aninhados ⁶⁷ ⁶⁸. Em desktop, vamos exibí-los na barra do header, possivelmente com submenus dropdown para “Especiais” e “Vozes”:
 - **Especiais:** ao passar o mouse (desktop) ou clicar (mobile), mostra uma lista dos especiais disponíveis (séries). No HTML, vemos que “Especiais” não tem um link direto, apenas abre submenu com “Todos os Especiais” e várias séries listadas ⁶⁹. Precisaremos replicar isso. Em desktop, use um dropdown (shadcn/ui tem componente **DropdownMenu** ou **NavigationMenu** que pode ser útil aqui). Em mobile, ao tocar “Especiais”, deve navegar para uma tela interna do menu com os itens (como o original, que mostra um “Voltar” e lista de séries) ⁷⁰. Podemos implementar o mobile menu como uma **off-canvas**

(Drawer/Sheet) cobrindo a tela; dentro dele, se clicar em Especiais, podemos usar um sub-view.

- **Vozes:** similar a especiais, mas lista os colonistas principais. No HTML, “Vozes” abre submenu com “Todas as vozes” e nomes dos colonistas ⁷¹. Faremos igualmente dropdown/hamburger sublista.
- Os demais itens (Newsletter, Poder, Direitos, etc.) são links diretos para suas páginas.
- **Ícones de redes sociais no header:** No HTML há uma série de ícones (WhatsApp, Bluesky, Instagram, Facebook, LinkedIn, YouTube, TikTok, Google News, Flipboard) listados após o menu ⁷². No site, esses podem estar dispostos no topo (por exemplo, como ícones discretos no canto superior direito) ou apenas no footer. Observando melhor, parecem estar dentro do header mesmo (talvez numa segunda linha ou um menu expandido?). Como há muitos, talvez no design atual eles fiquem em um menu “Siga-nos” expansível. Podemos optar por coloca-los no rodapé (mais convencional). No original, as redes sociais aparecem tanto no header quanto repetidas no footer ⁷² ⁷³. Para simplificar, podemos deixar apenas no footer. **Decisão:** Colocar os ícones de rede principalmente no rodapé (onde os usuários esperam encontrá-los) e eventualmente esconder no header em mobile. Se quiser fidelidade máxima, incluir no header desktop como uma lista horizontal de ícones ao lado do menu.
- **Botão de busca (lupa):** Não aparece no HTML extraído, mas deduzimos que há um ícone de busca (provavelmente um 🔍 no canto do header). Precisamos adicionar para permitir acesso à busca.
- **Comportamento responsivo do header:** Em telas desktop, o menu completo aparece (logo + itens + talvez doação + talvez ícones sociais + busca). Em telas **médias e pequenas**, vamos colapsar: usar um **menu hambúrguer**. O ícone de menu (três barras) ficará no canto (esquerdo ou direito, conforme design – provavelmente canto direito, com o logo alinhado à esquerda). Ao tocar, abre um painel (drawer) do lado ou um menu dropdown expandindo. O original provavelmente usa um off-canvas fullscreen. No HTML notamos a palavra “Voltar” associada aos submenus, típico de um menu mobile em tela cheia ⁷⁰ ⁷¹. Implementaremos com o componente **Sheet** do shadcn/ui para um painel lateral ou de baixo cobrindo toda a viewport e listando os links de menu. Dentro do menu mobile:
 - Listar todos os itens principais. Para itens com submenu (Especiais, Vozes), em vez de dropdown hover (que não funciona em touch), faremos o seguinte: ao tocar, o menu lista oculta os outros e mostra os subitens *com um cabeçalho “Voltar”* no topo para retornar. Isso espelha o funcionamento original (vimos “Voltar” no HTML) ⁷⁰. Usando React state ou shadcn’s nested navigation, podemos conseguir isso.
 - Alternativamente, podemos usar o componente **Accordion** para que itens como Especiais e Vozes expandam seus subitens dentro do menu mobile (isso é uma forma mais simples e acessível também). Ex.: cada menu item que tem children vira um AccordionItem, e os children são links indentados.
 - Garanta que o botão de doação e de busca apareçam também no menu mobile (podem ficar no topo do drawer).

Em termos de estilo, o header deve ser fixo no topo durante o scroll (pelo menos em desktop) – o original provavelmente fixa, mas precisamos confirmar. Considerando UX, seria bom fixar para acesso rápido à navegação e busca.

- **Footer (Rodapé):** O rodapé do Intercept Brasil contém:
 - Links para as principais seções e páginas utilitárias (possivelmente separados em colunas). No HTML vemos repetidos: Matérias, Vozes, Vídeos, Newsletter, Seja nossa fonte ⁷ e ao lado Quem Somos, FAQ, Política de Privacidade, Termos de Uso ⁶¹. Provavelmente o layout coloca: uma coluna com seções de conteúdo, outra com institucionais. Vamos fazer algo similar:
 - Coluna 1: Matérias (todas as notícias), Vozes, Vídeos, Newsletter, Seja nossa fonte.

- Coluna 2: Quem Somos, Perguntas Frequentes, Política de Privacidade, Termos de Uso.
- Ícones de redes sociais: novamente uma fileira de ícones (WhatsApp, Instagram, Facebook etc.)⁷³. No rodapé, esses podem ser apresentados como ícones pequenos clicáveis alinhados horizontalmente.
- Uma informação de copyright: “©2025 Intercept Brasil. Todos os direitos reservados.”⁶¹ – estilizar em fonte menor.
- Possivelmente uma frase de missão curta ou endereço, mas no HTML não vimos além do copyright.
- Estilo: geralmente rodapé tem fundo escuro ou cinza. No Intercept, pode ser preto com texto branco (o logo do Intercept Brasil em algumas versões é branco sobre preto). Se o original usa esquema claro, podemos manter branco, mas é comum contrastar. Precisamos decidir baseados no visual global: Como o site todo parece fundo branco, talvez o rodapé seja preto com texto claro para contraste. Essa seria uma boa escolha de design (muitos sites jornalísticos fazem rodapé escuro). Mas se quisermos 100% idêntico, teríamos que saber: O Intercept Brasil possivelmente usa preto no rodapé. Vamos supor que sim, e implementar rodapé com fundo **#000** e texto **#fff** para segurança, e links com possivelmente sublinhado ou cor diferenciada ao hover.
- **Card de Matéria (Article Card):** componente reutilizável para listar posts em páginas como homepage, categorias, etc. Características:
 - Inclui uma **imagem de thumbnail** da matéria (no site original, cada matéria tem uma imagem destacada). Nos trechos HTML, porém, vimos apenas “**[Image]**” placeholders sem detalhes. Provavelmente, eles exibem uma thumbnail (talvez 3:2 landscape format) para cada notícia listada. Precisamos suportar isso.
 - **Categoria ou série tarja:** se a matéria pertence a uma série especial, ou se for formato vídeo, eles adicionam um rótulo antes do título. Ex: “Cartas Marcadas” ou “VÍDEO:” em destaque no início do título nos cards⁵¹⁵. Vamos implementar isso como um pequeno `` estilizado (ex.: com cor de fundo ou texto uppercase) antes do título. Pode ser posicionado acima do título ou inline antes do texto do título, conforme design. Se for vídeo, usar uma cor diferente ou um ícone de play talvez.
 - **Título da matéria:** texto em destaque (talvez `<h2>` nos cards). Deve ser clicável (link para a página do post).
 - **Resumo/descrição curta:** algumas listas incluem um trecho do começo do artigo ou uma breve descrição. No HTML extraído, notamos que em algumas séries eles incluem um parágrafo resumido (por ex., em Cartas Marcadas, depois do título, há uma frase resumo: “Pressionado pelo bolsonarismo, Motta prefere atacar o jornalismo...”⁵). Ou em outros casos, uma parte do primeiro parágrafo truncado. Vamos implementar um campo de resumo opcional nos cards (exibir ~2 linhas de texto).
 - **Meta (data e autor):** nos cards deve aparecer a data de publicação e o autor ou autores. No HTML da categoria, isso aparecia ao final de cada item. Ex: “9 de setembro – Paulo Motoryn, Thalys Alcântara”⁷⁴. Formatemos a data no padrão do site (dia + mês abreviado + ano se necessário). Autor pode aparecer em itálico ou pequena caps.
 - **Layout dos cards em grid ou lista:**
 - Na homepage, possivelmente usam um layout modular: O topo tinha dois destaques horizontais, depois uma grade de 2 ou 3 colunas. Porém, olhando o HTML linear, fica difícil ter certeza. Suspeitamos que após os dois destaques principais, a lista de notícias comuns estava em duas colunas (cada item ocupando 50% da largura em desktop). Podemos fazer assim: em telas grandes, dispor os cards em uma grade de 2 colunas para

otimizar espaço; em telas muito grandes (xl), talvez 3 colunas. Nas páginas de categoria, possivelmente 2 colunas também. Em mobile, obviamente fica 1 por linha.

- Haverá variações: o **card destaque (hero)** no topo da homepage é provavelmente um componente similar mas maior (pode ocupar toda largura ou 2/3 + outro 1/3, etc). Poderemos criar um **HeroCard** variante do ArticleCard, que mostra a imagem em tamanho grande e o título maior, usado para a notícia principal.
 - Em páginas de série e voz, talvez mantêm 1 coluna (já que essas páginas são texto-intensivas), mas não certo. Para consistência, implementar grade 2 colunas padrão e cair para 1 em mobile, e se necessário permitir 1 col em CSS para certas páginas.
- **Componente reutilizável:** Criaremos `<ArticleCard>` para notícias comuns e, se necessário, `<FeaturedArticleCard>` para destaque (pode herdar ArticleCard com modificações de estilo).
 - **Componentes de Texto e Tipografia:** Definiremos estilos para headings, parágrafos, listas e outros elementos de texto:
 - **Tipografia base:** Precisamos escolher fontes similares às do Intercept. O Intercept (US) usava uma fonte sans-serif pesada custom para títulos (*TI Actua*), e uma serif ou sans para corpo. No Brasil, pela aparência do conteúdo, arriscamos que usam sans-serif para tudo ou uma combinação. Talvez títulos em sans-serif bold, corpo em serif para facilitar leitura prolongada.
 - *Sugestão:* Usar uma fonte sans-serif geométrica ou grotesca pesada (por exemplo **Archivo Black**, **Impact**, ou **Montserrat ExtraBold**) para **títulos e headings** – para replicar o impacto do texto do Intercept ²⁰. Para o corpo do texto, uma serif clássica (por exemplo **Georgia**, **Merriweather** ou **Libre Baskerville**) ou uma sans humanista (como **Source Sans** ou **Open Sans**). Como não sabemos ao certo, optar por uma combinação agradável: ex.: Títulos com *Arial Black* ou *Oswald Bold* em uppercase, corpo com *Georgia* regular. Outra opção: usar só sans-serif (ex. **Inter** ou **Roboto** para corpo, e alguma variação heavy para títulos). O importante é boa legibilidade e similaridade visual.
 - Tamanhos: O título principal do artigo (H1) deve ser bem grande (talvez ~2rem ou mais), headings (H2, H3) progressivamente menores. O texto do corpo ~1rem (16px) ou um pouco maior (18px) para legibilidade, com line-height ~1.6. As legendas e meta (autor/data) menores (0.875rem, 14px).
 - **Rubricas (kickers)** como “Censura oficial” acima do título podem ser estilizadas em **small-caps** ou uppercase, cor destaque (talvez a cor primária) e um tamanho pequeno ou igual ao corpo porém em bold ²⁰. Isso para diferenciá-las.
 - **Tarjas de série/vídeo nos cards:** estilizar como pequenas labels, talvez com fundo colorido (poderíamos usar uma cor padrão do Intercept – se for verde, por exemplo, usar esse verde de fundo com texto branco).
 - **Links:** Links no corpo do texto devem ter destaque. O Intercept possivelmente usa sublinhado cinza ou texto roxo. Para segurança, podemos estilizar com sublinhado sólido e cor azul escuro, ou uma cor de destaque do nosso design (ex: verde-escuro). No hover, mudar cor ou sublinhado. Importante: links externos devem abrir em nova aba (atributo `target="_blank" rel="noopener"`).
 - **Quotes (blocos de citação):** Se houver `<blockquote>`, estilize com margem, fonte itálica e talvez uma barra lateral.
 - **Listas:** utilizar estilo com marcadores simples (círculos) para ul e números para ol, com indentação adequada.
 - **Separadores:** No conteúdo do artigo, eles usam um separador `* * *` (três asteriscos centrados) para indicar uma quebra de seção ou fim do trecho da série ⁷⁵. Vamos interceptar

isso e renderizar como um `<hr>` estilizado (linha pontilhada ou três asteriscos centralizados conforme preferir).

- **Tabela de matérias recentes / relacionadas:** isso não é tanto tipografia quanto layout, mas caso precise mostrar artigos relacionados, usar um estilo consistente de mini-lista.

- **Componente de Compartilhamento**:** Pode ser um pequeno conjunto de botões com ícones (WhatsApp, Facebook, X/Twitter (no original Bluesky em vez de Twitter), etc.). Podemos usar ícones de FontAwesome ou Heroicons. Precisamos criar links de compartilhamento:

- WhatsApp: usar API link `https://api.whatsapp.com/send?text=[TITLE] [URL]`.

- Facebook: `https://www.facebook.com/sharer/sharer.php?u=[URL]`.

- Bluesky (se incluirmos): no original, Bluesky aparece, mas podemos substituir por Twitter para efeito similar, pois Bluesky não tem share via link simples. Ou incluir um link genérico para perfil.

- E-mail: não vi, mas às vezes incluem envelope mailto – opcional.

- Essa barra de compartilhamento aparece no topo e fim do artigo. Implementar um componente `<ShareBar>` que possa ser colocado em ambos os lugares. Em mobile, pode ser horizontal scroll se não couber.

- **Componente Newsletter Subscribe (Inline/Modal):** Precisamos de dois:

- **Página Newsletter** – já detalhada, com campos e tudo.

- **Pop-up/Overlay de gating** – um componente que pode ser invocado quando o usuário atinge limite de leitura. Este seria um `<NewsletterModal>` que contém: um título convidativo (ex: “Inscreva-se na newsletter para continuar lendo. É grátis!”), input de email, checkbox “Aceito os termos...”, botão enviar, e talvez uma explicação que não é paywall de verdade (no original: “Este não é um acesso pago, adesão é gratuita”³³). Também um botão “Continuar lendo” para fechar caso já inscrito. Shadcn/ui tem **Dialog** ou **AlertDialog** que podemos usar para isso.

- Precisamos controlar estado: exibir o modal após X artigos lidos – podemos simular com um contador global (context or localStorage). Essa lógica não precisa ser perfeita para agora, mas é bom mencionar no código/prompt.

- **Cookie Consent Banner:** No rodapé do HTML vimos uma mensagem de cookies com um botão “Ok”⁷⁶. Vamos implementar um pequeno banner fixo no rodapé da tela para conformidade:

- Texto curto: “Utilizamos cookies para garantir a melhor experiência... Ao continuar navegando você concorda... [Política de Privacidade]. [Botão OK]”.

- Pode usar o componente **Toast** ou **Popover** do shadcn. Ao clicar OK, esconde o banner (salva um flag no localStorage).

- Estilizar para não sobrepor o rodapé real (talvez posicionar acima dele ou com margin-bottom se rodapé fixo).

- **Outros Componentes Possíveis:**

- **Carousel/Slider:** O site tem alguns locais onde múltiplos itens passam como carrossel. Ex: depoimentos de newsletter poderiam ser slider, ou destaques da home. Se decidirmos implementar slider, podemos usar uma biblioteca leve (p.ex. embutir o CSS do Embla Carousel) ou usar scroll snapping CSS puro. Isso fica como detalhe adicional se houver tempo.

- **Accordion para FAQ e Menu Mobile:** já mencionado – reusamos para FAQ e possivelmente menu mobile.
- **Sheet (Offcanvas menu):** para menu mobile.
- **HoverCard ou Tooltip:** poderíamos usar para mostrar título completo de artigos se truncados, mas talvez desnecessário.

Comportamento Responsivo

O site deve funcionar bem em **diversos tamanhos de tela (desktop, tablet, mobile)**, adaptando o layout sem perder funcionalidades. Observações e requisitos de responsividade:

- **Menu responsivo:** Conforme detalhado, o header se transforma em um menu hambúrguer no mobile. Ao clicar, expande um painel de menu cobrindo a tela. O HTML original indica submenus aninhados com botão “Voltar” para retornar ⁷⁰ ⁷¹. Implementaremos esse comportamento usando JavaScript/React. Em telas médias (tablet), podemos ainda mostrar o menu completo ou parcialmente. Uma abordagem comum:
- **Breakpoint suggestion:** Usar `md` (768px) como ponto para ativar menu completo. Abaixo disso, usar hambúrguer.
- Em desktops grandes, todos os itens aparecem na barra; em intermediários, se faltar espaço, possivelmente alguns podem colapsar – mas dado que são ~10 itens, numa largura >1024px cabe.
- **Grade de notícias:**
 - Em **mobile (até ~640px)**: todos os cards de matéria empilhados em uma coluna única para leitura fácil. Imagens full-width (ou quase full, com um pouco de margem).
 - Em **tablets (≥768px)**: podemos usar 2 colunas para a lista de cards, já que há espaço. Os destaques principais ainda podem ser 1 por linha (ou um layout especial: ex. 2 colunas onde coluna esquerda é notícia principal grande e direita duas menores).
 - Em **desktop large (≥1024px ou 1280px)**: eventualmente 3 colunas se ficar bom, mas cuidado para não comprometer o tamanho do texto. Como o Intercept valoriza legibilidade, é provável que mantenham no máximo 2 colunas de artigos e optem por rolagem vertical longa em vez de colunas muito estreitas. Vamos seguir essa linha: **máximo 2 colunas** em listagens, para garantir textos (resumos, títulos) em largura adequada.
 - Em páginas de **Série** ou **Autor**, talvez manter 1 coluna (porque podem incluir descrição longa no topo). Depende do design final que preferirmos; podemos manter 2 colunas para consistência, mas se o texto descritivo no topo ocupa toda largura, a listagem pode iniciar abaixo. Simples: implementamos 2 colunas globalmente e usamos CSS grid; se quisermos 1 coluna em algum contexto, podemos override via CSS (ex: `.series-page .posts-grid { grid-template-columns: 1fr; }`).
- **Imagens responsivas:** devemos usar `<Image>` (no Next.js) ou `` com `max-width: 100%` para que todas as imagens encolham adequadamente em telas pequenas.
- Para imagens de destaque (hero), podemos mostrar versão maior; em mobile, talvez mostrar centralizada e título sobreposto? Mas no Intercept, creio que títulos não sobrepoem imagens, ficam abaixo. Então simplesmente reduzir a imagem para a tela do mobile.
- Se usamos Next Image, fornecer versões menores para mobile via `sizes` attribute.
- **Tipografia responsiva:** em telas muito pequenas, talvez reduzir um pouco os títulos para não estourar bordas. Ex: H1 no desktop 2.5rem, no mobile 1.75rem. O Tailwind facilita com utilidades responsivas (ex: `text-4xl md:text-5xl` etc).
- **Margens e espaçamentos:** Certifique-se de adicionar padding/margem nas laterais em mobile para que o texto não toque as bordas da tela. Um gutter de ~16px em mobile e ~32px em desktop nas extremidades é uma boa prática.


- **Ordem de elementos:** Em mobile, elementos como a barra de compartilhamento que no desktop podem aparecer lateral ou no topo, devem se reposicionar. Provavelmente, no Intercept:
- No desktop, talvez a barra de compartilhamento do artigo aparece lateralmente (fixa ao lado do texto)? Não temos certeza. O HTML os colocou em linha com o título mesmo ²¹. Talvez fica no topo mesmo. De qualquer forma, em mobile não caberia lateral, então ou some ou vai pro final. Podemos simplificar e deixar a barra somente no topo e final do artigo sempre, e ocultar a duplicata conforme o caso.
- **Tabela/responsividade de vídeos:** Se incorporarmos vídeos (YouTube iframes ou MP4s), usar estilos responsivos (ex: aspect-ratio 16/9).
- **Testar em tamanhos:** Após implementar, testar em ~360px (smartphone pequeno), ~768px (iPad), ~1024 (desktop padrão) e >1400px (monitores grandes). Ajustar breakpoints se necessário.

Em resumo, garantir que nenhum elemento fique cortado ou sobrepondo em mobile, e que em desktop o aproveitamento de espaço seja otimizado mas mantendo legibilidade.

Hierarquia de Conteúdo e Tipografia

A estrutura de títulos, textos e destaques no Intercept Brasil segue uma hierarquia clara, que orienta o usuário através do conteúdo:

- **Rubrica (Kicker) acima do título:** Quando presente, é um elemento de contexto que precede o título de uma matéria ou seção. Exemplos: “Censura oficial” antes do título do artigo ²⁰, ou “Especial” antes do título de uma série ¹⁰. Estilização sugerida: fonte **sans-serif** pequena, toda em maiúsculas ou small-caps, cor de destaque (talvez a cor primária do site), com algum espaçamento abaixo. Pode ser envolta em `<small>` ou classe específica.
- **Título Principal (H1):** usado para títulos de notícias em páginas de post, títulos de páginas principais (ex: nome da categoria, nome da série, manchete da home). Deve ser **marcante e legível**. Fonte sans-serif bem pesada ou serif display, tamanho grande. Ex.: “Hugo Motta usa até seguranças contra repórteres — e imprensa silencia” é um H1 de artigo ²⁰; “Israel: estado genocida” é H1 de página de série ⁷⁷. Em páginas de listagem (categoria, voz, etc.), podemos usar H1 para o nome da seção (Poder, Vozes) ³ ⁷⁸.
- **Subtítulo / Descrição (H2/H3):** Logo abaixo de alguns títulos de página aparece um texto descritivo. Ex.: na página da série, o parágrafo introdutório sobre o que a série aborda ⁷⁹; ou na newsletter page, o subtítulo “O Intercept faz reportagens profundas...” ⁸⁰. Podemos estilizar essas descrições em texto de destaque (talvez `<h2>` estilizado normal ou apenas um `<p>` com classe “lead” de Tailwind para aumentar um pouco).
- **Intertítulos no conteúdo (H2/H3):** Dentro de artigos longos, são usados para dividir seções. No exemplo do artigo, “O cercadinho do silêncio” aparece como um intertítulo (### no HTML) dentro do texto ²⁵, e “Estopim: pergunta sobre rachadinha” como próximo intertítulo ⁸¹. Estes devem ser estilizados claramente maiores que o corpo, mas menores que o título principal. Podemos usar `text-xl font-bold mt-6 mb-2` por exemplo.
- **Corpo do texto:** parágrafos regulares formam o corpo das reportagens e páginas informativas. Tipicamente fonte tamanho ~16px (1rem) ou 18px para conforto, serif se disponível ou sans-serif neutra. Alinhamento justificado ou alinhado à esquerda? O Intercept provavelmente alinha à esquerda (justificado às vezes causa rios de espaço). Vamos alinhar à esquerda e manter espaçamento de linha ~1.6. Parágrafos separados por margem vertical ~1em.
- **Ênfases no texto:** Utilize `` para negrito (Tailwind: font-semibold ou bold) e `` para itálico (Tailwind já aplica itálico). Se houver algo como `<mark>` ou destacar texto, podemos usar background amarelo claro para replicar marca-texto (caso necessário).

- **Listas:** As páginas FAQ e “Seja nossa fonte” têm listas pontuadas (bullet points) dentro das respostas ⁸² ⁸³. Estilize `` com list-disc e padding-left adequado, `` com margin-bottom para separar itens.
- **Citações (blockquote):** Embora não vimos explicitamente nos trechos, se algum artigo tiver citação destacada, estilo: borda/esquerda ou itálico, e margem. Podemos prever esse estilo.
- **Legendas de imagem:** Aparecem imediatamente após imagens, em tamanho menor. No exemplo do artigo, a legenda “Hugo Motta, do Republicanos da Paraíba, está usando estratégias...” descreve a imagem ⁸⁴. Estilo: font-size 0.875rem, color: #555 (cinza médio), talvez itálico. Se queremos identificar crédito do fotógrafo, poderia ser bold dentro da legenda se presente.
- **Metadados (autor, data, leitura):** No início do artigo temos autor e data ²³. Estilize-os como: autor em negrito normal ou small-caps, data em regular, separados por vírgula ou um ponto. Poderíamos também mostrar tempo de leitura estimado (não consta explicitamente, mas poderíamos calcular ~X min de leitura se quisermos adicionar funcionalidade).
- **Tarjas e etiquetas:** O termo “tarjas” referido no pedido se manifesta como aquelas etiquetas com nome de série, formato ou seção antes de títulos nos cards e artigos. Já cobrimos: exibir “Cartas Marcadas” antes de título nos cards de série ⁵, “VÍDEO:” antes de título nos cards de vídeo ¹⁵, etc. Visualmente, podemos fazer uma pequena caixa com background colorido e texto branco uppercase, ou texto colorido simples. Ex: estilo `.tag { font-size: 0.8rem; font-weight: bold; color: white; background: #000; padding: 2px 4px; margin-right: 4px; border-radius: 2px; }` ou similar.
- **Botões e links de ação:**
 - O botão “Quero Apoiar” no header e “Quero Doar” no footer ⁶⁶ ⁸⁵ devem ter destaque: cores contrastantes e talvez um ícone de coração/doação junto do texto. Arredondar bordas levemente e usar padding.
 - Botões de formulário (newsletter, continuar lendo) estilizados de maneira consistente, talvez cor de destaque do site (que poderíamos definir uma **cor primária** – ex: verde #00b050 ou um roxo – precisamos escolher e usaremos coerentemente em botões, links destacados e tarjas).
 - Links “Ver mais” que aparecem antes dos títulos nos cards não precisam ser visíveis como texto “Ver mais” (no HTML eles incluem essa frase dentro do `<a>`). Podemos decidir se vamos exibir o texto “Ver mais” no front-end ou se faremos apenas o título como link e usar CSS (ex: um pseudo-elemento “→” ou “...”). No original, possivelmente exibem “Ver mais” como um pequeno texto ou um ícone de seta indicando que é clicável ¹. Para fidelidade textual, podemos manter “Ver mais” visível antes de títulos de cards.
- **Contrast and Accessibility:** O site original provavelmente segue boas práticas de contraste (texto preto em fundo branco, links com contraste). Garantir que nossas cores escolhidas atendam WCAG para acessibilidade. Tamanhos de fonte também.
- **Iconografia:** Além dos logos de redes sociais e botão busca, outros ícones:
 - Um ícone de menu (três barras) para mobile.
 - Talvez um ícone de **voltar** ou seta para o “Voltar” no menu mobile.
 - Ícones ilustrativos na página de newsletter para cada bullet (poderíamos usar, por ex.,  para “sem paywall” (cadeado riscado),  para “análises profundas”,  para “investigações exclusivas”, etc., ou usar biblioteca de ícones).
 - Ícone de check ou sucesso quando cadastro newsletter for concluído (no HTML aparece mensagem “Concluído, email cadastrado com sucesso!” ⁸⁶, podemos exibir isso com um toast ou mensagem verde).

Resumindo, a hierarquia deve guiar o usuário: títulos claros, subtítulos informativos, corpo legível e elementos de interface (botões, links) facilmente identificáveis.

Padrões de Navegação e Interações do Usuário

A navegação do site Intercept Brasil é tradicional mas eficaz – queremos replicar não apenas a aparência, mas também a *experiência* ao usar o site:

- **Navegação por seções:** O menu principal permite que o usuário acesse rapidamente as seções de interesse (política, direitos, etc.) com um clique. No nosso clone, ao clicar em um item de menu:
- Se for seção direta (ex: Poder), levamos à página dessa categoria, carregando a lista de matérias filtradas.
- Se for um item com submenu (Especiais, Vozes), em desktop abrirá o dropdown on hover ou on click. Em mobile, fará a transição no drawer para a lista interna de subitens. Essa interação deve ser suave – use animações de transição (ex.: fade ou slide) para mostrar o submenu. O botão “Voltar” no topo do submenu mobile ao ser clicado retorna ao menu principal – implemente isso com animação de slide para dar a sensação de ir e voltar.
- **Scroll e sticky elements:** Quando o usuário rola a página:
- O header deve permanecer visível (fixo) no topo, para facilitar a mudança de seção a qualquer momento. Podemos aplicar um efeito de esconder/diminuir o header quando o usuário começa a rolar para baixo e mostrar de volta ao rolar para cima (opcional, melhora UX em mobile).
- Nenhum outro elemento principal é sticky no site original, exceto possivelmente a barra de compartilhamento. Se desejarmos, poderíamos fazer a barra lateral de compartilhamento (em desktop) ficar fixa enquanto se lê o artigo. Isso requer CSS position:sticky e top: calculado. Mas se não implementarmos barra lateral, ignorar.
- **Paginação de listas:** Nas páginas de categoria, série e autor, há paginação numérica no final ⁶
⁸⁷. Implementaremos a aparência disso (números com realce para página atual). Não precisamos ter dezenas de páginas de fato, mas podemos simular 2-3 páginas com rotas ou param query. Simplicidade: implementar um componente `<Pagination>` que mostra botões numerados e “...” se necessário, e manipula estado/rota. Ex.: `<Pagination totalPages={X} currentPage={Y} onPageChange={fn} />`. Se usar Next.js rota estática, podemos pré-gerar algumas páginas para demonstrar (ex.: /poder/page/2).
- **Carregamento de páginas:** Next.js por padrão fará transições rápidas (cliente-side nav) entre páginas, o que melhora a experiência. Podemos adicionalmente usar um **Loading indicator** (shadcn/ui tem `<Loading />` ou usar NProgress bar) ao mudar de página, para visual.
- **Interação com cards:** Quando o usuário passa o mouse (hover) sobre um card de notícia em desktop, podemos adicionar um efeito sutil – por exemplo, elevar o card (shadow-lg) ou mudar leve o background. Isso indica que é clicável. Em mobile, no toque abre diretamente o link.
- **Interações sociais:** Clicar nos ícones de compartilhamento abre uma nova aba para a rede social ou aplicativo. Configure `target="_blank"`. Para WhatsApp, esse link vai tentar abrir o app.
- **Buscas:** O ícone de busca ao ser clicado deve focar o campo de busca. Podemos implementar de duas formas:
- **Modal de busca:** um overlay fullscreen com um input central (muitos sites modernos fazem isso). Ao digitar e apertar Enter, redireciona para a página de resultados de busca com a query.
- **Barra expansível no header:** onde o ícone se transforma num input inline se há espaço. Como o header do Intercept tem muitos itens, talvez o modal seja melhor. Implementar possivelmente com shadcn **Command Palette** ou **Dialog** compondo um search inside.
- A página de resultados exibirá matching posts. Precisamos definir a lógica de busca: indexar títulos, autores e talvez conteúdo. Podemos preparar um JSON ou usar a listagem já disponível para filtrar.
- Certifique de lidar com “no results” casos, mostrando mensagem.
- **Newsletter sign-up fluxo:**

- Na página "Newsletter", quando o usuário preencher o e-mail e clicar "Enviar", exibimos a mensagem de sucesso (no original: "Concluído, email cadastrado com sucesso!") ou de erro ⁸⁶. Podemos simular esse comportamento no front: ao clicar, usar um `setState(submitted=true)` e então condicionalmente renderizar a mensagem de sucesso (e esconder o formulário ou limpar).
- No gating modal do artigo: similar, quando inserir email e enviar, podemos simplesmente fechar o modal e permitir leitura (e talvez marcar no localStorage `subscribed=true`).
- **Links externos e segurança:** Todas as interações de abrir links externos (doares, fontes originais nas matérias, etc.) devem usar `rel="noopener noreferrer"` com target blank, por segurança.
- **Carregamento de imagens preguiçoso:** Implementar **lazy loading** para imagens fora da viewport inicial (Next Image faz isso por padrão ou podemos usar `loading="lazy"`). Assim, a página inicial que tem muitos thumbs carregará mais rápido.
- **Feedback visual:** adicionar transições CSS em hovers de botões, mudanças de estado de accordions, etc., para ficar suave.
- **Acessibilidade:**
 - Incluir `:focus` styles nos links e botões para usuários teclado (Tailwind por padrão pode ser configurado para focus ring).
 - Aria-labels em ícones (ex.: botão hambúrguer `aria-label="Abrir menu"`).
 - Estruturar HTML semântico: usar `<header>`, `<nav>`, `<main>`, `<article>` (para posts), `<section>` com headings para seções de homepage, `<footer>`, etc., adequadamente. Isso ajuda leitura por screen readers. Ex: a lista de cards pode ser uma lista de `<article>` dentro de um `<section aria-label="Últimas notícias">`.
 - O modal de newsletter gating deve ter `aria-modal` e focar no primeiro elemento input quando aberto.
- Menu mobile deve trap focus enquanto aberto e ser fechável com Esc.
- **Estado login/inscrição:** O site do Intercept não tem login, apenas inscrição na newsletter. Portanto, não precisamos implementar sistema de autenticação completo, somente a coleta de email no form.
- **Conteúdo dinâmico simulando real:** Podemos armazenar as matérias e suas propriedades (título, resumo, data, autores, categoria, imagem, series, etc.) em arquivos JSON ou como data estática (ex.: arrays). Em Next.js, poderíamos usar `getStaticProps` (para pages dir) ou importá-los diretamente (for app dir use directly). Isso facilitará popular todas as listas e páginas de detalhe com conteúdo coerente (mesmo que placeholder). Assim, quando navegar para um artigo específico, podemos passar o objeto correspondente.
- **Ensinar o desenvolvedor sobre não reutilizar assets:** Em especial, **imagens:** o clone **não pode usar as imagens reais do Intercept (copyright)**. Então, orientaremos a obter imagens similares em bancos livres (Unsplash, Pexels) ou gerar placeholders. Por exemplo, se uma matéria for sobre política brasileira, usar uma foto ilustrativa genérica de Brasília ou congresso. Para ilustrações de séries, podemos criar gráficos simples ou usar cores sólidas com texto. A logo do Intercept Brasil – podemos criar uma versão textual estilizada ("Intercept" em fonte Impact, por ex) ou um logo fictício.
- No código, centralizar as imagens em uma pasta `/public/images` para fácil substituição.

Estilo Visual (Cores, Fontes e Espaçamentos)

Para replicar a identidade visual do Intercept Brasil sem usar seus assets, definiremos um estilo muito próximo:

- **Paleta de Cores:** O Intercept original utiliza predominantemente **preto e branco** como base (texto preto em fundo branco) com toques de cor para destaques. É conhecido pelo uso ocasional do **verde fluorescente** (#00FF66 aproximadamente) em elementos de destaque (no site global, links e botões às vezes têm esse verde). O Intercept Brasil pode usar esse verde ou possivelmente um **vermelho vivo** em alguns destaques (não confirmado). Para ficar consistente com a imagem “corajosa” do site, podemos adotar:
- **Cor primária:** Verde brilhante (#00b050 por ex.) para botões “Apoiar” e highlights de links. Essa cor em fundo preto se destaca bem. Alternativamente, um tom de **magenta ou vermelho** poderia ser usado (mas isso remete menos à marca; optaremos pelo verde).
- **Cor do texto principal:** #000000 (preto puro) ou um quase preto (#111) para um contraste forte.
- **Cor do fundo:** #FFFFFF (branco) para fundo principal de páginas. Secundariamente, podemos usar um cinza muito claro (#f9f9f9) em algumas seções para separar (por ex. depoimentos da newsletter ou a área do formulário).
- **Cores de link:** Se usarmos verde como primária, links poderiam ser verdes escuros para destacar, ou simplesmente sublinhados pretos (como o estilo editorial pode preferir sobriedade). Talvez um verde escuro ou azul escuro (porque verde claro em texto pode ser de baixa leitura). Podemos fazer: link default sublinhado preto e on hover o sublinhado fica verde, ou link texto verde escuro (#006644) e underline.
- **Cor de destaque inverso:** Em rodapé escuro, usar texto branco e links talvez em verde claro para contraste.
- **Tarjas e labels:** Podemos usar preto com texto branco para tarja de série (ficaria como uma tag). Ou usar o verde primário com branco. Decidir uniformemente: talvez tarjas de série = preto fundo, tarjas de formato (VÍDEO) = verde fundo, só para diferenciar visualmente.
- **Background do header/footer:** Para distinção, poderíamos fazer header branco (floating over page content) com texto preto (um estilo clássico nav). O footer, como mencionado, preferível preto com texto claro. Isso também confere peso visual ao fim da página.
- **Fontes:** (já abordado, recapitulando com final picks)
- **Título e headings:** Usar uma *sans-serif bold* parecida com a do Intercept. Se não podemos usar a custom, alternativas Google Fonts:
 - **IBM Plex Sans Bold** ou **Anton** (mas Anton é muito condensada).
 - **Archivo Black** (pesada e legível).
 - **Impact** (clássica, mas pode ser overkill e não web-safe).
 - **Bebas Neue** (caps only, not good for mixed case).
 - Talvez **Montserrat ExtraBold** ou **Noto Sans Black**.
 - Vamos supor “Archivo Black” para títulos e “Archivo Regular” para texto, ou combinação com serif.
- **Corpo do texto:** Se decidirmos sans para corpo também, pode ser a mesma família normal. Mas para nuance editorial, usar serif:
 - **Merriweather** or **Lora** for body (serif).
 - Suppose: Merriweather Regular, with Merriweather Bold/Italic for emphasis.
 - This yields a nice reading feel and pair with a sans for headings can look professional.

- **Monospace:** não há código ou similares, mas se precisássemos (talvez para mostrar email address?), não realmente necessário.

- **Espaçamento e layout grid:**

- Utilizar um **container central** para limitar a largura máxima do conteúdo em desktops. O Intercept não deve esticar linhas de texto muito compridas. Um max-width de ~ 800px para artigos seria bom. Para a página inicial com duas colunas, talvez um container de 1200px acomoda 2 colunas de 600px. Então, podemos:

- Set container `max-w-screen-xl` (1280px) e `mx-auto` para centralizar.
- Dentro, para artigos, podemos ter uma classe `.prose` (se usar Tailwind Typography plugin) ou definir manualmente `max-w-prose` ~ 65ch de largura de texto.

- **Margins/Padding:**

- Global horizontal padding: aplicar `px-4` (1rem) em mobile container e aumentar para `px-8` em md, `px-16` em xl, etc., para que haja margens laterais.
- Vertical spacing: Titles from content, etc., consistent. For instance:
- Section headings (like category title from list) might have margin-bottom 1rem.
- Space between paragraphs default in Tailwind's `prose` is fine or manually set `mb-4` on p.
- Cards grid row-gap and column-gap around 1rem (4).
- Footer padding: give it `py-8` and `px-4` etc.

- **Borders and dividers:** The site uses minimal borders; mostly separation via whitespace. Perhaps some thin horizontal lines between sections or grey lines under header or above footer. The HR or the * * * we handle.

- **Shadows:** Possibly used subtly on cards or header. Could implement a slight drop shadow on header if fixed, to make it stand out above content when scrolling.

Em geral, o estilo deve transmitir **seriedade e clareza**, refletindo o jornalismo combativo do Intercept. Vamos relembrar pontos-chave: - Preto, branco e um verde fluorescente como identidade (podemos ajustar o tom para acessibilidade). - Tipografia forte nos títulos, fácil de ler no corpo. - Layouts are spacious, not cluttered; let content breathe. - Consistência: todos os componentes compartilham o mesmo look&feel (botões com mesmo estilo, todos cards semelhantes, etc.).

Funcionalidades e Recursos Interativos

Por fim, enumeramos as funcionalidades que precisam ser implementadas para atingir equivalência funcional:

1. **Sistema de Roteamento de Páginas:** Usaremos Next.js para gerar páginas estáticas para todas as seções mencionadas. Organizar rotas conforme estrutura:
2. `/` - Home.
3. `/[categoria]` - para Poder, Direitos etc. Podemos usar file routes ou um route dinâmico com um mapa das categorias válidas.
4. `/materias` - arquivo geral.
5. `/especiais` - lista de séries.
6. `/especiais/[serie]` - página de série (dinâmica de acordo com slug da série).
7. `/vozes` - página das colunas.
8. `/videos` - página de vídeos.
9. `/sobre` - (ou `/quem-somos`) - página quem somos.
10. `/fontes` - página seja nossa fonte.

11. `/newsletter` – página de newsletter.
12. `/faq` – perguntas frequentes.
13. `/privacidade` – política de priv.
14. `/termos` – termos de uso.
15. `/equipe/[autor]` – perfil de autor (slug do nome).
16. `/buscar` – página de resultados de busca (pode ser implementada com query param e um componente, ou como route do Next 13 com searchParams).
17. `/[ano]/[mes]/[dia]/[slug]` – Notar: o site original em WordPress tem URL com data (ano/mês/dia) e slug do post. Ex.: `/2025/09/09/titulo-da-materia/`. Podemos seguir o mesmo padrão de URL para posts, criando uma estrutura dinâmica de rotas aninhadas (um pouco complexo mas possível com catch-all routes em Next). Entretanto, para simplicidade, podemos roteá-los como `/post/[slug]` ou apenas `/[slug]` se não conflitar com outras rotas. Como replicação fiel, usar datas no URL seria interessante. Em Next 13 app router, podemos fazer a pasta `app/[year]/[month]/[day]/[slug]/page.tsx` que capturaria. Fica avançado mas factível. Em alternativa, ignorar a data no URL e colocar posts em `/posts/[slug]` ou similar, a não ser que a exigência seja idêntica – o enunciado fala em cópia visual e funcional, mas URLs não precisam ser iguais, apenas acessos. Podemos escolher a via fácil e não usar data nos paths.

Configurar adequadamente o `<Link>` para navegar internamente no Next, e usar SSR/SG para pré-carregar.

1. **Dados fictícios:** Criar arrays/objetos JavaScript que armazenem:
 2. Lista de categorias e seus posts (ou uma lista global de posts com campo category).
 3. Lista de séries e quais posts pertencem.
 4. Lista de colunistas e seus posts.
 5. Assim podemos preencher as páginas. Talvez centralizar num único arquivo `data.js` para fácil edição.
 6. Para textos longos (conteúdo de posts, bio de autores, etc.), escrever alguns parágrafos manualmente (inspirados mas não idênticos aos originais) ou usar lorem ipsum contextualizado. É melhor criar 2-3 artigos de exemplo com conteúdo verossímil para demonstrar.
7. **Imagens:** usar URLs de placeholders (via unsplash source or local placeholders). Ex.: usar uma mesma imagem de política para todos posts de política, etc., até para simplicidade.
8. **Pesquisar (funcionalidade):** Implementar o campo de busca no header:
 9. Ao submeter, redirecionar para `/buscar?q=termo` (no Next, use router push with query).
 10. Na page `/buscar`, acessar query e filtrar a lista de posts criados. Exibir resultados com mesmo componente de lista de cards.
 11. Destacar a frase “Você pesquisou por *termo*” no topo da página ⁶⁰.
 12. Se nenhum resultado, mostrar mensagem tipo “Nenhum conteúdo encontrado para ‘termo’”.
 13. Tipicamente, a busca do Intercept atingiria título e corpo. Podemos limitar a título e talvez autor pra demonstrar.
14. **Inscrição na Newsletter:**
 15. Página Newsletter: form de inscrição. Não é necessário implementar chamada real de API. Pode-se usar um serviço de email marketing se quisesse, mas não no escopo. Apenas simular UI: ao clicar Enviar, substituir o form pela mensagem de sucesso (ou exibir abaixo).

16. *Double opt-in simulation*: O texto "Já se inscreveu? Confirme seu email para continuar lendo" aparece no gating ⁸⁸. Podemos ignorar double opt-in e assumir inscrição imediata.
17. Gating Modal: logic to count article views. This can be done via a simple counter in localStorage incremented each time a post page loads (use Next.js useEffect to increment). If count > 1 and not subscribed, show modal. Add a close/"Continuar Lendo" anyway to bypass if needed (the original appears to allow skipping after showing the prompt, by confirming email or maybe just one free article).
18. This is advanced, but we document it as part of full replication.
19. **Doação (Apoiar)**: O botão leva a uma página externa (no original, paybox.doare.org link) ⁶⁶. No nosso, podemos redirecionar para um dummy page "/doar" que agradece, ou simplesmente indicar target blank to an external placeholder. Implementar a página "Apoie" não foi solicitado, só garantir que o CTA exista.
20. **Rede sociais links**: Os ícones no header/footer devem linkar para as respectivas páginas do Intercept ou podemos redirecionar para nossas fictícias. Talvez melhor usar as do Intercept (não há problema em linkar externamente). Ex.: Instagram -> `https://www.instagram.com/theinterceptbrasil`. Ou colocar # no href só para demo.
21. Para Bluesky, link fornecido (bsky.app).
22. Para WhatsApp, o link dado leva a entrar num canal de notícias deles ⁷². Não replicaremos a funcionalidade do canal, mas podemos apontar para `https://wa.me/SEUNUMERO` fictício.
23. Certifique-se de usar ícones com alt text (aria-label) para acessibilidade.
24. **Vídeos**: Em página de vídeos, se decidirmos implementar players:
25. Podemos integrar vídeos do YouTube (ex.: se Intercept Brasil tem canal, incorporar alguns vídeos por embed iframe).
26. Ou usar um player HTML5 para vídeos locais. Simpler: just link to YT or use an <iframe> with a sample video. Put a play icon overlay on thumbnails maybe.
27. If not, at least show thumbnail and on click open an external link to YT.
28. **Performance considerations**: Use Next Image for optimized images. Use <Head> to set meta tags for SEO (title, description) for each page (you can craft these based on content).
29. Eg: Title tag of an article page = "Título da matéria – Intercept Brasil".
30. Include a meta description (maybe first sentence).
31. For multi-language, although site is pt-BR only, ensure lang attribute in html.
32. **Testing and QA**: Once built, manually test each link in the menu, each component in different screen sizes. Ensure no broken link. For example, clicking "Todos os Especiais" goes to the series index page listing series ⁷⁰; clicking a series goes to that series page.

Prompt de Desenvolvimento Passo-a-Passo

Agora, com todo esse levantamento, vamos compor um **prompt técnico detalhado**, passo a passo, que oriente o desenvolvedor a implementar o projeto. Incluiremos sugestões de estrutura de código, nomes de componentes e pastas:

Passo 1: Configuração do Projeto e Stack

- Inicialize um novo projeto Next.js (versão 13+). Use o TypeScript para maior segurança no desenvolvimento.
- Instale e configure o **Tailwind CSS** no projeto (via `postcss` e `autoprefixer`). Garanta que as configurações de conteúdo englobem os arquivos do app.
- Instale o pacote **shadcn/ui** (ou Radix UI + shadcn preset):
- Siga a documentação para adicionar o plugin shadcn. Isso facilitará adicionar componentes pré-estilizados (como Accordion, Dialog, etc.).
- Após configurado, gere os componentes necessários via CLI do shadcn (por exemplo: `npx shadcn-ui add accordion` para FAQ, `add dialog` para modal, `add sheet` para menu mobile, `add dropdown-menu` para menu desktop, `add toast` se quiser notificação, etc.).
- Configure fontes globais no `tailwind.config.js` ou via CSS:
- Importe as fontes escolhidas do Google Fonts no `<head>` do `_app` ou via CSS `@import`.
- Defina classes utilitárias ou use plugin `tailwind-typography` para estilos de texto base (pode ser útil para artigos: aplique classe `prose prose-lg` no container de artigo para estilização rápida).
- Configure paleta de cores no Tailwind:
- Adicione em `theme.extend.colors` as cores: por exemplo, `primary: '#00b050'`, `dark: '#000000'`, `light: '#ffffff'`, etc., para uso consistente.
- Configure a plugin `tailwindcss/forms` para estilizar inputs e checkboxes (útil para newsletter form).

Passo 2: Estrutura de Pastas e Arquivos (Next.js App Router)

Crie a seguinte estrutura dentro de `app/` (considerando uso do App Router):

```
app/
├── layout.tsx           // layout root com Header e Footer comuns
├── globals.css          // estilos globais Tailwind (importado aqui)
├── page.tsx             // Página Home
├── [categoria]/
│   └── page.tsx         // Página dinâmica para categorias (Poder, etc.)
├── materias/
│   └── page.tsx         // Página "Acesse mais matérias" (arquivo geral)
├── especiais/
│   ├── page.tsx        // Página lista de todos os especiais
│   └── [serie]/
│       └── page.tsx     // Página de série específica (dinâmico por slug)
├── vozes/
│   └── page.tsx        // Página "Todas as vozes"
└── videos/
```

```

├── page.tsx      // Página Vídeos
├── newsletter/
│   └── page.tsx  // Página Newsletter
├── fontes/
│   └── page.tsx  // Página "Seja nossa fonte"
├── sobre/        // ou "quem-somos"
│   └── page.tsx  // Página Quem Somos (Sobre)
├── faq/
│   └── page.tsx  // Perguntas Frequentes
├── privacidade/
│   └── page.tsx  // Política de Privacidade
├── termos/
│   └── page.tsx  // Termos de Uso
├── equipe/
│   └── [autor]/
│       └── page.tsx // Perfil de membro/autor
├── buscar/
│   └── page.tsx   // Página de resultados de busca (opção: usar
searchParams)
└── (opcional: posts/[slug]/page.tsx caso prefira posts sem data no URL)

```

(Obs: Caso opte pelo formato de URL com datas para posts, crie `app/[year]/[month]/[day]/[slug]/page.tsx`. Dentro, use os params para buscar o post correto do conteúdo. Isso é avançado; você pode primeiro implementar sem data e depois adicionar se houver tempo.)

- Dentro de cada page, você usará dados simulados para renderizar conteúdo. Por exemplo, em `[categoria]/page.tsx`, extrair o parâmetro `categoria` de `params` e filtrar a lista de posts global para aqueles que têm categoria = param, então renderizar.
- Use **generateStaticParams** se for App Router para pré-gerar páginas para cada categoria existente (Poder, Direitos, etc.), cada série slug, cada autor slug. Isso melhora performance e evita 404.
- Crie um arquivo de **dados** (p.ex. `data/siteData.ts`) exportando objetos:
- `categories = ['poder', 'direitos', ...]`
- `posts = [{ slug, title, date, category, authors, excerpt, content, series?, format?}, ...]`
- `series = [{ slug, title, description, posts: [postId,...] }, ...]`
- `authors = [{ slug, name, role, bio, twitter, email, photo, posts: [...]}, ...]` Use esses dados nas páginas correspondentes.

Passo 3: Desenvolvimento dos Componentes Reutilizáveis

Crie uma pasta `components/` para colocar os componentes React isolados. Alguns componentes a implementar:

- `Header.tsx`: Componente para o cabeçalho do site.
- Inclui o logo (pode ser um `<Link href="/">` com `Intercept Brasil` estilizado como logo).
- Inclui o botão "Quero Apoiar": `<Link href="/doar" className="btn-primary">QUERO APOIAR</Link>` (use classes Tailwind para btn-primary).

- Incluir o menu de navegação principal. Pode usar `<nav>` com lista ``.
 - Em desktop: pode mapear uma array de menu items. Itens sem submenu (Newsletter, Poder, etc.) renderizam `<Link>`. Itens com submenu (Especiais, Vozes) renderizam um **DropdownMenu** (shadcn) contendo os sublinks.
 - Utilize shadcn/ui `DropdownMenu` components: e.g., for "Especiais":

```
<DropdownMenu>
  <DropdownMenuTrigger className="nav-item">Especiais</
DropdownMenuTrigger>
  <DropdownMenuContent>
    {specialSeries.map(serie => (
      <DropdownMenuItem key={serie.slug}>
        <Link href={`/${especiais}/${serie.slug}`}>{serie.title}</
Link>
      </DropdownMenuItem>
    ))}
    <DropdownMenuSeparator />
    <DropdownMenuItem>
      <Link href="/especiais">Todos os Especiais</Link>
    </DropdownMenuItem>
  </DropdownMenuContent>
</DropdownMenu>
```

(Ou colocar "Todos os Especiais" primeiro, conforme achar melhor).

- Similar para "Vozes" com subitens autores principais + "Todas as vozes".
- Em mobile: Renderize um botão hambúrguer visível (`md:hidden` Tailwind) que ao clicar abre o off-canvas menu.
- Use shadcn/ui **Sheet**:

```
const [open, setOpen] = useState(false);
<Sheet open={open} onOpenChange={setOpen}>
  <SheetTrigger><MenuIcon /></SheetTrigger>
  <SheetContent side="left" className="w-[250px] p-4">
    <SheetHeader>
      <SheetTitle>Menu</SheetTitle>
    </SheetHeader>

    {/* Inside content, place navigation links similar to desktop but
    in vertical list */}
    <ul className="flex flex-col gap-2">
      <li><Link href="/newsletter">Newsletter</Link></li>
      <li>
        {/* For Especiais with submenu: maybe handle manually */}
        {showEspecialSub ? (
          <>
            <button onClick={() => setShowEspecialSub(false)}
            >Voltar</button>
            <ul>{...specialSeries links}</ul>
          </>
        ) : null}
      </li>
    </ul>
  </SheetContent>
</Sheet>
```

```

        ) : (
          <button onClick={() => setShowEspecialSub(true)}
>Especiais</button>
        )}
      </li>
      { /* other items */ }
    </ul>
  </SheetContent>
</Sheet>

```

Ou, mais fácil: use **Accordion** for collapsible submenus:

```

<Accordion type="single" collapsible>
  <AccordionItem value="especiais">
    <AccordionTrigger>Especiais</AccordionTrigger>
    <AccordionContent>
      <ul>{specialSeries.map(...)}<li><Link href="/
especiais">Todos os Especiais</Link></li></ul>
    </AccordionContent>
  </AccordionItem>
  ... // Vozes similarly
</Accordion>

```

Isso evita gerenciar estado manual de voltar. Ajuste estilo do Trigger para parecer um item de menu.

- In the sheet, also include the donate button and search icon if desired, or they can remain outside.
- **Search bar/icon:** Place a search icon (magnifier) in header (perhaps next to donate or at far right).
- On desktop: could be just an icon that toggles a search input overlay. On click, open a **Dialog** with a search field.
- Simpler: have an input hidden by default (`hidden md:block` toggling etc.). But better UX: implement a *command palette* style search.
- Using shadcn **Command** component is an option: It gives a palette that can fuzzy search through items. But maybe overkill. Instead:
 - Implement `SearchDialog`: using `Dialog` from shadcn. In header:

```

const [searchOpen, setSearchOpen] = useState(false);
<button onClick={() => setSearchOpen(true)} aria-
label="Buscar"><MagnifyingGlassIcon /></button>
<Dialog open={searchOpen} onOpenChange={setSearchOpen}>
  <DialogContent>
    <input type="text" placeholder="Buscar..." className="w-
full border-b"
      onKeyDown={e => { if(e.key==='Enter'){
router.push('/buscar?q='+e.target.value) }}} />
    { /* Optionally, list suggestions or top results */ }
  </DialogContent>
</Dialog>

```


Style the DialogContent to perhaps full-screen (by overriding styles or using a large size).

- Or use a separate page for search with an actual input (less dynamic).
- On mobile: the same approach; maybe directly navigate to `/buscar` page when clicking search (which could open the keyboard immediately if `<input autoFocus>` on that page).
- `Footer.tsx`: Contains the structured footer.
- Use a `<footer className="bg-black text-gray-100 py-8">` (assuming dark footer).
- Inside, a container `max-w-screen-xl mx-auto px-4 flex flex-col md:flex-row justify-between`.
- Column 1:

```
<div className="mb-6 md:mb-0">
  <h4 className="font-bold">Seções</h4>
  <ul className="mt-2 space-y-1">
    <li><Link href="/materias">Matérias</Link></li>
    <li><Link href="/vozes">Vozes</Link></li>
    <li><Link href="/videos">Vídeos</Link></li>
    <li><Link href="/newsletter">Newsletter</Link></li>
    <li><Link href="/fontes">Seja nossa fonte</Link></li>
  </ul>
</div>
```

- Column 2:

```
<div>
  <h4 className="font-bold">Sobre</h4>
  <ul className="mt-2 space-y-1">
    <li><Link href="/sobre">Quem Somos</Link></li>
    <li><Link href="/faq">Perguntas Frequentes</Link></li>
    <li><Link href="/privacidade">Política de Privacidade</Link></li>
    <li><Link href="/termos">Termos de Uso</Link></li>
  </ul>
</div>
```

- Below or aside these columns, list social media icons horizontally:

```
<div className="mt-6 md:mt-0 flex space-x-4">
  <a href="https://instagram.com/theinterceptbrasil" target="_blank"
    rel="noopener"><InstagramIcon /></a>
  ... other icons
</div>
```

Use appropriate SVG icons (you can import from Heroicons or use icon libraries).

- At bottom (maybe as separate row or beneath columns):

```
<div className="mt-4 text-sm text-gray-400">
  ©2023 Intercept Brasil. Todos os direitos reservados.
</div>
```

(You can keep dynamic year with `new Date().getFullYear()`).

- Ensure the footer is responsive: in mobile, columns stack (we used `flex-col` for mobile above, and `row` for md up).
- `ArticleCard.tsx`: component to display an article preview card.
- Props: article object (title, slug, image, date, authors, excerpt, seriesLabel?, format?).
- Return:

```
<div className="group border-b pb-4 mb-4 last:mb-0 last:pb-0"> { /* or
use card style */}
  {article.image &&
    <Link href={article.url}>
      <Image src={article.image} alt={article.title} className="w-full
h-auto mb-2" />
    </Link>
  }
  <div>
    { /* Label/tag if exists */}
    {article.series && <span className="text-xs font-bold text-white bg-
black px-1 mr-2">{article.series}</span>}
    {article.format === 'video' && <span className="text-xs font-bold
text-white bg-green-600 px-1 mr-2">VÍDEO</span>}
  </div>
  <h2 className="text-xl font-bold leading-snug">
    <Link href={article.url} className="hover:underline">{article.title}
  </Link>
  </h2>
  {article.excerpt && <p className="text-sm text-gray-700
mt-1">{article.excerpt}</p>}
  <div className="text-xs text-gray-600 mt-1">
    {article.dateFormatted} {article.authors.join(', ')}
  </div>
</div>
```

- Style adjustments: perhaps on hover, `.group:hover h2 { color: primary; }` to give feedback.
- We use a bottom border to separate items (like a list style).
- Could also choose a card style with shadow and space between, but the original likely lists them with subtle separators.
- `FeaturedArticleCard.tsx`: for the homepage top story. Could extend `ArticleCard` with larger image and text.

- E.g., a variant prop or separate component.
- Maybe layout side-by-side or fullwidth with overlay text.
- Considering the HTML, top of homepage had three images and two stories side by side ¹. Possibly they had one large image left (covering first story), and the second story with a smaller image to the right above its title. This is speculative.
- Simpler: one big featured story (full width image) and title over or under it, then one secondary below or aside.
- Implement as needed: e.g.,

```
<div className="lg:flex lg:space-x-6 mb-8">
  <div className="lg:w-2/3">
    <Image src={feat1.image} ... className="w-full h-auto" />
    <span className="tag">{feat1.series}</span>
    <h2 className="text-3xl font-bold mt-2"><Link href={feat1.url}>
    >{feat1.title}</Link></h2>
    <p>{feat1.excerpt}</p>
  </div>
  <div className="lg:w-1/3 lg:flex lg:flex-col lg:justify-between">
    <div>
      <Image src={feat2.image} ... className="w-full h-auto" />
      <h3 className="text-2xl font-bold mt-2"><Link href={feat2.url}>
      >{feat2.title}</Link></h3>
      <p>{feat2.excerpt}</p>
    </div>
  </div>
</div>
```

Ensure responsiveness (stack on small screens).

- `ShareBar.tsx`: for article page share icons.
- Props: article URL, title (for constructing share text).
- Return a div with icons:

```
<div className="flex space-x-3 text-gray-600 text-lg">
  <a href={`https://api.whatsapp.com/send?text=${
    encodeURIComponent(article.title + ' ' + article.url)}}`
  target="_blank" aria-label="Share on WhatsApp"><WhatsAppIcon /></a>
  <a href={`https://www.facebook.com/sharer.php?u=${article.url}`}
  target="_blank" aria-label="Share on Facebook"><FacebookIcon /></a>
  <a href={`https://twitter.com/share?url=${article.url}&text=${
    encodeURIComponent(article.title)}}` target="_blank" aria-label="Share
  on Twitter"><TwitterIcon /></a>
  </* Use Twitter as Bluesky alt */>
  <button onClick={copyLinkToClipboard} aria-label="Copy
  link"><LinkIcon /></button>
</div>
```

Possibly incorporate share count if wanted (not required). Use small icon components (FontAwesome or Heroicons). This component can be placed in the article header and footer.

- `NewsletterForm.tsx`: could be used both on newsletter page and gating modal.
- Props: `onSuccess` callback (maybe to show message).
- Renders an `<form>` with email input and checkbox for terms:

```
<form onSubmit={handleSubscribe} className="mt-4">
  <div>
    <input type="email" required placeholder="Seu email"
className="border p-2 w-full"/>
  </div>
  <label className="text-xs text-gray-700 block mt-2">
    <input type="checkbox" required className="mr-1"/>
    Aceito receber e-mails e concordo com a <Link href="/
privacidade">Política de Privacidade</Link> e os <Link href="/
termos">Termos de Uso</Link>.
  </label>
  <button type="submit" className="mt-3 px-4 py-2 bg-primary text-white
font-bold">Enviar</button>
  {status === 'success' && <p className="text-green-600 mt-2">Cadastro
enviado com sucesso!</p>}
</form>
```

Simule o `handleSubscribe` com a `onSuccess` (`setStatus success` etc.). For gating, `onSuccess` can close modal and mark user as subscribed.

- `AccordionItem` and `AccordionTrigger` for FAQ: We use shadcn's ready Accordion in the page, but can create a small wrapper if needed to map Q&A easily.
- `CookieConsent.tsx`: a small bar component at bottom:

```
{showConsent &&
  <div className="fixed bottom-0 inset-x-0 bg-gray-900 text-gray-100
text-sm p-4 flex justify-between items-center">
    <span>Utilizamos cookies... <Link href="/privacidade">Política de
Privacidade</Link>.</span>
    <button onClick={acceptCookies} className="bg-gray-700 px-3 py-1
rounded">Ok</button>
  </div>
}
```

Set state if accepted in `localStorage` to not show next time.

After building these components, remember to import and use them in the appropriate pages.

Passo 4: Construção das Páginas usando os Componentes

Agora, monte cada página conforme o layout planejado, reusando componentes:

- `app/layout.tsx`: Este é o layout global. Deve incluir o `<Header />` no topo e `<Footer />` no final, e um `<main>{children}</main>` no meio para o conteúdo da página atual. Também incluir o `CookieConsent` component at bottom.
- Importe os estilos globais (Tailwind base, etc.) aqui ou em `globals.css`.
- Exemplo:

```
import Header from '@components/Header';
import Footer from '@components/Footer';
import '../styles/globals.css';
export default function RootLayout({ children }) {
  return (
    <html lang="pt-BR">
      <head />
      <body className="min-h-screen flex flex-col">
        <Header />
        <main className="flex-1 bg-white">{children}</main>
        <Footer />
        <CookieConsent />
      </body>
    </html>
  );
}
```

(Using flex-col to push footer to bottom if content short.)

- **Homepage (`app/page.tsx`):**
- Aqui, importe os dados de posts e escolha quais serão destaques. Por exemplo, pegar os dois ou três mais recentes como destaque(s).
- Estruturar:

```
import { featuredPosts, latestPosts } from '@data/siteData';
import FeaturedArticleCard from '@components/FeaturedArticleCard';
import ArticleCard from '@components/ArticleCard';

export default function HomePage() {
  return (
    <div className="container mx-auto px-4 py-6">
      { /* Destaques */ }
      <section>
        <div className="grid md:grid-cols-2 gap-6">
          { /* Supondo dois destaques */ }
          {featuredPosts.map(post => (
```

```

        <FeaturedArticleCard key={post.slug} article={post} />
      )})
    </div>
  </section>

  {/* Lista de últimas matérias */}
  <section className="mt-8">
    <h2 className="sr-only">Últimas notícias</h2>
    <div className="grid md:grid-cols-2 gap-8">
      {latestPosts.map(post => (
        <ArticleCard key={post.slug} article={post} />
      ))}
    </div>
    <div className="text-center mt-6">
      <Link href="/materias" className="text-primary font-bold
hover:underline">ACESSE MAIS MATÉRIAS</Link>
    </div>
  </section>
</div>
);
}

```

- Use sr-only para o heading da seção se não quiser mostrar "Últimas notícias" textualmente.
- Os featured posts podem ser um ou dois; adapt layout accordingly (maybe not even use grid if one, etc.).
- O link "Acesse mais matérias" estilizado como no original ⁸⁹.

• **Página de Categoria** (`app/[categoria]/page.tsx`):

• Pegar o param categoria. Validar se existe no data. Se não, pode fallback 404.

• Renderizar:

```

export async function generateStaticParams() {
  return categories.map(cat => ({ categoria: cat.slug }));
}

export default function CategoryPage({ params }) {
  const { categoria } = params;
  const category = categories.find(c => c.slug === categoria);
  const posts = allPosts.filter(p => p.category === categoria);
  if (!category) notFound(); // handle invalid

  return (
    <div className="container mx-auto px-4 py-6">
      <h1 className="text-3xl font-bold mb-4
capitalize">{category.title}</h1>
      {/* share icons perhaps */}
      <ShareBar url={...} title={`Categoria ${category.title}`} />
      <div className="grid md:grid-cols-2 gap-8 mt-6">

```

```

    {posts.map(post => <ArticleCard key={post.slug} article={post} /
>)}
  </div>
  { /* Pagination */ }
  <Pagination currentPage={1} totalPages={Math.ceil(posts.length /
perPage)} baseUrl={`/${categoria}`} />
</div>
);
}

```

- Use `capitalize` to uppercase first letter of category if needed (since slug might be lowercase).
- Include share icons if needed (maybe not necessary on category page, but original HTML did show share for category ³).
- Implement `<Pagination>` logic: If we want static multi-page, either use catch-all route `[page]`. But simpler: if content set small, skip actual page nav. But since pediram, poderia ter `app/[categoria]/page/[pageNumber]/page.tsx` – gets complicated. Alternatively do client-side pagination with a Load More button. Given time, can skip actual multiple pages, or generate a couple for show.
- For fidelity, you might generate a few pages with fewer items each (like 10 per page).

• **Página “Matérias” (arquivo geral) (`app/materias/page.tsx`):**

- Muito similar à categoria, mas sem filtrar – lista todos os posts (ou muitos deles).
- Estrutura basicamente idêntica, apenas H1 = “Matérias”.
- Pode usar a mesma grid e pagination.

• **Página “Todos os Especiais” (`app/especiais/page.tsx`):**

- Exibe lista de séries especiais.
- Dados: use array `series` (cada com título, descrição).
- Layout:

```

<div className="container mx-auto px-4 py-6">
  <h1 className="text-3xl font-bold mb-4">Especiais</h1>
  <div className="grid md:grid-cols-2 gap-8">
    {series.map(serie => (
      <div key={serie.slug} className="border-b pb-4 mb-4">
        <h2 className="text-xl font-bold">
          <Link href={`/${especiais}/${serie.slug}`}
className="hover:underline">{serie.title}</Link>
        </h2>
        <p className="text-sm text-gray-800 mt-1">{serie.description}</
p>
      </div>
    ))}
  </div>
  { /* If many series, maybe pagination here too (the original had 2

```

```
pages for series index) 90 */}
</div>
```

- Estilizar similar aos cards, mas aqui não há imagem, só texto descritivo. Você pode adicionar uma imagem representativa se quiser (serie.image).

• **Página de Série** (`app/especiais/[serie]/page.tsx`):

- Pegar o slug da série, encontrar o objeto da série.

- Layout:

```
const serie = series.find(s => s.slug === params.serie);
const posts = allPosts.filter(p => p.series === serie.title);
return (
  <div className="container mx-auto px-4 py-6">
    {/** Cover image and label */}
    {serie.coverImage && <Image src={serie.coverImage}
alt={serie.title} className="w-full h-auto mb-4" />}
    <div className="text-sm text-primary font-bold uppercase">Especial</div>
    <h1 className="text-3xl font-bold mb-2">{serie.title}</h1>
    <p className="text-lg text-gray-800 mb-6">{serie.description}</p>
    <ShareBar url={currentURL} title={serie.title} />
    <div className="mt-6 grid md:grid-cols-2 gap-8">
      {posts.map(post => <ArticleCard key={post.slug} article={post} /
    >)}
    </div>
  </div>
);
```

- O label "Especial" acima do título 10 .
- Listagem das matérias dessa série (com prefixo do nome da série se quisermos, mas como já estamos na página da série, podemos omitir repetição ou deixá-la para estilo).
- Se a série tiver muitas partes, incluir paginação.
- Também poderíamos indicar "Parte X" se essa informação existir (o original mostrava "Parte 12" etc. Isso seria meta de cada post ou do serie listing?). No snippet, "Parte 12" aparecia dentro do artigo, não na lista 24 . Então não replicar aqui.

• **Página "Vozes"** (`app/vozes/page.tsx`):

- Temos colunistas principais (vozes fixas) e as colunas em geral.
- Suponha que temos uma lista `mainColumnists` (Fabiana, João, etc.) e `allPosts` filtered by type "Opinião" for all columns.
- Layout:

```
<div className="container mx-auto px-4 py-6">
  <h1 className="text-3xl font-bold mb-4">Vozes</h1>
```



```

<h2 className="text-2xl font-semibold mb-3">Colunistas</h2>
<div className="grid md:grid-cols-2 lg:grid-cols-3 gap-6">
  {mainColumnists.map(col => (
    <div key={col.slug} className="border-b pb-4">
      {/* If we have col.photo */}
      <Link href={`\equipe/${col.slug}`}>
        <Image src={col.photo} alt={col.name} className="w-16 h-16
rounded-full mb-2" />
      </Link>
      <h3 className="text-xl font-bold">
        <Link href={`\equipe/${col.slug}`}
className="hover:underline">{col.name}</Link>
      </h3>
      {/* Last article of that columnist */}
      {col.lastPost && (
        <p className="text-sm">
          <Link href={col.lastPost.url}
className="hover:underline">{col.lastPost.title}</Link>
          <span className="text-gray-600"> 📅
{formatDate(col.lastPost.date)}</span>
        </p>
      )}
    </div>
  )})}
</div>

<h2 className="text-2xl font-semibold mt-8 mb-3">Mais Vozes</h2>
<div className="grid md:grid-cols-2 gap-8">
  {voicePosts.map(post => <ArticleCard key={post.slug}
article={post} />)}
</div>
</div>

```

- Assim exibimos os colunistas principais com nome e link para perfil, e a chamada da última coluna (como no original: “Fabiana Moraes – [título] 19 de agosto” etc.) ¹².
- Depois listamos todas as colunas (o original parece listar inclusive repetindo algumas já listadas acima, mas está ok).
- Poder adicionar paginação se muitas colunas.

• **Página de Vídeos** (`app/videos/page.tsx`):

- Pegar posts do tipo vídeo.

- Layout:

```

<div className="container mx-auto px-4 py-6">
  <h1 className="text-3xl font-bold mb-4">Vídeos</h1>
  {/* Featured video (the latest maybe) */}
  {videoPosts.length > 0 && (
    <div className="mb-8">

```

```

    <div className="relative pb-[56.25%] h-0">
      <iframe src={videoPosts[0].embedUrl} frameBorder="0"
allowFullScreen
        className="absolute top-0 left-0 w-full h-full"></
iframe>
    </div>
    <h2 className="text-2xl font-bold mt-2"><Link
href={videoPosts[0].url}>{videoPosts[0].title}</Link></h2>
    <p className="text-sm text-gray-700">{videoPosts[0].excerpt}</p>
  </div>
)}
<div className="grid md:grid-cols-2 gap-8">
  {videoPosts.slice(1).map(post => (
    <div key={post.slug}>
      <Image src={post.thumb} alt={post.title} className="w-full h-
auto" />
      <h3 className="text-xl font-bold mt-2"><Link href={post.url}
>{post.title}</Link></h3>
      <div className="text-xs text-gray-600">{formatDate(post.date)}
❑ {post.authors.join(', ')}</div>
    </div>
  ))}
</div>
<Pagination ... />
</div>

```

- O exemplo acima incorpora o vídeo mais recente via iframe (usando embedUrl in data).
- Os demais listados como thumbnails com link para a página ou modal. O Intercept possivelmente abre vídeos em página própria com contexto ou simplesmente como posts com embed (no data above, we treat video post as a variant of article page containing an iframe).
- Dependendo do volume, adicionar paginação.

• **Página de Artigo (Post)** (`app/[slug]/page.tsx` **ou similar**):

• Se optou por `[year]/[month]/[day]/[slug]`, extrair do params para achar o post.

• Aqui montamos a estrutura de leitura:

```

const post = allPosts.find(p => p.slug === slug);
return (
  <div className="container mx-auto px-4 py-6 max-w-prose">
    </* Kicker if exists */>
    {post.kicker && <div className="text-sm font-bold text-primary
mb-1">{post.kicker}</div>}
    <h1 className="text-4xl font-bold mb-2">{post.title}</h1>
    </* Share bar top */>
    <div className="flex items-center space-x-3 text-gray-600 mb-4">
      <ShareBar url={currentURL} title={post.title} />
      <div className="ml-auto text-sm">

```

```

        {post.authors.map(a => (
          <span key={a}><Link href={` /equipe/${getAuthorSlug(a)} `}>{a}
</Link></span>
        ))}
        <span className="mx-1">•</span>
        <time>{formatDate(post.date, { verbose: true })}</time>
      </div>
    </div>
    { /* If part of a series, show series info box */ }
    {post.series &&
      <div className="bg-gray-100 border-l-4 border-primary p-4 mb-4">
        <div className="text-primary font-bold uppercase text-sm">{post.series}</div>
        {post.seriesPart && <div className="font-bold text-lg">Parte
{post.seriesPart}</div>}
        <p className="text-sm text-gray-800">{getSeriesDescription(post.series)}</p>
        <Link href={` /especiais/${seriesSlugFromTitle(post.series)} `}
className="text-primary font-semibold mt-2 inline-block">Confira o
especial completo</Link>
      </div>
    }
    { /* Article content */ }
    <article className="prose prose-gray max-w-none">
      { /* If image cover */ }
      {post.image && <Image src={post.image} alt={post.title}
className="w-full h-auto mb-4" />}
      {post.contentParagraphs.map((para, idx) => {
        if (para.type === 'text') return <p key={idx}>{para.text}</p>;
        if (para.type === 'subtitle') return <h2 key={idx}>{para.text}
</h2>;
        if (para.type === 'image') return (
          <figure key={idx} className="my-4">
            <Image src={para.src} alt={para.alt} className="w-full h-auto" />
            {para.caption && <figcaption className="text-sm text-gray-600 text-center mt-1">{para.caption}</figcaption>}
          </figure>
        );
        if (para.type === 'quote') return <blockquote key={idx}
>{para.text}</blockquote>;
      })}
    </article>
    { /* Donation call-to-action */ }
    <div className="bg-gray-100 p-4 my-8 text-center">
      <p className="mb-2 text-gray-800">O Intercept é sustentado por
quem mais se beneficia do nosso jornalismo: o público.</p>
      <Link href="/doar"
className="bg-primary text-white font-bold py-2 px-4 rounded">Apoie o
Intercept Hoje</Link>
    </div>

```

```

    { /* Related topics tags */
    {post.category && (
      <div className="text-sm mb-4">TEMAS RELACIONADOS: <Link href={`/${$
{post.category}`} className="underline">{capitalize(post.category)}</
Link></div>
    )}
    { /* Author contacts */
    <div className="border-t pt-4 mb-8">
      {post.authors.map(authorName => {
        const author = authors.find(a => a.name === authorName);
        return author ? (
          <div key={author.slug} className="mb-4">
            <h4 className="font-bold">{author.name}</h4>
            <p className="text-sm text-gray-700">{author.role}</p>
            <p className="text-sm">{author.bioShort}</p>
            <p className="text-sm mt-1">
              {author.email && <a href={`mailto:${author.email}`}
className="mr-4 underline">Email</a>
              {author.twitter && <a href={`https://twitter.com/$
{author.twitter}`} className="underline">@{author.twitter}</a>
            </p>
          </div>
        ) : null;
      )}
    </div>
    { /* Recent Articles */
    <div className="border-t pt-4">
      <h3 className="font-bold mb-2">Artigos recentes</h3>
      <div className="grid md:grid-cols-2 gap-4">
        {recentPosts.slice(0,4).map(rp => (
          <ArticleCard key={rp.slug} article={rp} />
        ))}
      </div>
    </div>
    { /* Newsletter gating modal (conditionally render) */
    {!userSubscribed && <NewsletterModal onSubscribeSuccess={...} />}
    { /* Share bar bottom */
    <div className="mt-8">
      <ShareBar url={currentURL} title={post.title} />
    </div>
  </div>
);

```

- Bastante coisa, mas replicamos todos os elementos:
- Kicker, Title, ShareBar, Author&Date, SeriesBox, Main content, CTA donate, Related category, Author contact, Recent posts, Gating, and bottom share.
- Observação: definimos `prose prose-gray` (se Tailwind Typography plugin installed) para estilo do artigo (isso automaticamente estiliza headers, p, blockquote etc.). Precisariamos talvez ajustar para não afetar manual figcaption styles. Podemos custom via config or simply style manually as above.

- `NewsletterModal`: a componente de gating discutido. Rendeirizá-lo condicionalmente. Ele pode ser um `<Dialog>` com `open={!subscribed}` triggered after a delay or certain condition. Simpler: if not subscribed, include an absolutely positioned overlay with the form and a backdrop. However, as this is complex, você pode implementar gating de forma mais simples: exibir a seção "Inscreva-se na newsletter para continuar lendo..." inline quando `!subscribed` após first few paragraphs (like a divider). Mas no original, pareceu aparecer depois de um certo scroll (após leitura do artigo). Podemos comprometer: exibir no final do content as bloqueio:

```
{!subscribed && (
  <div className="my-8 p-4 bg-gray-50 border border-gray-300 text-center">
    <h3 className="text-xl font-bold mb-2">Inscreva-se na newsletter para continuar lendo. É grátis!</h3>
    <p className="text-sm mb-4">Este não é um acesso pago e a adesão é gratuita.</p>
    <NewsletterForm onSuccess={() => setSubscribed(true)} />
  </div>
)}
```

E se subscribed, mostrar botão "Continuar Lendo" que simplesmente some ou se fosse modal, fechar. Entretanto, isso diverge do original que aparentemente bloqueava meio do artigo. Dado a complexidade, essa simplificação serve.

- O texto "Você possui 1 artigo para ler sem se cadastrar" etc. poderíamos mostrar se quisermos replicar totalmente. Ex: se localStorage contagem at 0: show "Você possui X artigos..."
- Esses detalhes são extras, mas inclua um comentário no código indicando onde estaria isso caso fosse implementado.

• **Página Sobre (Quem Somos)** (`app/sobre/page.tsx`):

- Usar dados da equipe (authors where author.role includes staff).
- Layout:

```
<div className="container mx-auto px-4 py-6">
  <h1 className="text-3xl font-bold mb-4">Quem Somos</h1>
  <h2 className="text-2xl font-semibold mb-4">Sobre o Intercept Brasil</h2>
  <p className="mb-6">[Missão do Intercept em alguns parágrafos...]</p>
  <h2 className="text-xl font-bold mb-3">Equipe e Colaboradores</h2>
  <div className="grid md:grid-cols-2 lg:grid-cols-3 gap-8">
    {teamMembers.map(member => (
      <div key={member.slug}>
        <Image src={member.photo} alt={member.name} className="w-24 h-24 object-cover rounded-full mb-2" />
        <h3 className="text-lg font-bold">{member.name}</h3>
        <p className="text-sm text-gray-700 mb-1">{member.role}</p>
        <p className="text-sm text-gray-600">{member.bioShort} <Link href={` /equipe/${member.slug}`} className="text-primary font-bold"> </p>
      </div>
    ))}
  </div>
</div>
```

```

Link></p>
    <p className="text-sm mt-1">
      <a href={`mailto:${member.email}`} className="mr-3
underline">Email</a>
      {member.twitter} && <a href={`https://twitter.com/${
member.twitter}`} className="underline">@{member.twitter}</a>
    </p>
  </div>
)}
</div>
</div>

```

- Assim exibimos foto, nome, cargo, bio curta e contatos. O link "→" leva à página de perfil completa ⁹¹.

• **Página de Perfil do Autor** (`app/equipe/[autor]/page.tsx`):

- Dados: find author by slug, list posts by author.
- Layout:

```

const author = authors.find(a => a.slug === params.autor);
const posts = allPosts.filter(p => p.authors.includes(author.name));
return (
  <div className="container mx-auto px-4 py-6 max-w-prose">
    <div className="flex items-center mb-4">
      <Image src={author.photoLarge} alt={author.name} className="w-32
h-32 object-cover rounded-full mr-4" />
      <div>
        <h1 className="text-3xl font-bold">{author.name}</h1>
        <p className="text-gray-700">{author.role}</p>
      </div>
    </div>
    <div className="prose mb-6">{author.bioFull}</div>
    <div className="mb-6">
      <h2 className="text-xl font-bold mb-2">Contatos:</h2>
      <p>
        <a href={`mailto:${author.email}`} className="underline
mr-4">Email</a>
        {author.twitter} && <a href={`https://twitter.com/${
author.twitter}`} className="underline mr-4">@{author.twitter}</a>
        {/* other contacts like bluesky if any */}
      </p>
    </div>
    <h2 className="text-2xl font-semibold mb-4">Últimas publicações de
{author.name}</h2>
    <div className="grid md:grid-cols-2 gap-6">
      {posts.map(post => <ArticleCard key={post.slug} article={post} /
>)}
    </div>
  </div>
)

```

```
</div>
);
```

- Mostramos a foto em tamanho maior, nome, função, bio completa (vários parágrafos formatados), contatos e lista de matérias.
- Adicionar paginação se muitos posts.

• **Página FAQ** (`app/faq/page.tsx`):

- Você pode codar estático (fácil) ou mapear perguntas de um array.
- Usar o Accordion do shadcn:

```
<div className="container mx-auto px-4 py-6 max-w-prose">
  <h1 className="text-3xl font-bold mb-4">Perguntas Frequentes</h1>
  <Accordion type="single" collapsible>
    {faqItems.map(item => (
      <AccordionItem key={item.q} value={item.q}>
        <AccordionTrigger className="text-left">{item.q}</
        AccordionTrigger>
        <AccordionContent><p>{item.a}</p></AccordionContent>
      </AccordionItem>
    ))}
  </Accordion>
</div>
```

- Estilize AccordionTrigger para parecer h4 (talvez add class font-semibold text-lg).
- Itens do item.a podem ter listas; ensure they render properly (maybe put raw HTML or parse).

• **Páginas Política de Privacidade & Termos de Uso** (`app/privacidade/page.tsx` e `app/termos/page.tsx`):

- Podem ser texto estático longo. Inserir headings (h1, h2) e parágrafos. Formatar de forma simples (usando `<article className="prose mx-auto">` por ex.).
- O conteúdo pode ser escrito manualmente ou placeholders. É menos crítico visualmente; principal é ter esses links válidos e apresentáveis.

• **Página de Busca** (`app/buscar/page.tsx`):

- Pegar o search query via `searchParams` (Next 13 feature):

```
export default function SearchPage({ searchParams }) {
  const query = searchParams.q?.toLowerCase() || '';
  const results = allPosts.filter(p =>
    p.title.toLowerCase().includes(query) ||
    p.contentText.toLowerCase().includes(query));
  return (
```

```

<div className="container mx-auto px-4 py-6">
  <h1 className="text-2xl font-bold mb-4">Você pesquisou por <span
className="italic">"{query}"</span></h1>
  {query === '' ? <p>Digite um termo na busca.</p> : results.length
=== 0 ? (
    <p>Nenhum resultado encontrado.</p>
  ) : (
    <div className="grid md:grid-cols-2 gap-6">
      {results.map(r => <ArticleCard key={r.slug} article={r} />)}
    </div>
  )}
</div>
);
}

```

- Talvez inclua a possibilidade de realizar a busca novamente (um input no topo?), mas se já temos o header search, pode não precisar.
- Use `decodeURIComponent` if needed for query.

Passo 5: Testes e Refinamento

- Monte a aplicação localmente e abra em navegador.
- **Teste cada link do menu** para ver se navega corretamente.
- Inspeção em responsivo dev tools para tamanhos de celular, tablet, desktop. Ajuste classes Tailwind se algo quebra (e.g., muito comprido -> talvez adicionar `overflow hidden` ou `break-words` para títulos muito longos).
- Teste o **menu mobile**: Abra em tela pequena, clique hambúrguer, navegue nos submenus. Ajuste animação e focus trap (shadcn's Sheet cuida de focus).
- Teste o **formulário newsletter**: tente submeter sem email (deve requerir), com email (veja se aparece sucesso).
- Para gating: simule leitura de 2 artigos e veja se modal aparece (se implementado).
- Teste **busca**: procure termos sabidamente existentes e inexistentes.
- Acessibilidade: Use Tab para navegar, veja se vai aos menus, se foca corretamente em modal search e sheet.
- Performance: Run `next build` e `next start` se possível, ou ver Lighthouse. Otimize se imagens grandes etc.
- SEO: Cheque `<title>` e `<meta>` tags (pode adicionar via `head.js` in each route or use `next/head` in pages dir).
- Correções visuais: compare com site original se possível (mesmo que não possamos ver CSS, pelas capturas que fizemos).
- Exemplo, certificarmo-nos de que o link "ACESSE MAIS MATÉRIAS" está em caps e com setas se necessário.
- Ajuste cores de fundo do modal gating e etc., se elas diferem.

Passo 6: Documentação e Entrega

- Adicione comentários no código explicando partes importantes, principalmente onde fez substituições por conteúdo fictício e lembrando que imagens e texto são placeholders não reais.
- Informe ao cliente quaisquer diferenças intencionais (por exemplo, "URLs de posts sem data por simplificação", ou "não implementamos contagem real de artigos lidos, apenas simulado" etc.).

- Inclua instruções de como adicionar novos conteúdos nas estruturas de dados (caso queiram no futuro).
- Confirme que nenhum asset do Intercept original foi usado – verifique logos, imagens e remova se acidentalmente ficou algo.
- Teste final cross-browser (Chrome, Firefox, mobile Safari via devtools).

Seguindo estes passos e referências, um desenvolvedor conseguirá recriar um clone **visual e funcionalmente idêntico** ao site do Intercept Brasil, utilizando ferramentas modernas e código original. O resultado final terá a mesma estrutura de navegação, layout responsivo, componentes visuais (cards, tarjas, menus) ⁹² ⁹³ e funcionalidades (busca, newsletter, compartilhamento, etc.) presentes no site de referência – tudo implementado do zero, respeitando os direitos autorais de conteúdo e ativos originais. Boa codificação! ⁹² ⁹³

¹ ² ⁶⁶ ⁶⁷ ⁶⁸ ⁶⁹ ⁷⁰ ⁷¹ ⁷² ⁸⁹ ⁹² Intercept Brasil

<https://www.intercept.com.br/>

³ ⁴ ⁵ ⁶ ⁷ ⁶¹ ⁷³ ⁷⁴ ⁸⁵ Poder | Intercept Brasil

<https://www.intercept.com.br/poder/>

⁸ ⁹⁰ Intercept Brasil | Séries

<https://www.intercept.com.br/todos-os-especiais/>

⁹ ¹⁰ ¹¹ ⁷⁷ ⁷⁹ Série Israel: estado genocida

<https://www.intercept.com.br/especiais/israel-estado-genocida/>

¹² ¹³ ¹⁴ ⁷⁸ Vozes | Intercept Brasil

<https://www.intercept.com.br/todas-as-vozes/>

¹⁵ ¹⁶ ¹⁷ ¹⁸ ¹⁹ Vídeos | Intercept Brasil

<https://www.intercept.com.br/videos/>

²⁰ ²¹ ²² ²³ ²⁴ ²⁵ ²⁶ ²⁷ ²⁸ ²⁹ ³⁰ ³¹ ³² ³³ ³⁴ ⁷⁵ ⁷⁶ ⁸¹ ⁸⁴ ⁸⁸ ⁹³ Hugo Motta usa até
seguranças para limitar trabalho da imprensa

<https://www.intercept.com.br/2025/09/09/motta-seguranca-restricoes-imprensa-camara/>

³⁵ ³⁶ ³⁷ ³⁸ ³⁹ ⁴⁰ ⁹¹ Intercept Brasil - Quem Somos

<https://www.intercept.com.br/sobre/>

⁴¹ ⁴² ⁴³ ⁴⁴ ⁴⁵ ⁸⁷ Andrew Fishman, presidente e cofundador, Intercept Brasil

<https://www.intercept.com.br/equipe/andrew-fishman/>

⁴⁶ ⁴⁷ ⁴⁸ ⁴⁹ ⁵⁰ ⁵¹ ⁸² O Intercept Brasil quer receber suas denúncias | Intercept Brasil

<https://www.intercept.com.br/fontes/>

⁵² ⁵³ ⁵⁴ ⁵⁵ ⁵⁶ ⁵⁷ ⁵⁸ ⁵⁹ ⁸⁰ ⁸⁶ Newsletter do Intercept. Assine. É de graça.

<https://www.intercept.com.br/newsletter/>

⁶⁰ Você pesquisou por aborto | Intercept Brasil

<https://www.intercept.com.br/search/aborto/feed/rss2/>

⁶² ⁶³ ⁶⁴ ⁶⁵ ⁸³ Perguntas Frequentes

<https://www.intercept.com.br/perguntas-frequentes/>