Introdução à Programação

Aprendendo as ferramentas No Code

PROJETO DESENVOLVE

Sumário

Introdução 4
High Code 5
Low-Code e No-Code 5

Vantagens e Desvantagens 7
Personalização 8
Programação e Tempo de Aprendizado 10
Custos 12
O que dá e o que não dá para fazer? 12

Ferramentas 16
Como escolher uma ferramenta 19

Parte II Bubble 39
O editor do Bubble 39
Abas e seções 42
Principais recursos 45
Ferramentas 47
Pré-visualização e Implantação 48
Liberando para visualização 49
Traduzindo para Português 50
Configuração de responsividade 50
Construindo seu primeiro aplicativo 51
Planejando os recursos (features) 52
MVP 53
Design - UX e UI 54
Estudo de caso: Lista de supermercado 56
Configurando o layout 57
Cabeçalho 59
Cadastrando um item 64

Criando e preparando um banco de dados 67
Mostrando a lista na tela 74
Modificar o estado de um item 81
Estudo de caso: Aprimorando a lista de supermercado 90
Ordenando a lista 90
Editando um item 94
Página Dinâmica 94
Popup 99
Implantação(Deploy) 103

Parte III FlutterFlow 107
Dashboard do FlutterFlow 109
App Builder 112
Menu de Navegação 113
Barra de Ferramentas (Tool bar) 116
Tela (Canvas) 119
Painel de Propriedades 122
Propriedades de Widgets 123
Propriedades da Página 125
Estudo de caso: Lista de Supermercado 127
Firebase 127
Criando o projeto 136
Criando as telas do aplicativo 139
CRUD - Create 150
CRUD - Read 157
CRUD - Update 162
CRUD - Update da tela de edição 167
CRUD - Delete 174
Estudo de caso: Aprimorando nosso aplicativo 178
Implantação (Deploy) 183

Introdução

Esse é nosso curso de No Code do Projeto Desenvolve. Pode parecer contraintuitivo um curso "sem código" dentro de um curso de formação de pessoas desenvolvedoras full stack. Mas não se assuste! Me acompanhe nessa jornada para entendermos onde podemos aplicar esse tipo diferente de criação de programas.

Para apresentarmos o conceito do No Code precisamos conhecer também o que é High Code e Low Code.

High Code

High-Code ("Alto") é uma abordagem padrão de desenvolvimento em que você deve escrever todos os comandos que o dispositivo deve executar. É o que chamamos de **programação tradicional**, na qual se utiliza uma linguagem de programação, com um conjunto de códigos, dados e algoritmos específicos daquela programação. Aqui no Desenvolve, você está estudando isso com a linguagem Python, por exemplo.

Para conhecer um pouco mais sobre a história de onde vem a programação tradicional, assista aos vídeos indicados no nosso Material Complementar.

Low-Code e No-Code

Low-Code ("Baixo") vem da ideia de ser possível criar aplicativos usando menos código de maneira prática e intuitiva. Estas plataformas disponibilizam modelos prontos, componentes de código pré-configurados, recursos visuais e assim, a programação se concentra mais na customização da aplicação do que na criação da estrutura básica.

A tecnologia de Low-Code tem suas raízes no início dos anos 2000, com o surgimento das plataformas de desenvolvimento rápido de aplicativos (rapid application development - RAD). Essas plataformas tinham como objetivo fornecer uma abordagem mais visual, de arrastar e soltar, para o desenvolvimento de aplicativos, permitindo que os usuários construíssem aplicativos com mais rapidez e menos codificação.

Em meados da década de 2010, o low-code tornou-se uma tecnologia dominante, adotada por organizações de todos os tamanhos pela sua capacidade de acelerar o desenvolvimento e reduzir custos.

Dizer que as plataformas de desenvolvimento de baixo código tiveram impacto no mundo da programação seria um eufemismo monumental; eles revolucionaram a forma como as empresas e os desenvolvedores trabalham.

Em 2014, a empresa **Forrester** cunhou o termo "Low-code" para classificar plataformas de desenvolvimento com foco na simplicidade do desenvolvimento e na facilidade de uso. Essas plataformas permitiram que desenvolvedores e usuários de todos os níveis de habilidade codificassem aplicativos, sem a necessidade absoluta de saber codificação.

A **Forrester**[1] afirma que o termo "desenvolvimento low-code" foi usado pela primeira vez em um relatório de 2011 sobre novas plataformas de produtividade para a construção de aplicativos personalizados.

Mas ainda assim, Low-code tem como ponto importante a ideia de ser híbrido. Você ainda pode manipular um pouco de código dentro dele. Por isso vem sua evolução natural: o No-Code.

No-code significa "sem código". Permite que você desenvolva soluções web e mobile sem programação. Muito usada por desenvolvedores e iniciantes na área, o serviço permite a criação de aplicativos, somente arrastando e soltando elementos em um ambiente de design ("Drag and Drop).

Bubble[2] foi a primeira plataforma sem código lançada em 2012 que permitiu a programação visual no lugar da codificação tradicional.

Um estudo comparativo de algumas plataformas de baixo código pode ser encontrado em: EconStor: Low code platforms: Promises, concepts and prospects. A comparative study of ten systems

Os autores realizaram uma comparação entre plataformas Low-Code e No-Code para o desenvolvimento de aplicativos móveis, destacando suas características, benefícios e desafios. Além disso, o artigo apresenta uma avaliação comparativa das principais plataformas Low-Code e No-Code disponíveis no mercado.

[1] https://www.forrester.com/
[2] https://bubble.io/

Outro estudo muito interessante sobre a usabilidade deste tipo de plataforma pode ser encontrado em: What about the usability in low-code platforms? A systematic literature review - ScienceDirect A usabilidade, em síntese, significa aquilo que promove o uso mais fácil das coisas seja ferramentas ou produtos. É a partir dela que você saberá se os usuários conseguem localizar as funções e entendê-las de forma rápida. Este estudo nos ajuda muito na escolha de qual ferramenta usar.

Plataformas low-code e no-code têm aplicações bastante amplas, desde a criação de aplicativos de gerenciamento de processos de negócios, passando por aplicativos de gestão de equipe e até soluções para Internet das Coisas (IoT). Além disso, elas podem ser utilizadas para criar soluções customizadas para atender às necessidades específicas de um projeto ou desafio de inovação.

A adoção de plataformas Low-Code e No-Code podem reduzir os custos de desenvolvimento, já que não é necessário contratar equipes de desenvolvedores ou investir em tecnologias complexas. Isso permite que pequenas e médias empresas acessem soluções tecnológicas avançadas, sem precisar investir muito.

No entanto, apesar de todas essas vantagens, ainda existem alguns desafios que precisam ser observados.

Vantagens e Desvantagens

Vantagens e Desvantagens

Para apresentar as vantagens e desvantagens dos diferentes conceitos apresentados até então, é preciso usar uma palavra que é muito importante dentro do nosso mundo de desenvolvimento, que é **depende**. **Depende** é a palavra que você mais vai ouvir de uma pessoa que é sênior, que é pleno, que vai estar há muito tempo nessa área. Como que eu determino vantagem e desvantagem? Vai depender do que você está se propondo a fazer. Então neste

curso vamos entender as vantagens e desvantagens em termos de tempo de aprendizado, de uso da ferramenta, do que você consegue fazer, da precificação, mas, ainda sim, teremos um grande depende ao final.

Personalização

Vamos tentar entender um pouco sobre até que ponto eu consigo usar um High-code, um Low-code, um No-code em termos de personalização (customização), em termos de conhecimento de programação, que já temos ideia de que o No-code não vai precisar de programação. Em termos de tempo, tanto tempo de curva de aprendizado de uma ferramenta ou de uma linguagem, ou tempo de produção mesmo de algum programa, algum aplicativo. Por que isso? Porque às vezes a gente quer realmente fazer só um teste. Por exemplo, o que a gente chama de MVP, mínimo produto viável, ou ainda uma prova de conceito: Eu tenho uma ideia, eu quero fazer ela rapidinho para ver se ela funciona. Nesse sentido, Low-code ou No-code irão me dar um tempo menor. E, por fim, a questão do preço também. Porque, querendo ou não, quando a gente trabalha em alguns aspectos de No-code e Low-code, não necessariamente é tudo gratuito.

Então, vamos começar pela personalização, a ideia da customização, do que você vai conseguir fazer dentro dessas ferramentas. Bom, quando a gente vai para High-code, você pode customizar o que você quiser. Como é programação tradicional, você pode personalizar da maneira que você quiser. Quando a gente vai para um Low-Code e para um No-Code, bom, isso pode diminuir, dependendo da ferramenta que se está utilizando. Em geral, as ferramentas Low-code vão te dar um pouquinho mais de personalização justamente porque elas são híbridas. Então a parte que é totalmente low-code, você faz em cima de modelos pronto, e quando você passa por um pouquinho de código, você consegue personalizar. Já o No-code não. A maioria das ferramentas você tem que pegar muita coisa pronta para você trabalhar. Existem formas de personalização, mas às vezes ela é mais restrita, principalmente nas versões gratuitas das ferramentas.

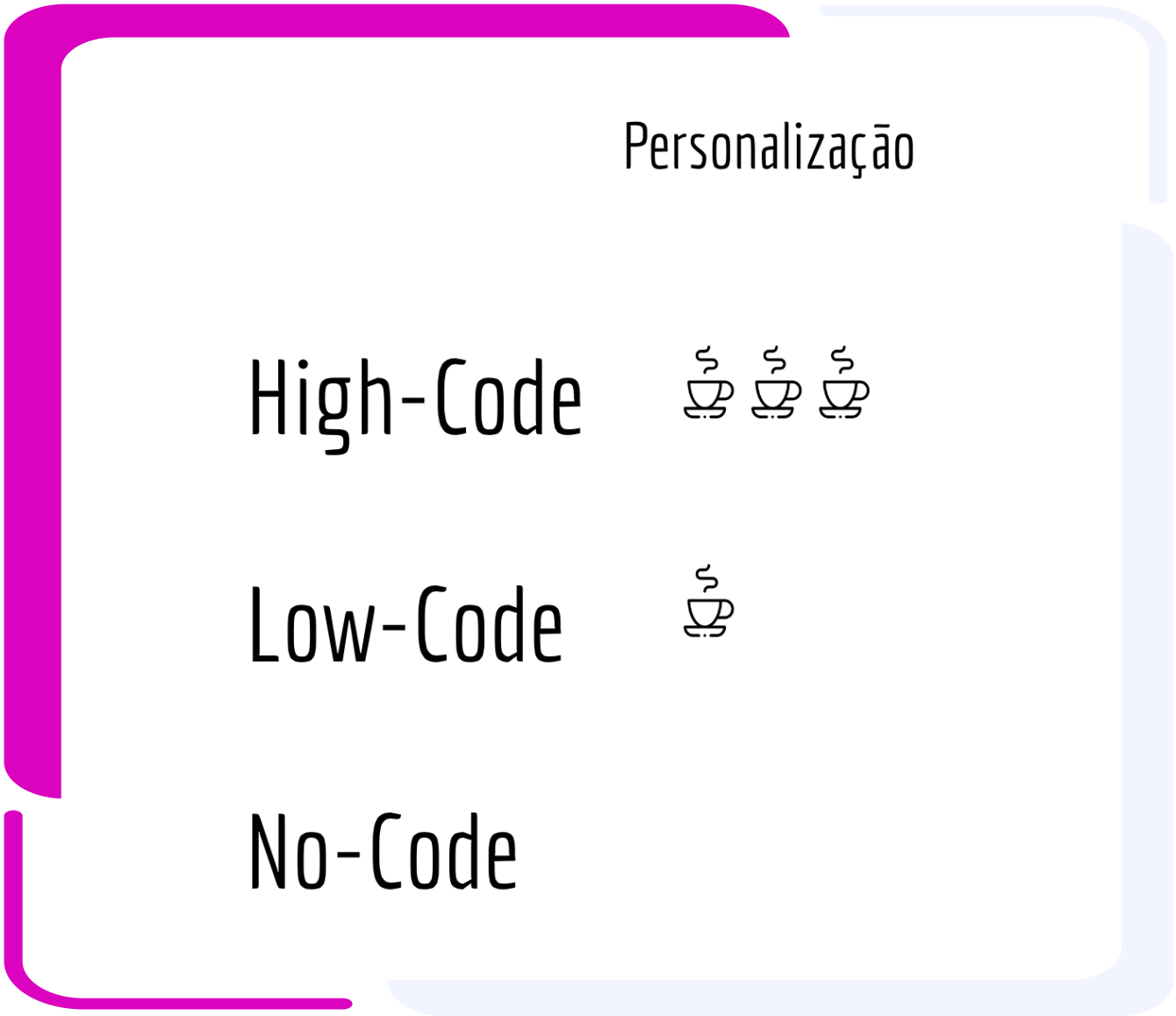


Figura 1. Comparação entre os tipos de ferramentas baseado no nível de personalização. Quanto menos xícaras de café, menos personalizável. Baseado nas versões gratuitas das ferramentas, as versões pagas Low-Code e No-Code ganham mais cafés.

Programação e Tempo de Aprendizado

Quando falamos sobre conhecimento de programação, obviamente com High-code você precisa saber programar, porque é a **programação tradicional**. Já com o Low-Code, já podemos mexer só com a parte mais de arrastar e clicar, não necessariamente precisamos mexer com a parte do código. Já com oLow-code você não precisa ter conhecimentos de programação mesmo, você não precisa saber uma linguagem de programação.

Vamos falar sobre o que é legal e o que não é, especialmente quando se trata do tempo e do esforço necessários para aprender a usar uma ferramenta. Quando você migra para um ambiente de High-code, tudo fica mais complicado e demorado. Você precisa aprender muita coisa, desde lógica de programação até diferentes linguagens de acordo com o que você quer criar, seja um aplicativo, um site ou um sistema para desktop. É verdade que, depois de entender os princípios e a lógica por trás, você pode se virar em várias linguagens. Mas ainda assim, a curva de aprendizado continua sendo um desafio, afetando diretamente o produto final que você consegue criar.

Certamente, ao lidar com a curva de aprendizado de uma linguagem de programação, o tempo necessário para desenvolver um Produto Mínimo Viável ou uma prova de conceito aumenta consideravelmente. Isso se deve ao investimento de tempo necessário para aprender a linguagem em profundidade. No entanto, ao optar por soluções low-code, essa curva de aprendizado é reduzida, pois podemos focar principalmente em provar meu conceito, sem mergulhar profundamente na codificação. E quando se trata de no-code, a redução é ainda mais drástica. Nesse caso, não é necessário aprender a linguagem de programação, mas apenas entender como usar a ferramenta. Portanto, o tempo necessário para criar um Produto Mínimo Viável ou uma prova de conceito é significativamente menor. Além disso, é importante notar que a curva de aprendizado para essas ferramentas é muito mais suave em comparação com o desafio de aprender uma ou várias linguagens de programação.

Nesse cenário, a escolha entre essas abordagens realmente depende do que você está procurando. Se você quer algo altamente personalizado ou precisa de algo rápido para testar suas ideias, é importante considerar esses aspectos.

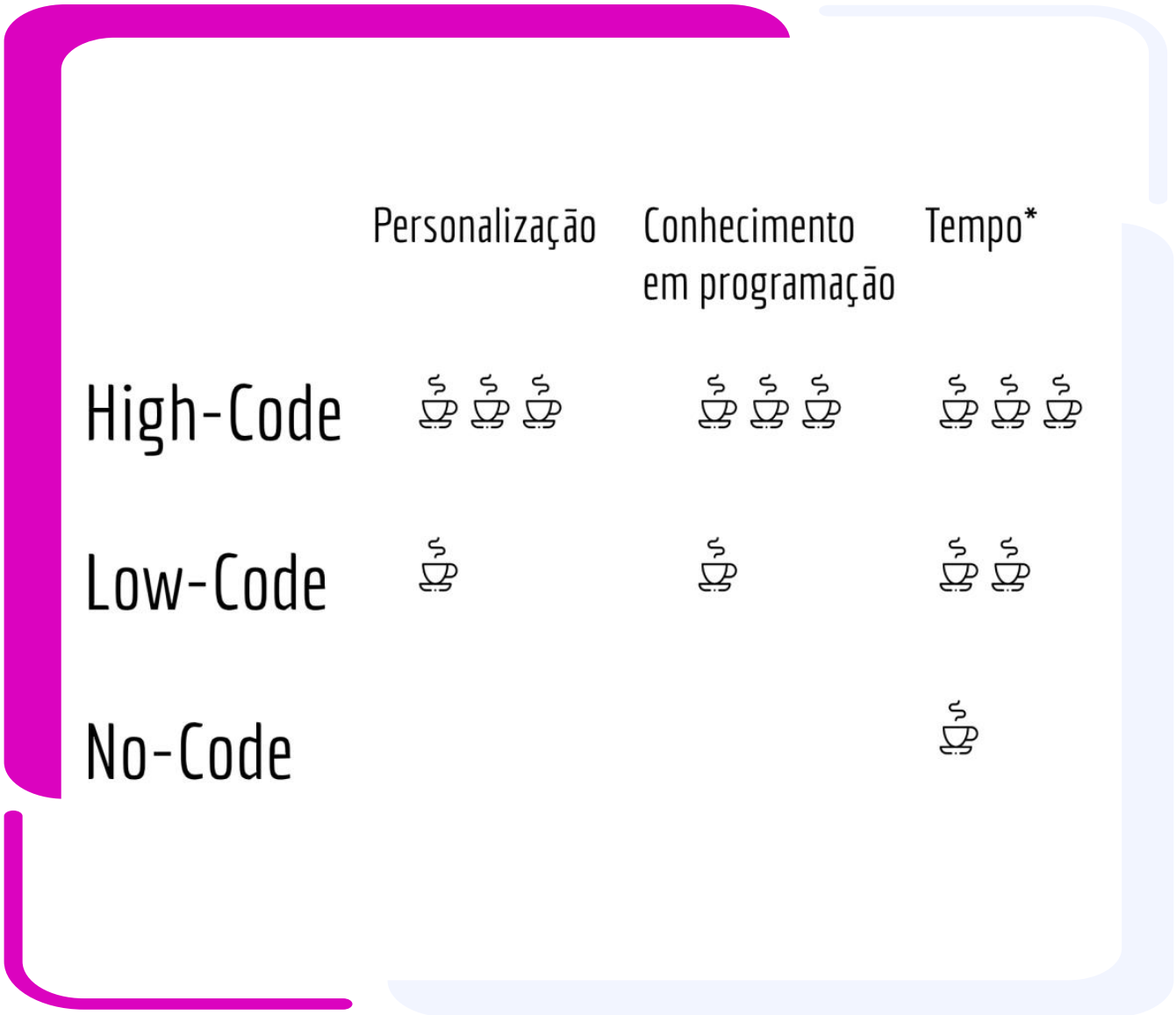


Figura 2. Comparação entre as ferramentas considerando o conhecimento em programação e o tempo de aprendizado, considerando o tempo de aprendizagem de uma linguagem de programação e/ou de uma dessas ferramentas.

Custos

Então entramos no último fator: custo. O High-Code não exige um pagamento direto, já que muitas das ferramentas que usamos, como compiladores, Ambientes Integrados de Desenvolvimento (IDEs), e até mesmo o Jupyter, são gratuitas. Por exemplo, o Visual Studio Code, comumente usado em disciplinas de web e outras, é gratuito. Então, o investimento principal está no tempo que você gasta aprendendo, não necessariamente no ato de criar e executar.

Com certeza, tanto o low-code quanto o no-code têm seu custo. Mesmo que haja uma versão gratuita, que é a que estamos usando aqui, se você quiser fazer algo mais complexo, vai precisar investir um pouco mais. É uma questão de avaliar se o que estão oferecendo, como os modelos e recursos prontos, compensa o gasto extra. A escolha entre essas ferramentas ou até mesmo a decisão de começar com uma e depois mudar para outra, por exemplo, começar com no-code e depois passar para high-code, vai depender muito do que você está tentando alcançar com o seu projeto. Então, é mais uma questão de entender suas próprias necessidades e objetivos para fazer a escolha certa.

Ou seja, é o nosso **depende**. A escolha pelo low-code e no-code depende inteiramente dos objetivos específicos de um projeto. Portanto, é preciso considerar cuidadosamente as limitações e capacidades associadas ao uso dessas ferramentas. No próximo capítulo, apresentaremos algumas ferramentas, delineando claramente as funcionalidades alcançáveis e aquelas que, sem dúvida, estão além do alcance das plataformas de low-code/no-code.

O que dá e o que não dá para fazer?

Iniciemos nossa análise pela exploração das possibilidades. Em linhas gerais, constatamos que as abordagens de desenvolvimento low-code e no-code se destacam pela sua aplicabilidade na concepção de marketplaces. O termo

marketplace denota uma concepção na qual os adquirentes são integrados aos seus respectivos vendedores, configurando, assim, uma plataforma destinada às transações comerciais. Por exemplo, destacam-se a OLX[3] , voltada para a comercialização diversificada de produtos, o Airbnb[4] , especializado na locação de propriedades residenciais, e o Dog Hero[5] , para a contratação de cuidadores para animais de estimação.

Esses exemplos evidenciam o caráter inovador dos marketplaces, nos quais a interação entre os indivíduos visa atingir propósitos específicos. Ao observarmos qualquer plataforma deste tipo, vemos uma uniformidade nas interfaces e experiências do usuário, uma característica preponderante que facilita consideravelmente a implementação por ferramentas no-code.

Desta forma, temos ferramentas muito interessantes que são feitas puramente no-code. Um exemplo é a Comet[6] . A Comet é um sistema todo feito em Bubble, que é uma ferramenta no-code. Ela é uma ferramenta interessante porque é uma ferramenta de busca de pessoas de T.I. Por exemplo, temos empresas que estão buscando pessoas desenvolvedoras, elas se cadastram nessa plataforma e se conectam. Como se fosse um LinkedIn[7] , mas só para pessoas da área de T.I. Navegue pela página para vocês terem ideia do potencial do no-code.

A utilização do no-code oferece amplas possibilidades, especialmente ao contemplar a ideia de padronização. Um exemplo notável é a criação de marketplaces, que se destaca como uma aplicação relevante sem a necessidade de programação convencional. Além disso, outra aplicação expressiva do no-code é na área financeira, notadamente em soluções fintech. Estas ferramentas são comumente empregadas para a gestão de contas, organização financeira, entre outras funcionalidades. Vale ressaltar que, nesse contexto, a ênfase não é necessariamente ser um banco, mas sim promover a colaboração com instituições bancárias para a eficaz organização de investimentos, por exemplo.

Sistemas de gestão em diversas áreas, como escolas e lojas, seguem padrões

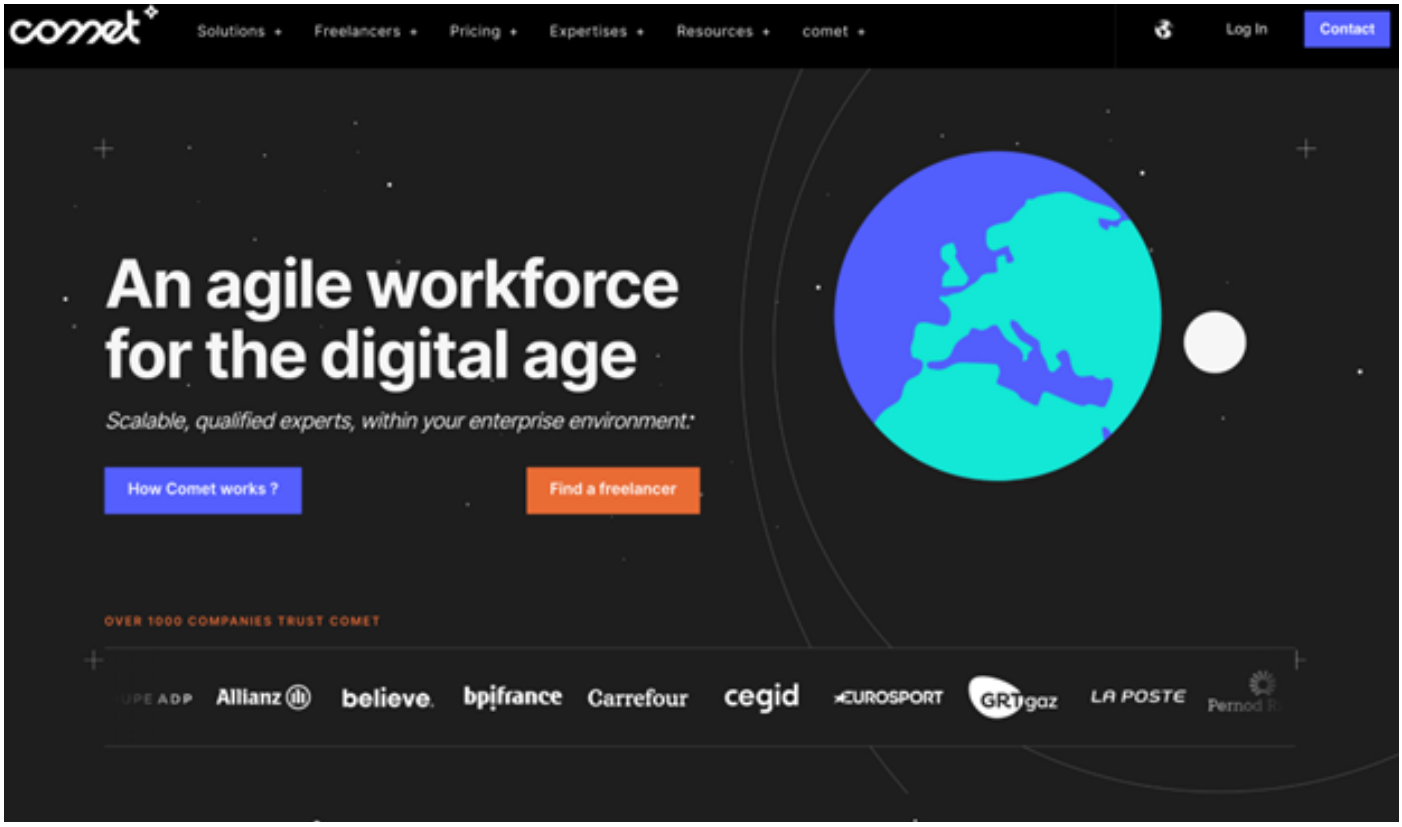


Figura 3. Página inicial da Comet.

específicos, facilitando o gerenciamento de estoques e compras. Os sistemas de entrega, que se tornaram proeminentes na última década, também seguem essa lógica padronizada, tornando-os adequados para implementação com abordagens no-code e low-code, assim como marketplaces e soluções fintech.

Essa uniformidade facilita a escolha de ferramentas para no-code e low-code. É justamente nesse ponto que precisamos destacar que tarefas que não se encaixam nesses padrões claros tornam-se desafiadoras para essas abordagens. A criação de jogos, por exemplo, é uma área complexa para no-code. Mesmo em casos como Candy Crush, com várias versões, a padronização é limitada. Ferramentas como RPG Maker[8] oferecem facilidades, mas o desenvolvimento de jogos geralmente demanda tecnologias além das capacidades no-code e low-code.

[3]https://olx.com.br/
[4]https://www.airbnb.com.br/
[5]https://www.doghero.com.br/
[6]https://www.comet.co/en-us/
[7]linkedin.com/

Outros sistemas que costumam ser complicados são os chamados sistemas embarcados. Esses sistemas lidam diretamente com o hardware, o que significa que precisam ser pensados considerando essa interação. Isso inclui sistemas de carros, aviões e até mesmo coisas mais simples, como geladeiras e relógios. No entanto, lidar com no-code nesses casos fica difícil, porque é necessário ter uma comunicação direta com o hardware. Geralmente, as empresas, especialmente as automotivas, criam seus próprios sistemas para lidar com essa complexidade.

Assim, a aplicação de no-code e low-code torna-se impraticável para soluções que fogem de padrões estabelecidos, como em situações que demandam alta personalização ou abordagens não convencionais. É fundamental compreender que no-code não equivale à programação convencional, sendo mais adequado, na maioria das vezes, para soluções genéricas e muito bem definidas.

Em resumo, a escolha entre no-code, low-code ou abordagens mais tradicionais depende do tipo de desenvolvimento desejado. no-code pode ser eficiente para soluções bem definidas e padronizadas, como sistemas de gestão, marketplaces e sistemas de delivery. E também quando queremos testar uma ideia, fazer uma prova de conceito[9]. Sua principal vantagem reside na rapidez de criação, manutenção e melhorias, bem como na curva de aprendizado mais acessível. Contudo, suas desvantagens incluem a limitação na abordagem de problemas mais complexos e não padronizados, os quais podem demandar a transição para soluções high code.

Exercício

Exercício de pesquisa 1: Será que existe solução fintech feita com Bubble? Pesquise a respeito e monte um pequeno resumo sobre a ferramenta que você encontrar.

Exercício de pesquisa 2: Pesquise duas ferramentas de gestão feitas com no-code ou low-code e monte um pequeno resumo sobre elas.

[8]https://www.rpgmakerweb.com/

[9]Prova de Conceito (Proof of Concept, em inglês, ou simplesmente PoC) é o nome que se dá à demonstração da possibilidade de validação de uma ideia (ou conceito).

Ferramentas

Ferramentas

Até aqui vimos as possibilidades e limitações, assim como as vantagens e desvantagens entre no-code, low-code e programação tradicional. Agora, veremos o extenso universo de ferramentas, caracterizando-o como um "zoológico" de opções disponíveis para no-code e low-code. A imagem abaixo mostra a enorme quantidade de ferramentas presentes no mercado.

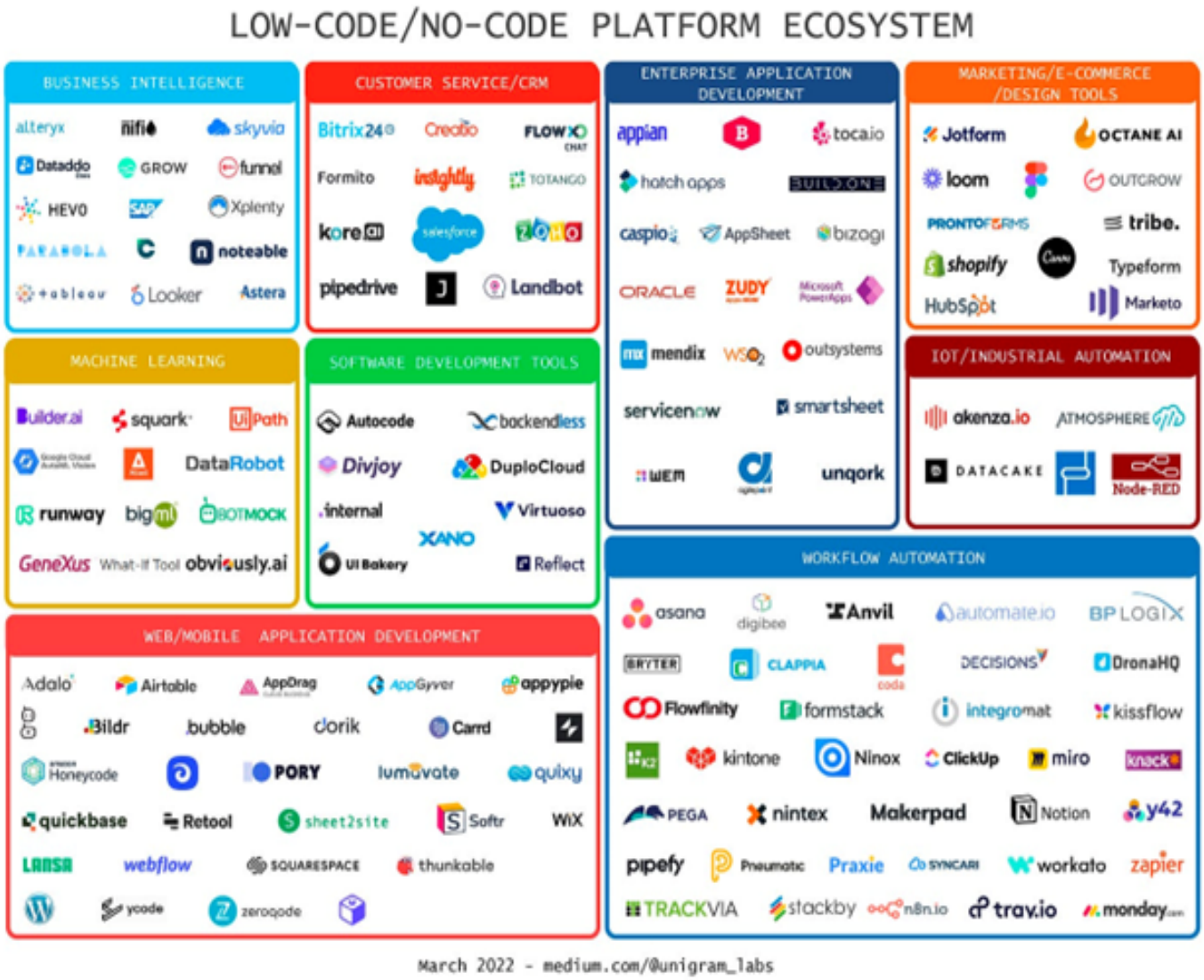


Figura 4.Compilado de ferramentas feito por https://medium.com/@unigram_labs/the-low-code-no-code-ecosystem-8a0e0ba757c1.

Há uma variedade maior de ferramentas e empresas do que mostra a Figura 4, no entanto, esta lista abrange um conjunto representativo e interessante de empresas para destacar. Outras listas mais abrangentes podem ser encontradas em:

- https://nocodelist.co/
- https://www.nocode.tech/

As ferramentas no-code e low-code nasceram para facilitar a vida da pessoa desenvolvedora e devem ser escolhidas dependendo do problema a ser solucionado. Na Figura 4, podemos separar as ferramentas em 9 categorias:

- Business Intelligence: Ferramentas que têm como objetivo transformar informações em conhecimento, visando aprimorar o processo de tomada de decisões em uma empresa. Exemplos desses tipos de ferramentas incluem Tableau[10] , Looker[11] e SAP Business Objects[12] , todas proporcionam aos usuários a capacidade de manipular, analisar e visualizar dados comerciais.
- Customer Service / CRM: Plataformas que gerenciam todas as comunicações, oportunidades e documentação relacionadas aos clientes, além de auxiliarem em todas as etapas do processo de compra e venda. Salesforce[13] é um dos maiores nomes nessa área, mas outras ferramentas, como Zoho[14] ,Bitrix24[15] e Creatio[16] , são plataformas que permitem às empresas automatizar seus processos de negócios para criar e gerenciar relacionamentos com os clientes.
- Enterprise Application Development (Desenvolvimento de Aplicações Empresariais): Essas plataformas de desenvolvimento visual possibilitam que desenvolvedores, com diversas habilidades, construam aplicativos empresariais completos, desenvolvendo e gerenciando bancos de dados personalizados e ferramentas para executar processos de negócios completos. Alguns exemplos grandes são ServiceNow[17] , Appian[18] e Mendix[19] .
- IoT / Automação Industrial: Esta categoria inclui tanto as plataformas IoT (Internet das Coisas), que conectam hardware à Internet, quanto às plataformas de automação industrial, que possibilitam a operação automática de processos industriais e maquinaria sem a necessidade de operadores humanos. Akenza.io[20] , por exemplo, é uma plataforma de IoT de autoatendimento que conecta, controla e gerencia dispositivos IoT, permitindo aos usuários construir produtos e serviços de IoT.

• Machine Learning (Aprendizado de Máquina): Com o crescimento da inteligência artificial e aprendizado de máquina, houve um aumento de ferramentas que buscam disponibilizar modelos de aprendizado de máquina para usuários não especializados em ciência de dados. Amazon, Microsoft e Google têm todas plataformas poderosas nesse sentido, mas empresas dedicadas ao no-code, como a DataRobot[21] , também estão fornecendo um número crescente de ferramentas do tipo AutoML (Aprendizado de Máquina Automático).

• Marketing / E-Commerce / Ferramentas de Design: A tecnologia de marketing (ou martech) já está no mercado há um bom tempo, com grandes empresas como Mailchimp[22] , Hubspot[23] e Marketo[24] capacitando os profissionais de marketing a construir uma gestão poderosa de campanhas e relacionamento com o cliente. Também incluímos ferramentas de comércio eletrônico, incluindo gigantes como Shopify[25] , que permitem que empresas de todos os tamanhos conduzam transações online enquanto gerenciam seus produtos, serviços e toda a sua loja virtual. Além disso, nesta categoria, adicionamos plataformas de design como Canva[26] e Figma[27] , onde você pode criar e compartilhar designs e modelos facilmente, sem a necessidade de habilidades especializadas.

• Ferramentas de Desenvolvimento de Software: Essas ferramentas são semelhantes ao desenvolvimento de aplicativos empresariais, mas direcionadas aos tecnólogos. Virtuoso[28] é um exemplo dessas ferramentas, oferecendo automação de garantia de qualidade (QA) e geradores de código para aplicativos web, o que possibilita a criação mais rápida e fácil de websites.

• Automatização de Fluxo de Trabalho (Workflow): Um fluxo de trabalho (workflow) pode ser definido como um conjunto de atividades relacionadas, que são concluídas em uma determinada ordem para atingir uma meta organizacional específica. Enquanto a automação permite que essas atividades sejam controladas por um sistema que garante sua completude, correção e repetibilidade. Zapier [29] ,Kissflow[30] e Pipefy[31] são exemplos de ferramentas que oferecem integração e/ou facilitam fluxos de trabalho de processos de negócios, possibilitando que usuários de no-code e low-code descrevam e executem processos de negócios completos de ponta a ponta.

• Desenvolvimento de Aplicações Web/Mobile: Esta é uma categoria extensa que

tem evoluído ao longo do tempo e inclui ferramentas consolidadas como WordPress[32] , bem como novas opções como Bubble[33] , Adalo[34] e Webflow[35] . Essas ferramentas frequentemente incluem um sistema de gerenciamento de conteúdo (CMS) e um conjunto de recursos de fácil utilização.

[10] https://www.tableau.com/

[11] https://cloud.google.com/looker/?hl=pt_br

[12] https://www.sap.com/products/technology-platform/bi-platform.html

[13] https://www.salesforce.org/

[14] https://www.zoho.com/?zredirect=f&zsrc=langdropdown&lb=es-xl

[15] https://www.bitrix24.com.br/?redirect_from=www.bitrix24.com%2F

[16] https://www.creatio.com/

[17] https://www.servicenow.com/

[18] https://appian.com/

[19] https://www.mendix.com/

[20] https://akenza.io/

[21] https://www.datarobot.com/

[22] https://mailchimp.com/pt-br/?currency=USD

[23] https://www.hubspot.com/

Como escolher uma ferramenta

[24] https://business.adobe.com/products/marketo/adobe-marketo.html

[25] https://www.shopify.com/

[26] https://www.canva.com/pt_br/

[27] https://www.figma.com/

[28] https://www.virtuoso-software.com/

[29] https://zapier.com/

[30] https://kissflow.com/

[31] https://www.pipefy.com/pt-br/

[32] https://wordpress.com/pt-br/

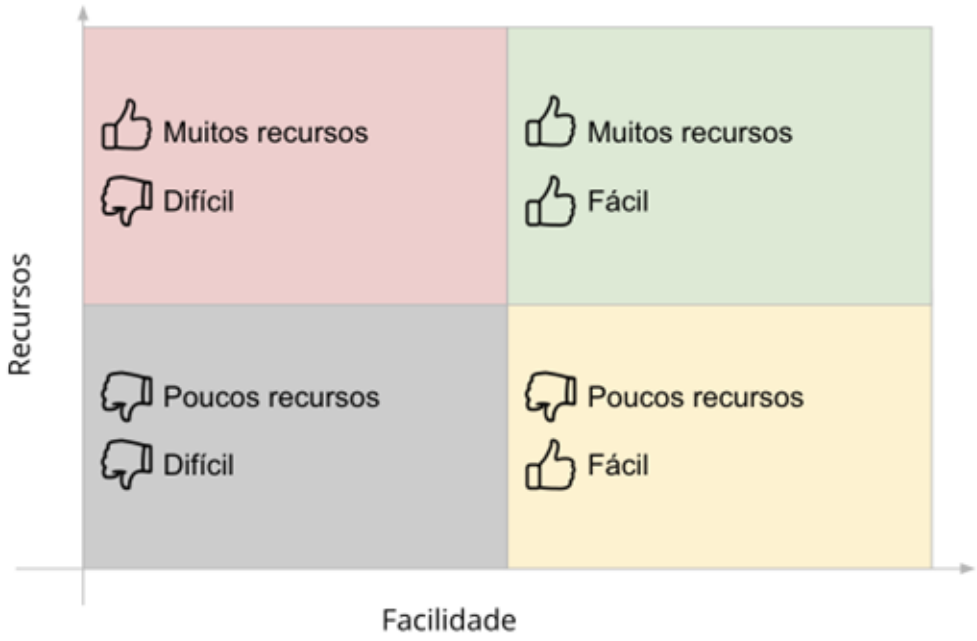
[33] https://bubble.io/

[34] https://www.adalo.com/

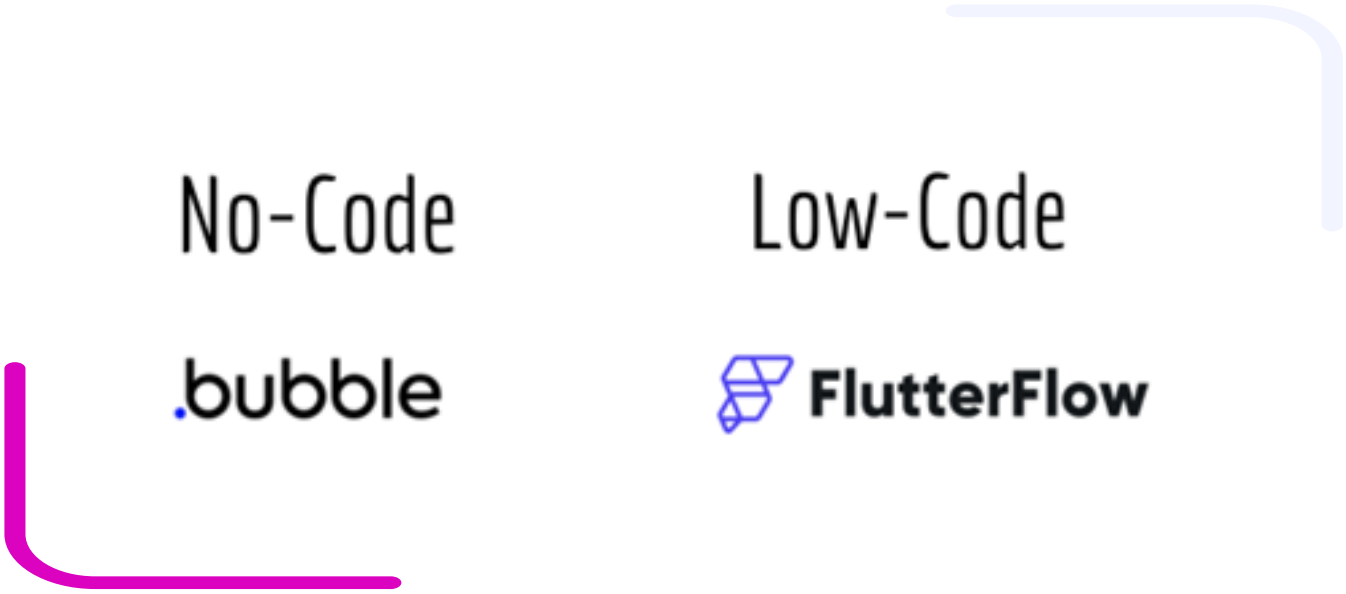
[35] https://webflow.com/

Neste curso vamos focar na última categoria de ferramentas, trabalhando com a parte de aplicativos para dispositivos móveis (Mobile).

Mesmo focando só nesta área, observa-se uma ampla variedade de ferramentas disponíveis. A escolha entre elas requer critérios específicos, considerando a quantidade de recursos oferecidos e a facilidade de uso. A facilidade abrange tanto o tempo necessário para criar um aplicativo quanto o tempo demandado para aprender a utilizar a ferramenta. A seleção adequada dependerá das necessidades específicas, utilizando como guia a premissa fundamental do desenvolvimento: "depende".



A escolha da ferramenta ideal envolve um equilíbrio entre a quantidade de **recursos** disponíveis e a **facilidade** de uso. Ter muitos recursos é positivo, mas se a complexidade de utilização for excessiva, o benefício diminui. Da mesma forma, uma ferramenta muito simples pode não atender às necessidades específicas. O objetivo é encontrar um meio-termo, onde a ferramenta seja fácil de usar, mas ofereça recursos suficientes para realizar as tarefas desejadas. Descartamos opções que são difíceis de usar e têm poucos recursos, buscando uma solução que seja uma vitória tanto em termos de facilidade quanto de funcionalidade. Das diversas ferramentas apresentadas, escolhi duas que oferecem um equilíbrio favorável, sendo de fácil aprendizado e ricas em recursos: Bubble[36] e FlutterFlow[37].



O Bubble.io é uma plataforma de desenvolvimento que possibilita a criação de aplicativos sem a necessidade de codificação, ou seja, totalmente no-code. Atualmente, o Bubble hospeda uma ampla variedade de aplicativos web, contando com milhões de usuários e abrangendo diversos modelos e formatos. A plataforma proporciona um editor totalmente visual, permitindo a rápida concepção e visualização de aplicativos. Além disso, dispõe de um editor de fluxo de trabalho que viabiliza a atribuição de ações aos elementos nas páginas. O

Bubble também oferece um banco de dados exclusivo, completo e plenamente configurável. As aplicações desenvolvidas na plataforma são hospedadas de forma segura e escalável.

FlutterFlow é uma plataforma no-code que apresenta uma interface de arrastar e soltar baseada em navegador para a criação de aplicativos móveis. Porém, é uma ferramenta que também possibilita seu uso como low-code. Isso porque a plataforma é construída com base no Framework Flutter, criado pela Google que utiliza a linguagem de programação Dart. Flutter é utilizado para criar aplicativos nativos de Android, iOS, Mac, Windows e Web também. Aqui no Desenvolve, você poderá fazer o curso de Desenvolvimento com Flutter futuramente.

Ambas as plataformas possuem opções gratuitas e pagas. No nosso curso, utilizaremos somente a versão gratuita.

[36]https://bubble.io/

[37]https://flutterflow.io/

Free	Starter (RECOMMENDED)	Growth	Team	Enterprise
Best for learning how to use Bubble	Best for launching your app and testing	Best for growing your user base	Best for scaling your team and business	Best for maximum security and scale
$0 / month	$29 / month Billed annually	$119 / month Billed annually	$349 / month Billed annually	Contact us
Try Free	Start a free trial	Get started	Get started	Talk to sales
Great for: Projects that are under construction	**Great for:** MVPs and simple tools with small to moderate user bases	**Great for:** Consumer projects with complex functionality	**Great for:** Scaling projects with high usage	**Great for:** Internal tools and customer-facing apps
Free plan features:	**Everything in Free, plus:**	**Everything in Starter, plus:**	**Everything in Growth, plus:**	**Everything in Team, plus:**
Development version	Live app	2 app editors	5 app editors	Choice of hosting location
API connector	Custom domain	Premium version control	Sub apps	Centralized admin
Component library	Recurring workflows	Two-factor authentication	25 custom branches	Dedicated server
1 app editor	Basic version control	10 custom branches	500k workload units/mo	Priority support
50k workload units/mo	175k workload units/mo	250k workload units/mo	20 days of server logs	Enhanced security
6 hours of server logs	2 days of server logs	14 days of server logs		Custom workload units

.bubble

Free	Standard	Pro	Teams
$0 per month	~~$30~~ $15 per month (regional discount)	~~$70~~ $35 per month (regional discount)	~~$70~~ $35 per user / month (regional discount)
Start building and testing your application today!	Download your project source code or APK.	Get access to all of our features & build applications faster than ever!	For teams that need collaboration and more advanced features.
Create Account →	Start Free Trial →	Start Free Trial →	Start Free Trial →
Visual Application Builder	Free Features +	Standard Features +	Pro Features +
100s of page & component templates	Code Download	Source Repository Integration	Real-Time Collaboration
Build Mobile, Web, and Desktop Apps	APK Download	One-Click Localization	Project Level Access Controls
API & Data Integration	Custom Domain Web Publishing	GitHub Integration	7 Day Audit Logs
Web Publishing		One-Click Deployment to Apple & Google App Stores	Centralized Billing

FlutterFlow

Prática I Bubble

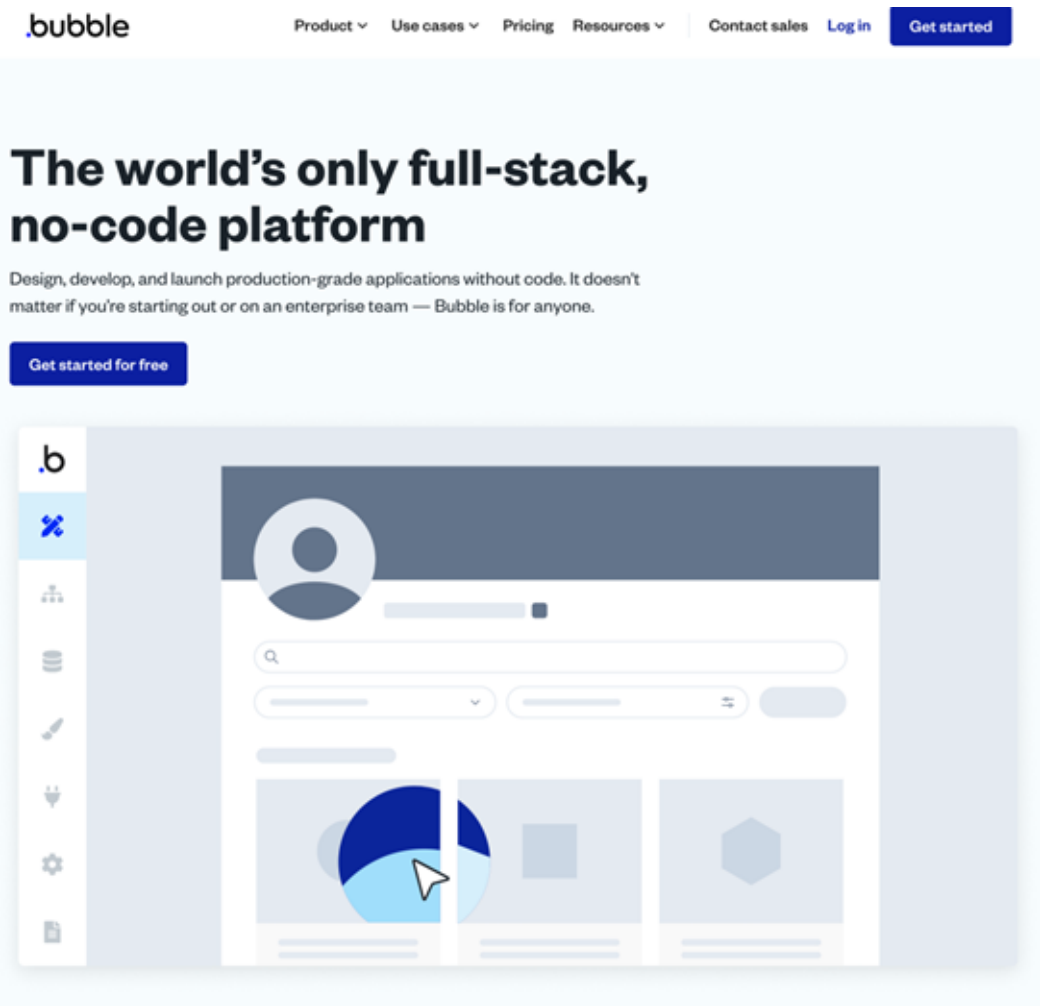
Bubble

Nesta nossa primeira prática, vamos criar uma conta na plataforma Bubble e entender um pouco sobre como ela funciona.

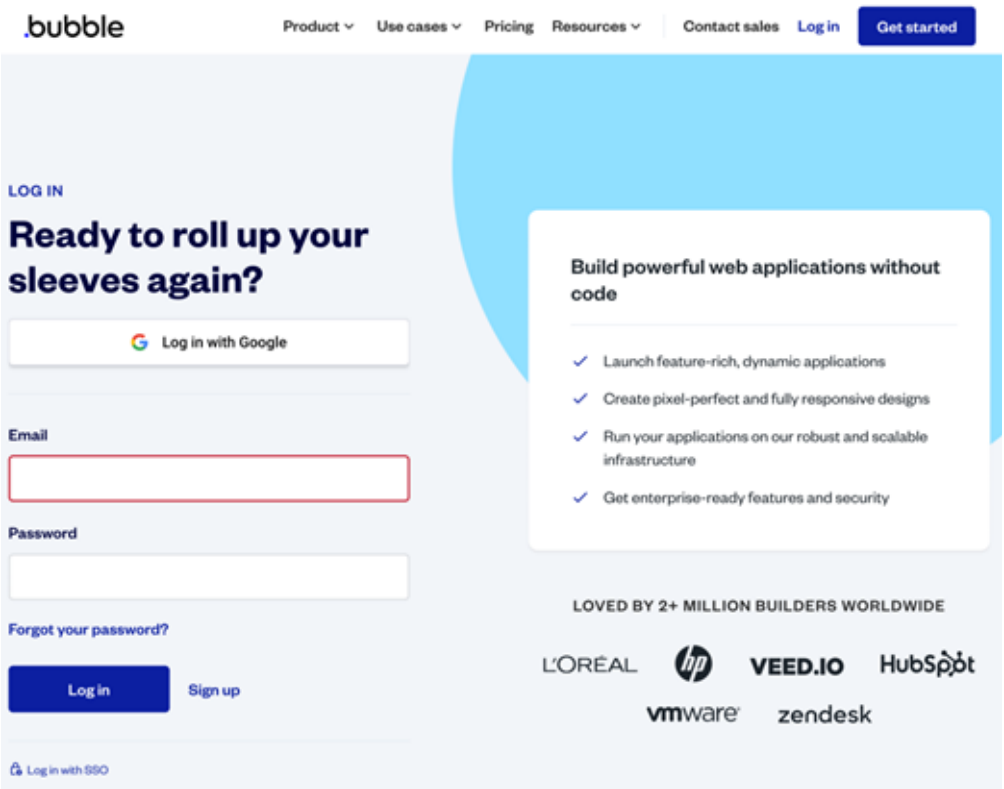
Criando uma conta no Bubble

Vamos seguir as seguintes etapas:

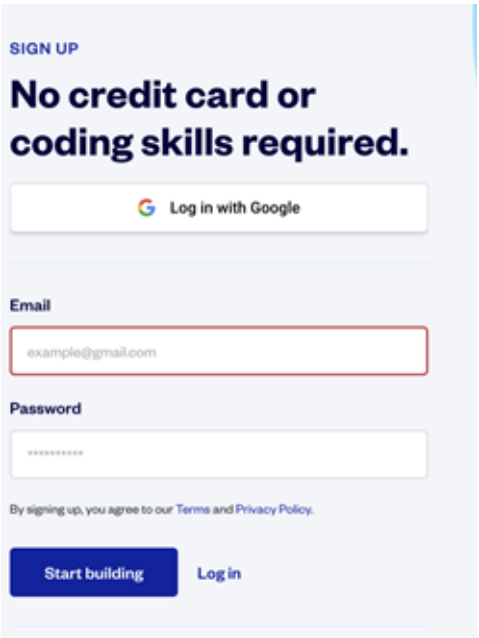
- Entrar em https://bubble.io/
- Ir em **Log in**



Na página do **Log in**, clique em **Sign up** para criar uma nova conta.

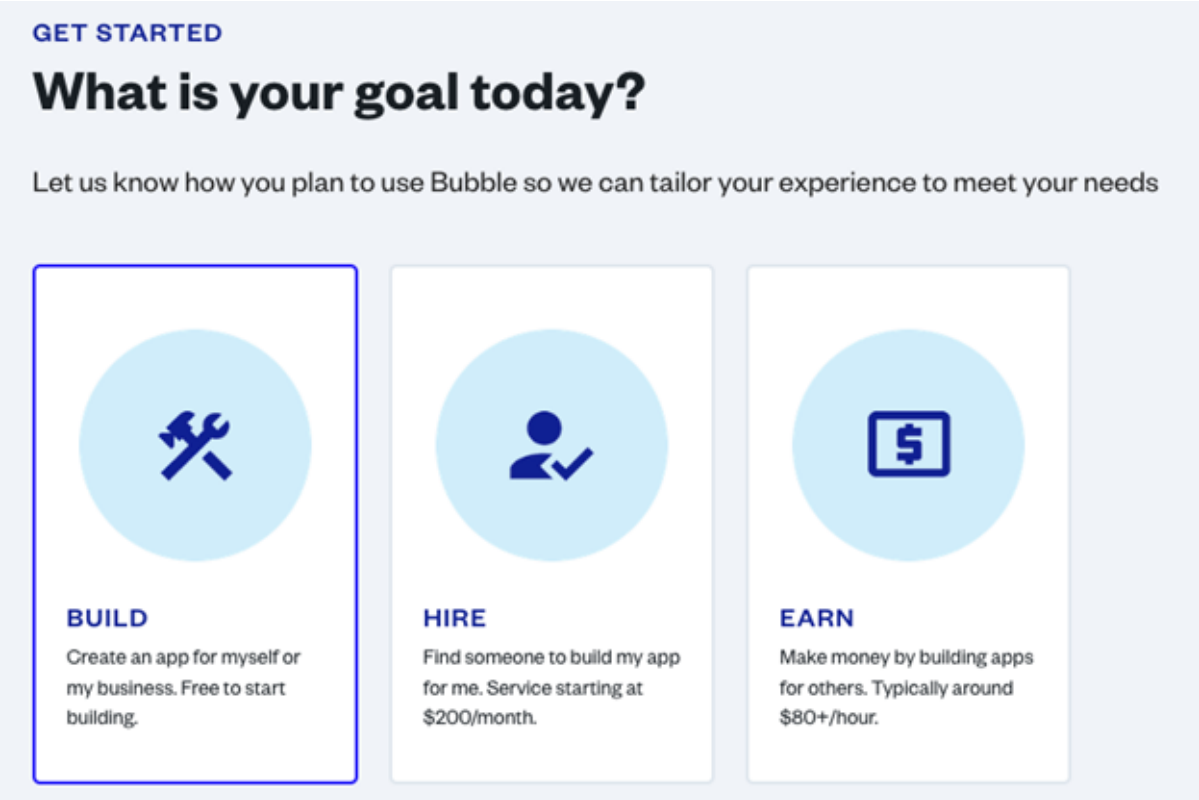


Você pode criar sua conta com o Log in da Google ou com email e senha. Pode escolher.

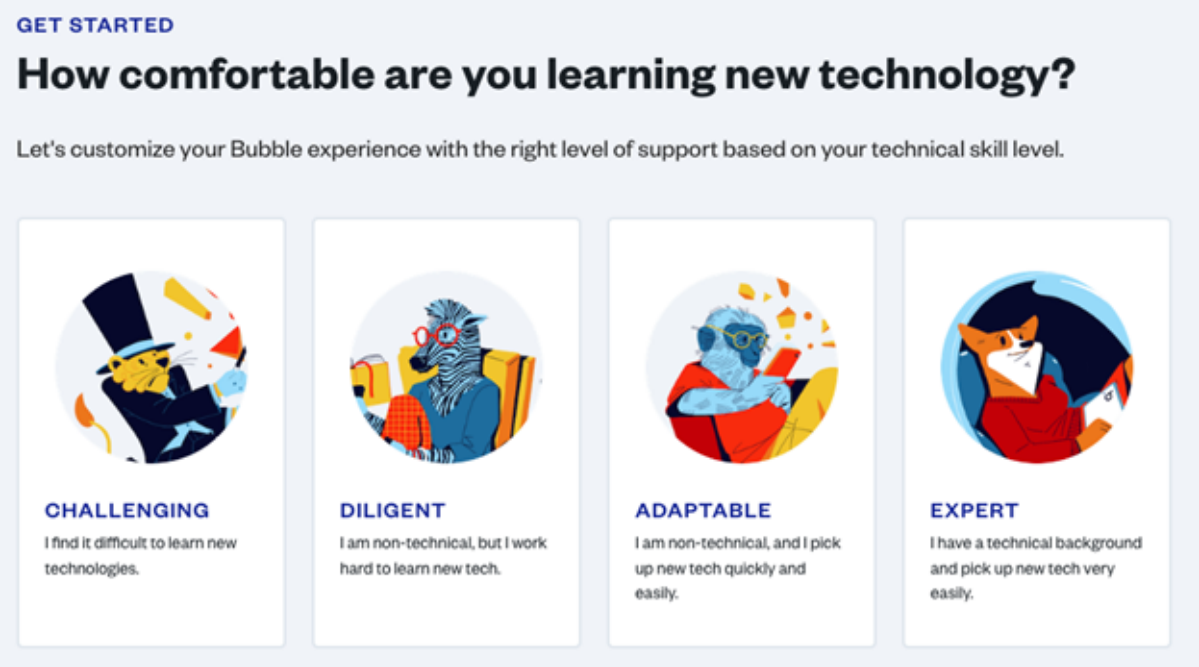


Clicando em **Log in with Google** você escolhe qual conta do Gmail você gostaria de vincular à sua conta do Bubble. Se quiser continuar apenas com email e senha, basta inserir esta informação nos campos e clicar em **Start Building**.

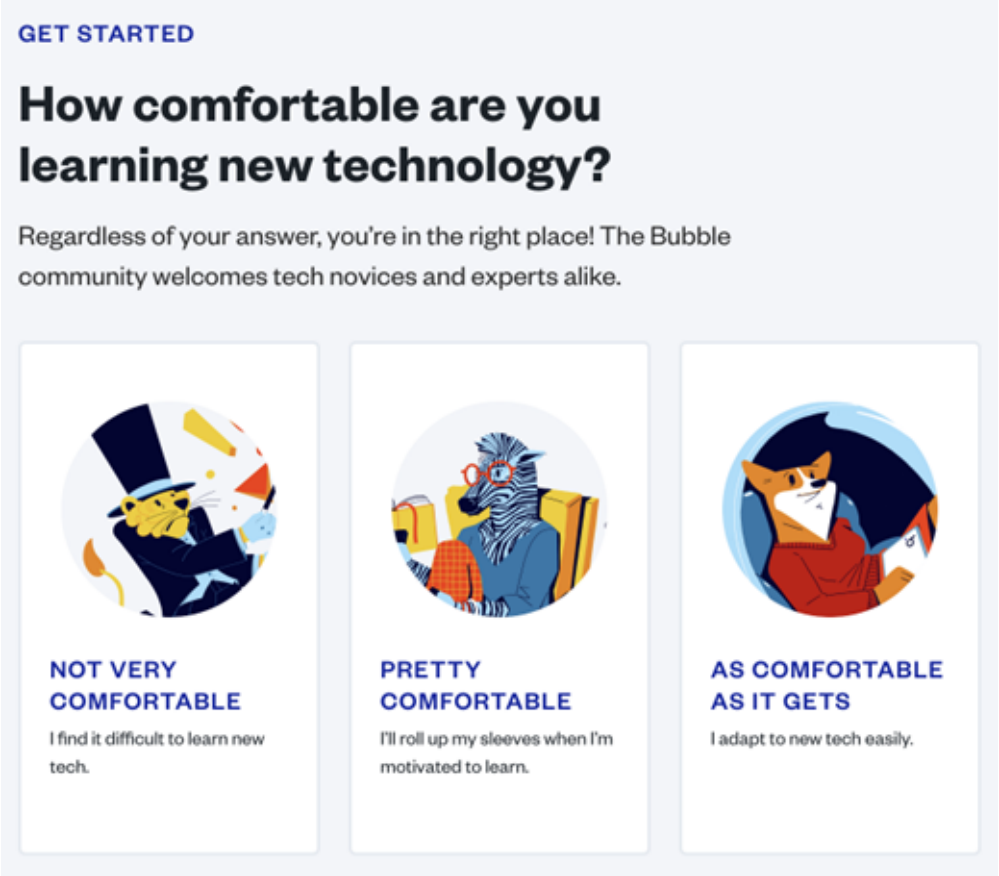
Nos dois casos você irá para a seguinte tela, na qual você deverá clicar em **Build**.



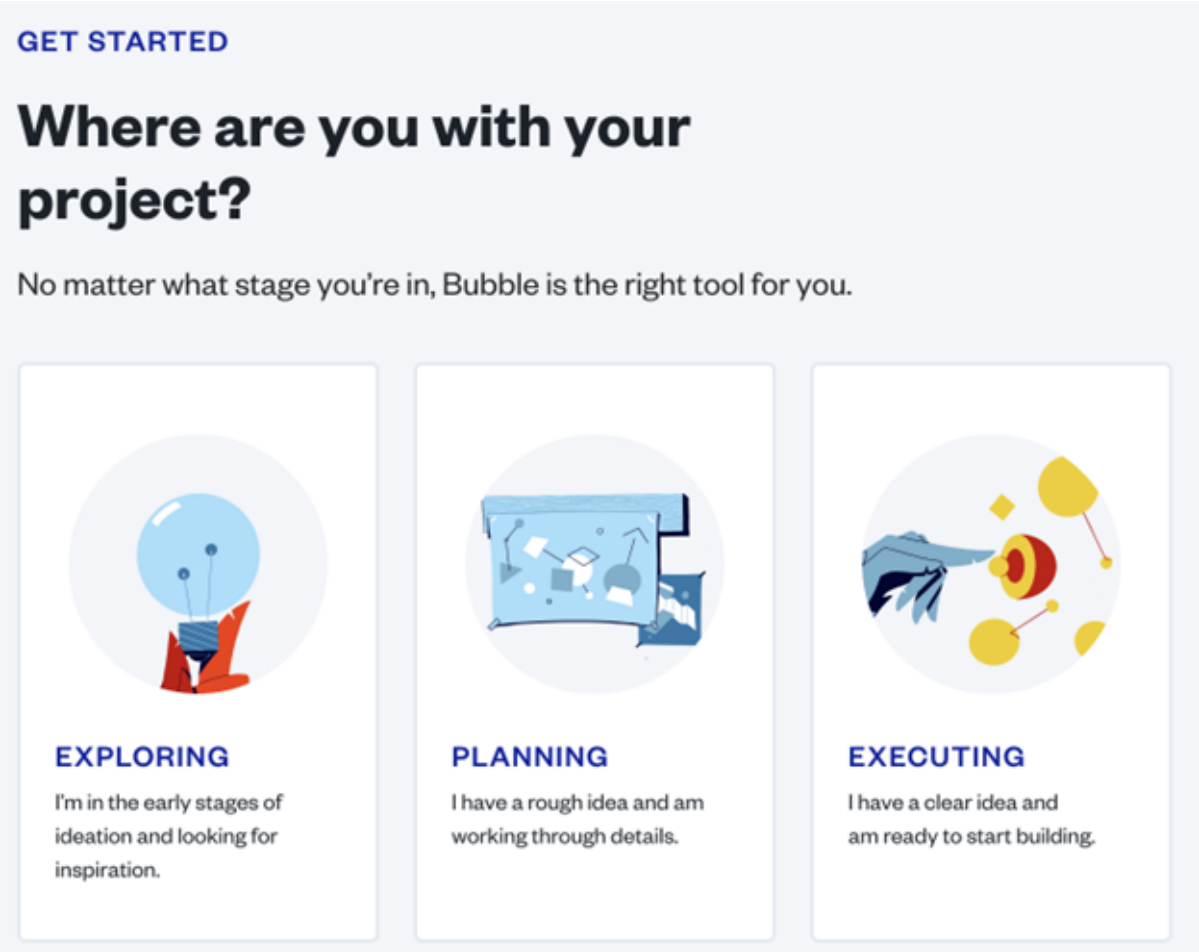
Depois escolha "O quão confortável você se sente em relação a uma nova tecnologia?". As opções são **Challenging**, se você tem muita dificuldade, **Diligent**, se você tem dificuldade mas se esforça, **Adaptable**, se você aprende rápido e **Expert**, se você já trabalha com tecnologia.



Em alguns casos, o **Expert** não aparece ou ainda os termos mudam para: Não muito confortável, bem confortável e tão confortável quanto possível.

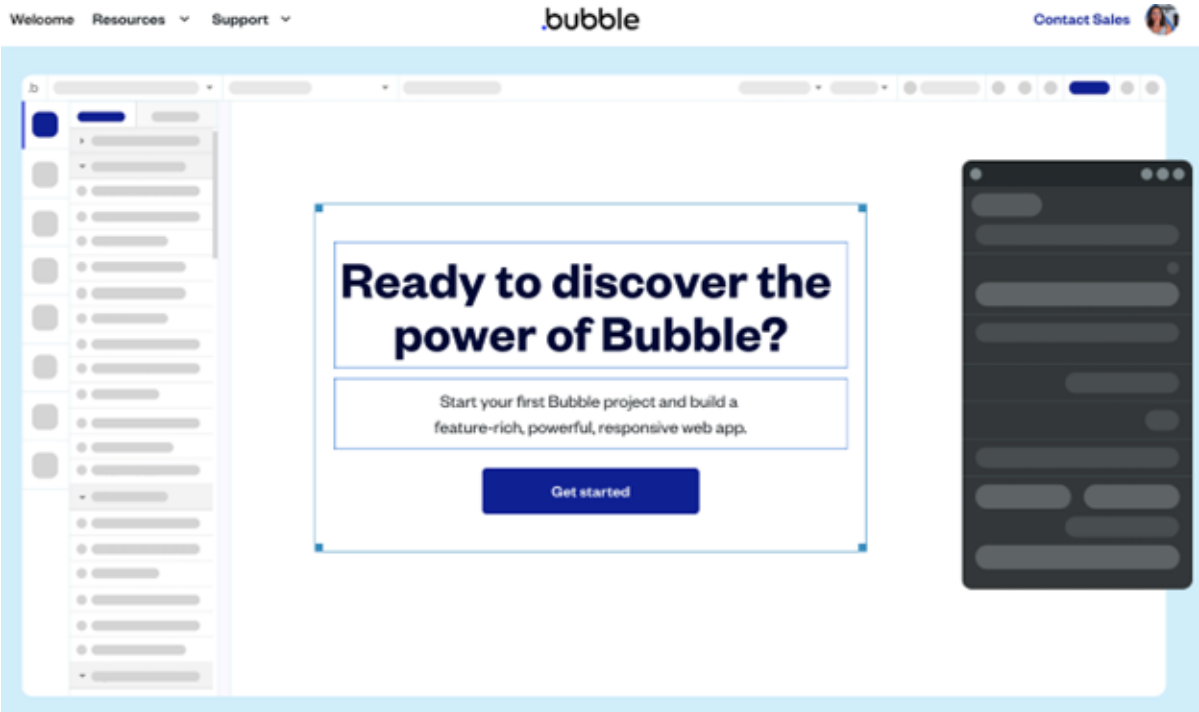


Depois, você deve escolher em que ponto está sua ideia: Explorando, planejando ou executando. No nosso caso estamos **Exploring**, explorando.



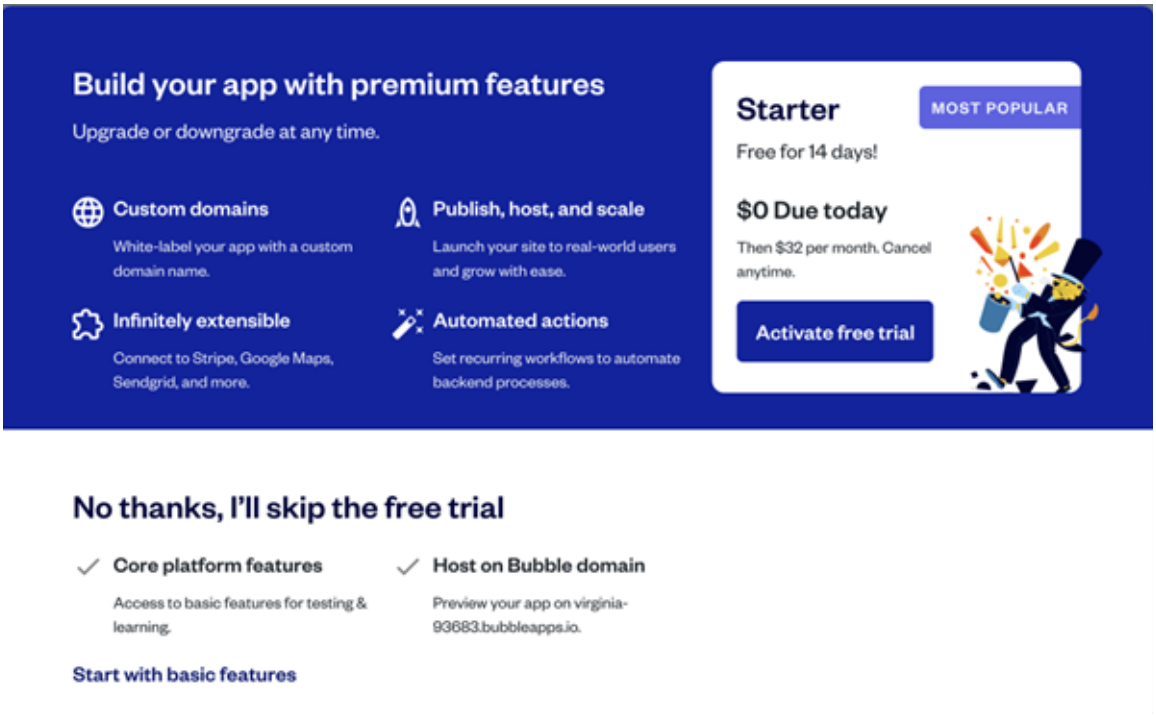
Por fim, a plataforma irá te perguntar como você ficou sabendo sobre ela. Você pode marcar **Course**, curso, pois você ficou sabendo dela por esta aula.

Pronto, a tela inicial está pronta para você trabalhar. Basta clicar em **Get started**, vamos começar.



Neste ponto, caso você ainda não tenha selecionado o modelo gratuito, esta tela irá aparecer:

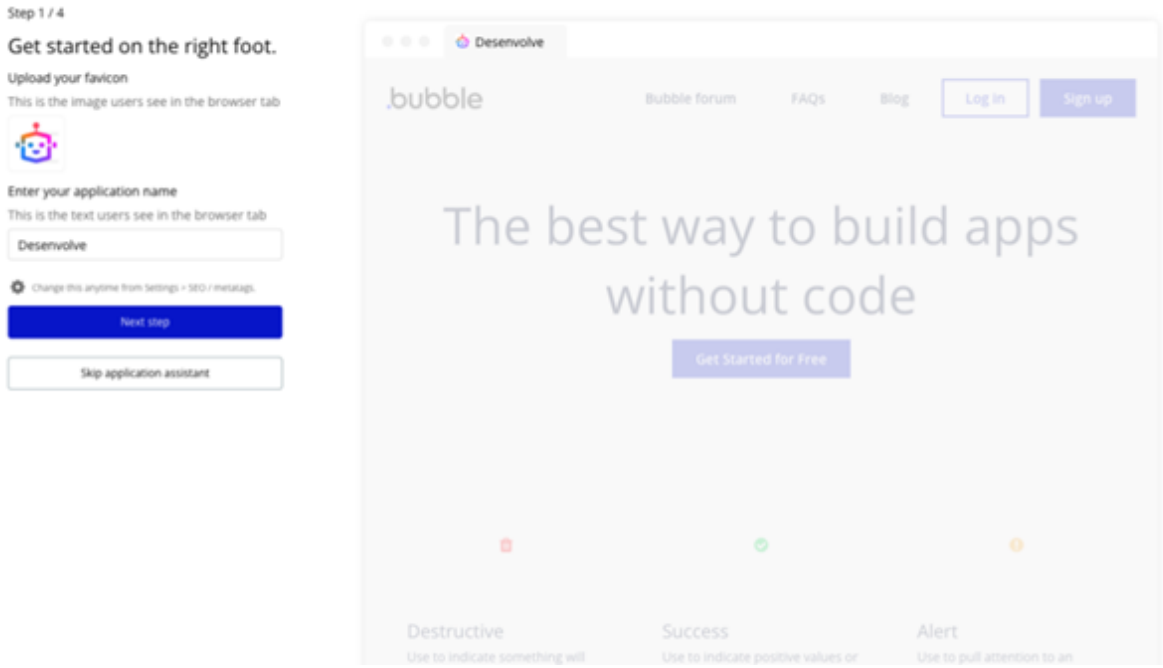
Clique em **START WITH BASIC FEATURES.**

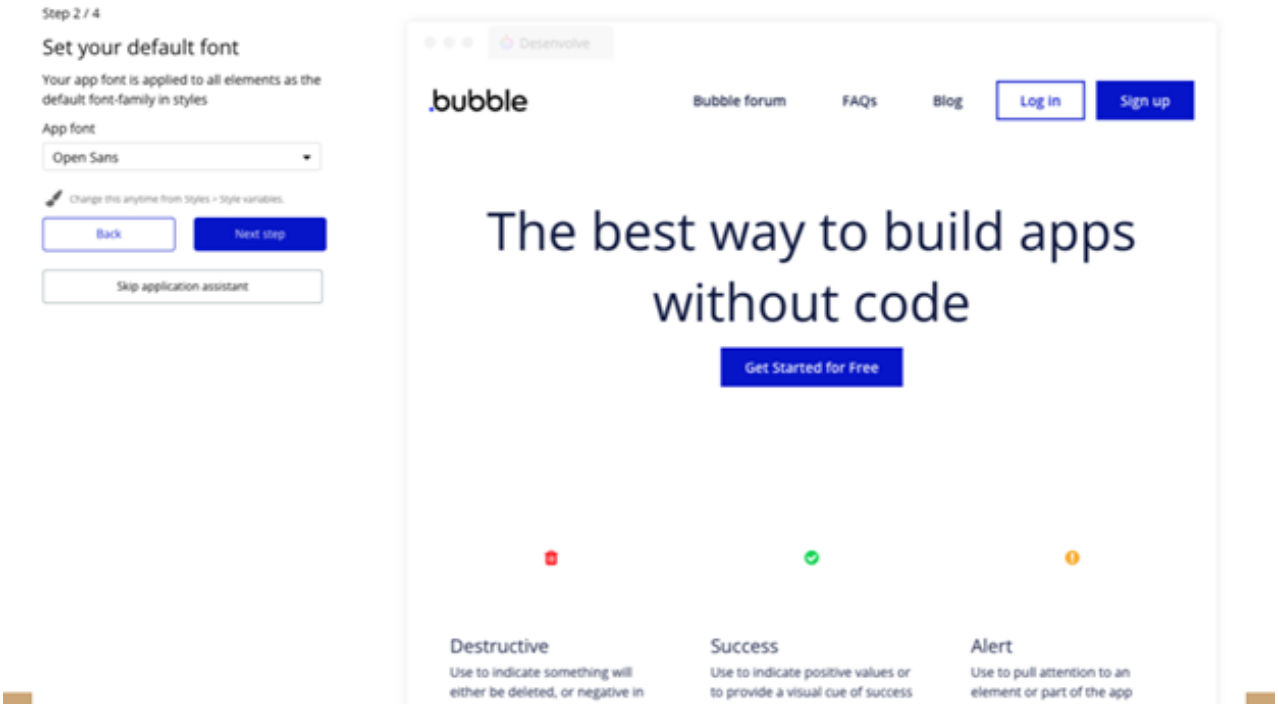


Pronto! Agora você terá 4 etapas para iniciar seu aplicativo:

1. Colocar uma imagem e um nome
2. Qual a fonte principal que deseja usar
3. Quais as cores você deseja usar
4. Quais plugins você deseja usar

Não se preocupe muito nesta parte, teremos aulas só para lidar com cada tópico. Pode seguir as opções padrão e não clicar em nada na etapa 4. Mas caso não queira visualizar as telas, basta clicar em **Skip application assistant.**





Step 3 / 4

Set your default colors

Your color variables will apply to your default elements and the component library

Primary	#0205D3
Primary contrast	#FFFFFF
Text	#091747
Surface	#FFFFFF
Background	#FFFFFF
Destructive	#FF0000
Success	#17DB4E
Alert	#FAB515

Change this anytime from Styles > Style variables.

Back | Next step

Skip application assistant

Desenvolve

bubble

Bubble forum | FAQs | Blog | Log in | Sign up

The best way to build apps without code

Get Started for Free

Destructive
Use to indicate something will either be deleted, or negative in

Success
Use to indicate positive values or to provide a visual cue of success

Alert
Use to pull attention to an element or part of the app

Step 4 / 4

Want to add plugins to your application?

Plugins allow you to define APIs, install elements, events and more

Change this anytime from Plugins > Add plugins.

Back | Get started building

Skip application assistant

Explore some official Bubble plugins

API Connector
The API lets you define your own API calls directly in the Bubble Editor and use them in your app.
Install now

Stripe
Use Stripe to process payments, including transactions, delayed charges, subscriptions, and allowing users to charge each other. Includes optional support for SCA-compliant Stripe Checkout.
Install now

Full Calendar
Add a calendar element to your application and display events from your database.
Install now

Rich Text Editor
Enable your users to write rich, formatted text in an input. Compatible with text elements and email.
Install now

Facebook
Let users log in with their Facebook accounts, and fetch their Facebook profile data.
Install now

Google
Let users log in with their Gmail account, and fetch their profile data.
Install now

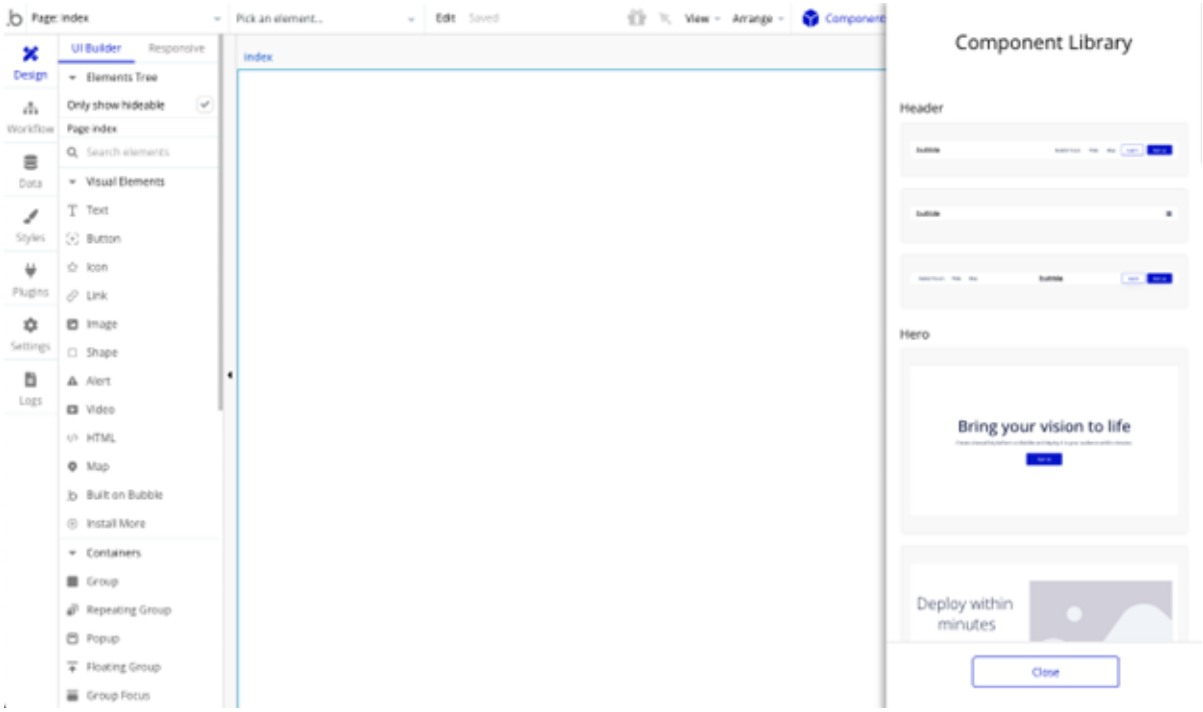
Google Analytics
Google Analytics lets you track visits on your website and get some insights on your audience for free.
Install now

Google Places
Fetch restaurant, schools, etc. from the Google Places API.
Install now

SQL Database Connector
This plugin lets you access databases by running SQL queries directly from Bubble.
Install now

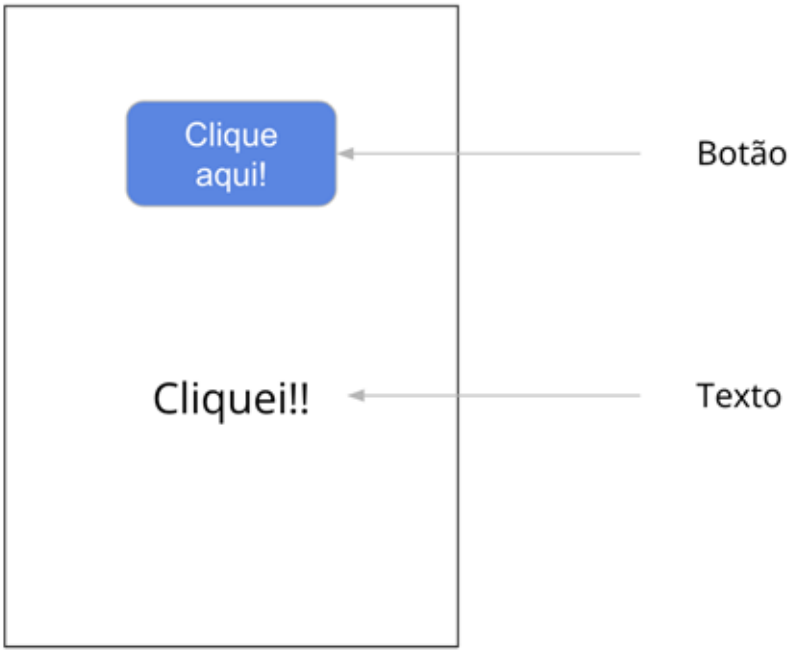
Tela principal do Bubble

Enfim, chegamos na tela de trabalho do Bubble. Explore ela toda para se familiarizar.



Exercício

Exercício prático será criar um botão escrito "Clique aqui!" e um texto "Cliquei".



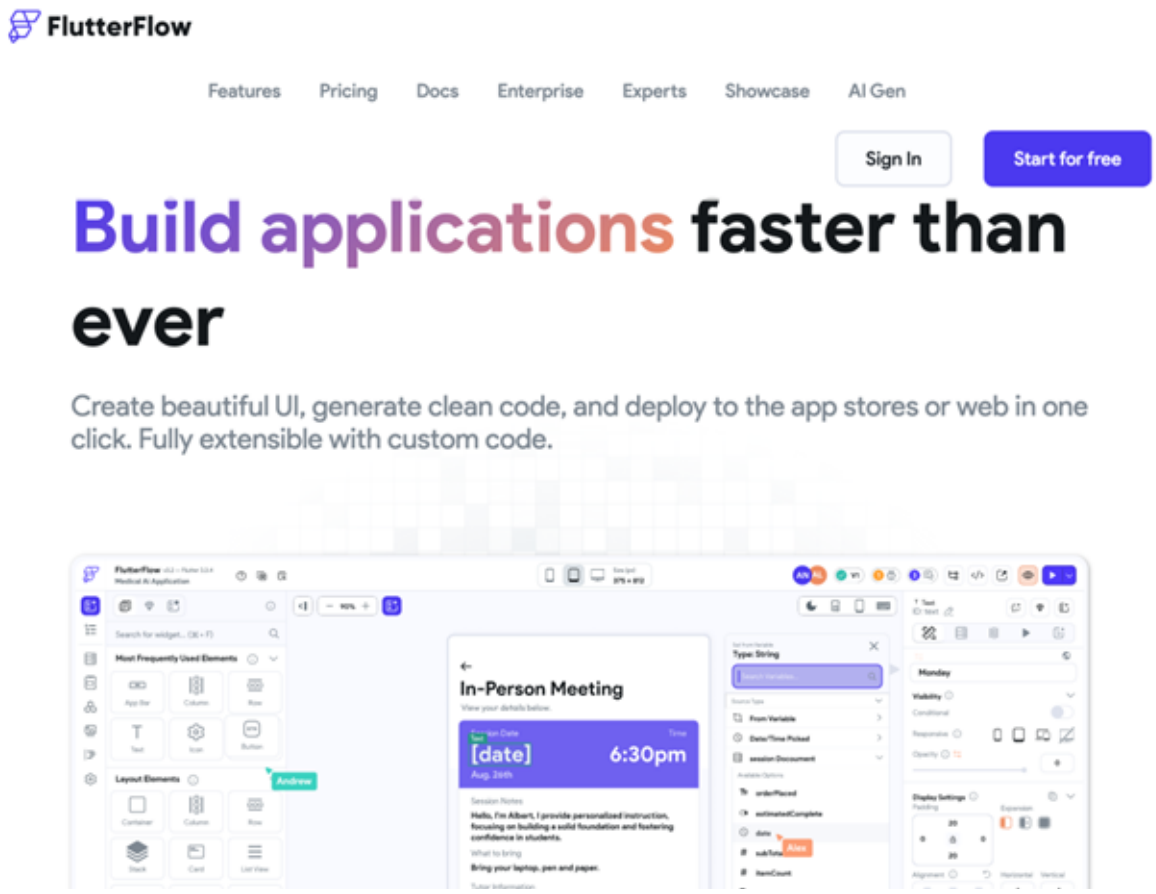
Prática II FlutterFlow

Agora que vimos um pouco sobre a plataforma do Bubble.io, vamos ver como funciona a FlutterFlow.io.

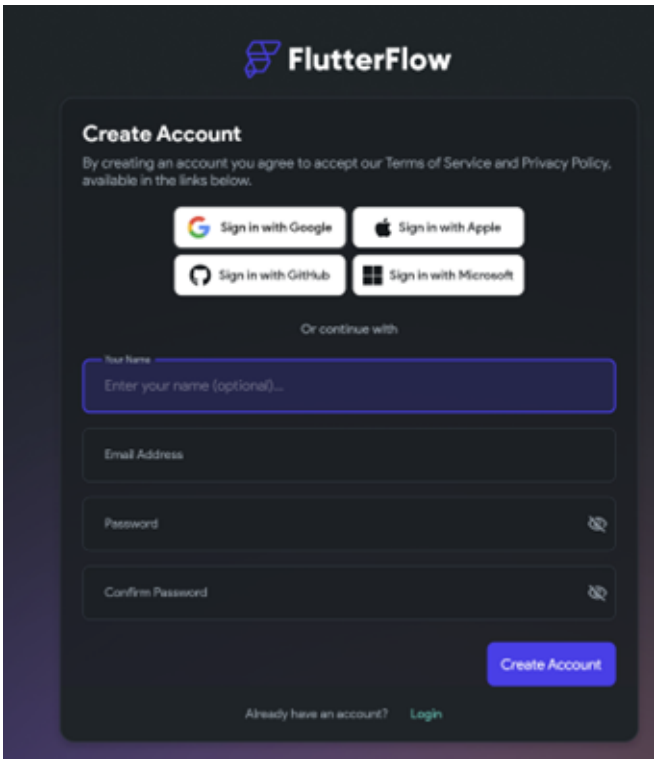
Criando uma conta no FlutterFlow

Da mesma forma, vamos criar uma conta gratuita na plataforma. Vamos seguir as seguintes etapas:

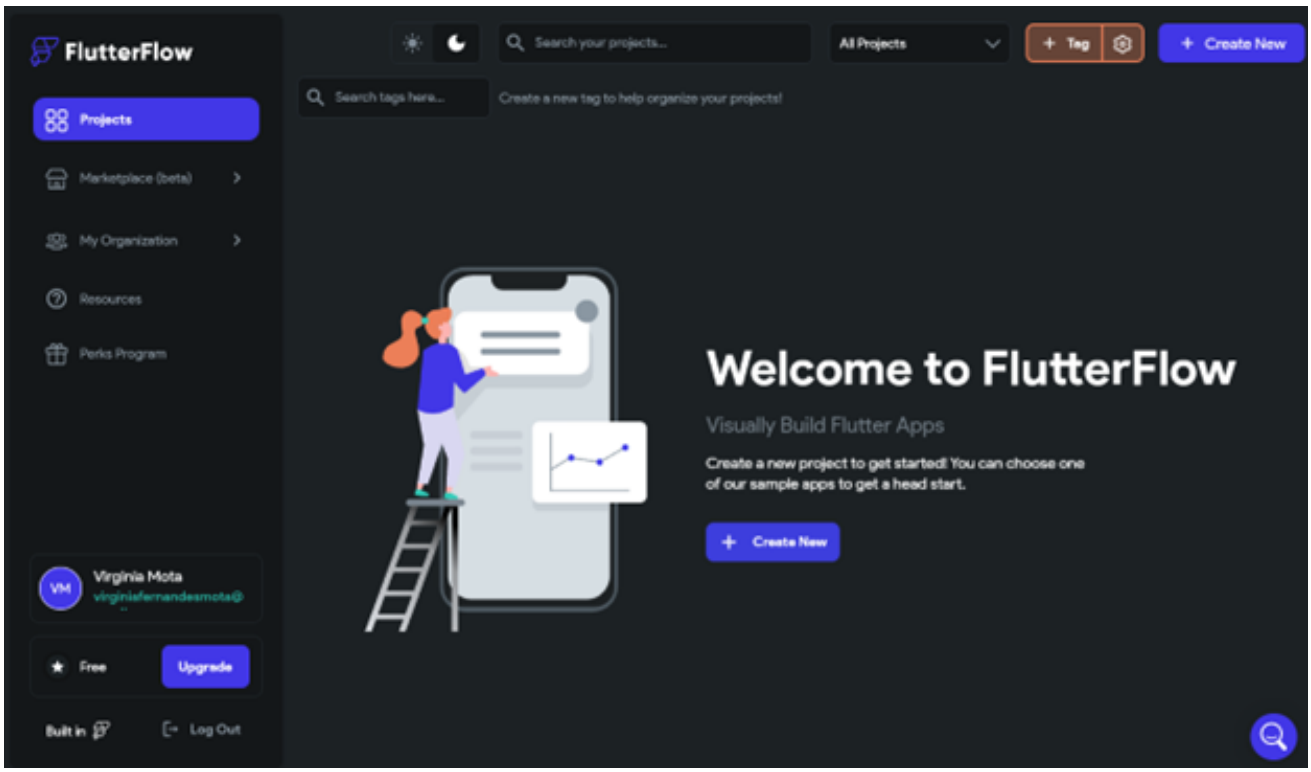
- Entrar em https://flutterflow.io/
- Ir em **Start for free**



Da mesma forma que no Bubble, a plataforma FlutterFlow permite a criação de uma conta a partir de e-mail e senha ou a partir de contas como Google. Aqui temos ainda a opção via contas Apple, GitHub e Microsoft. Pode escolher a melhor opção para você.

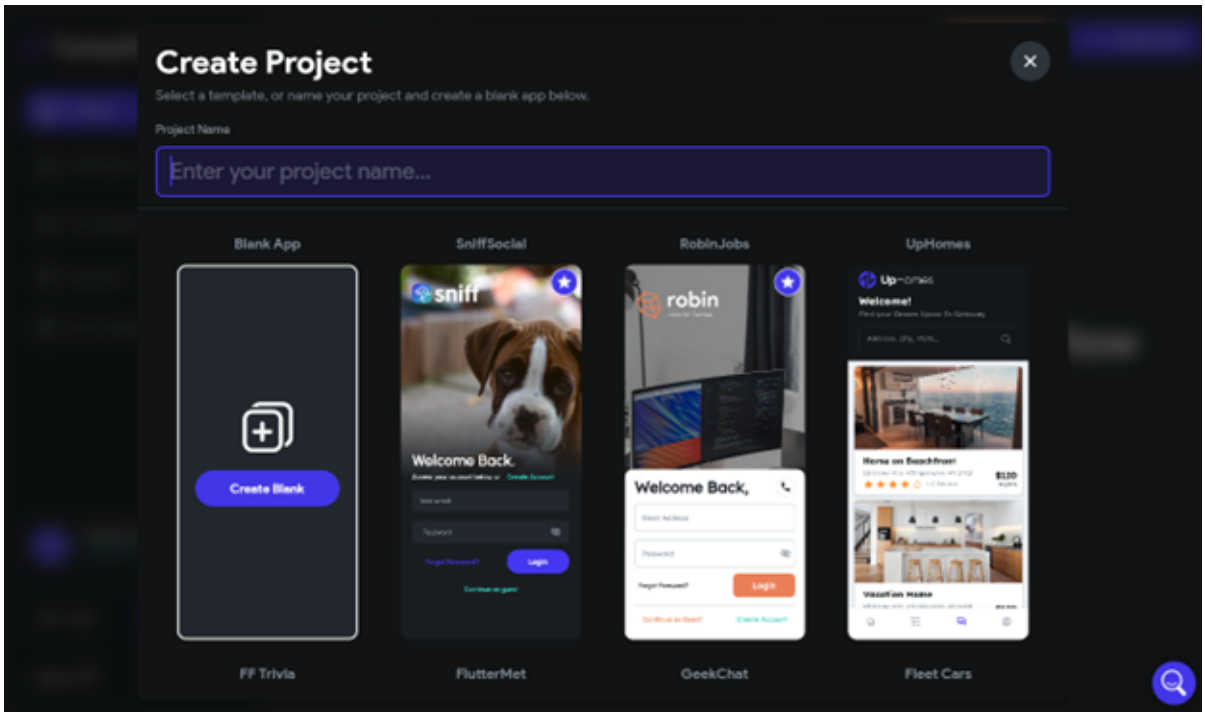


Depois disso, você irá para o Dashboard da plataforma:

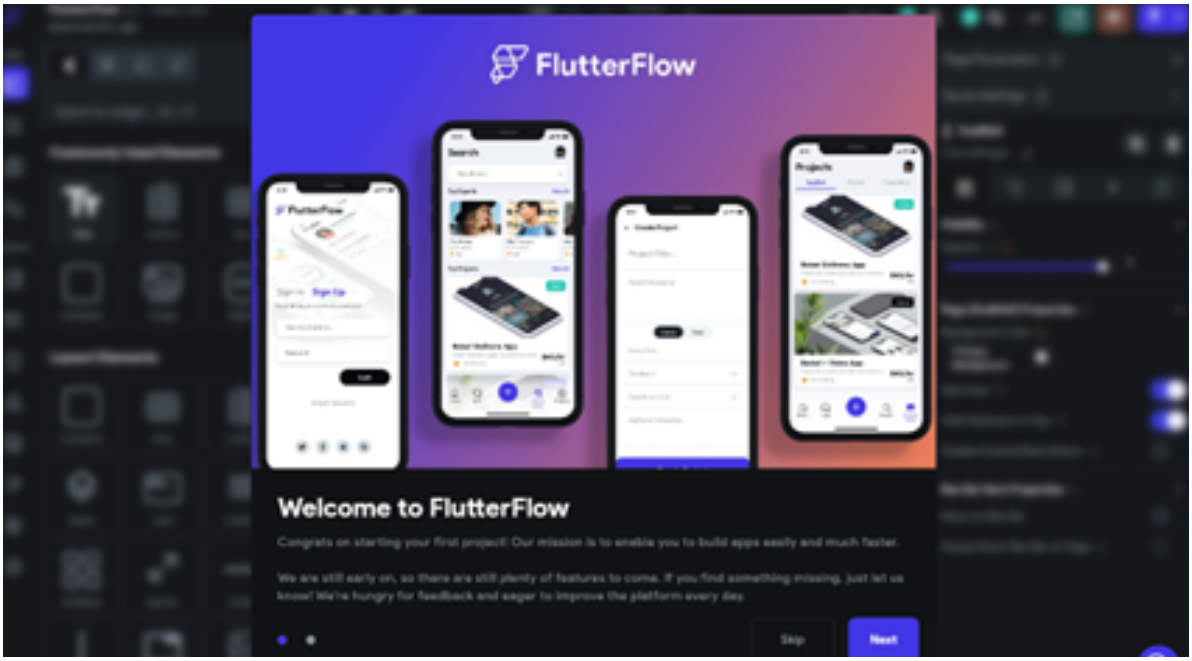


Em breve exploraremos mais deste Dashboard. Por enquanto, vamos criar um novo projeto para vermos a tela principal de trabalho do FlutterFlow. Clique em **Create New.**

Irá aparecer uma página para colocar o nome do projeto e também algumas opções. Repare como ele já mostra aplicações pré-criadas. É o que comentamos muito sobre as ferramentas no-code: Se é possível a padronização, então é possível trabalharmos com elas. Nesta página, vamos selecionar a opção **Create Blank.**



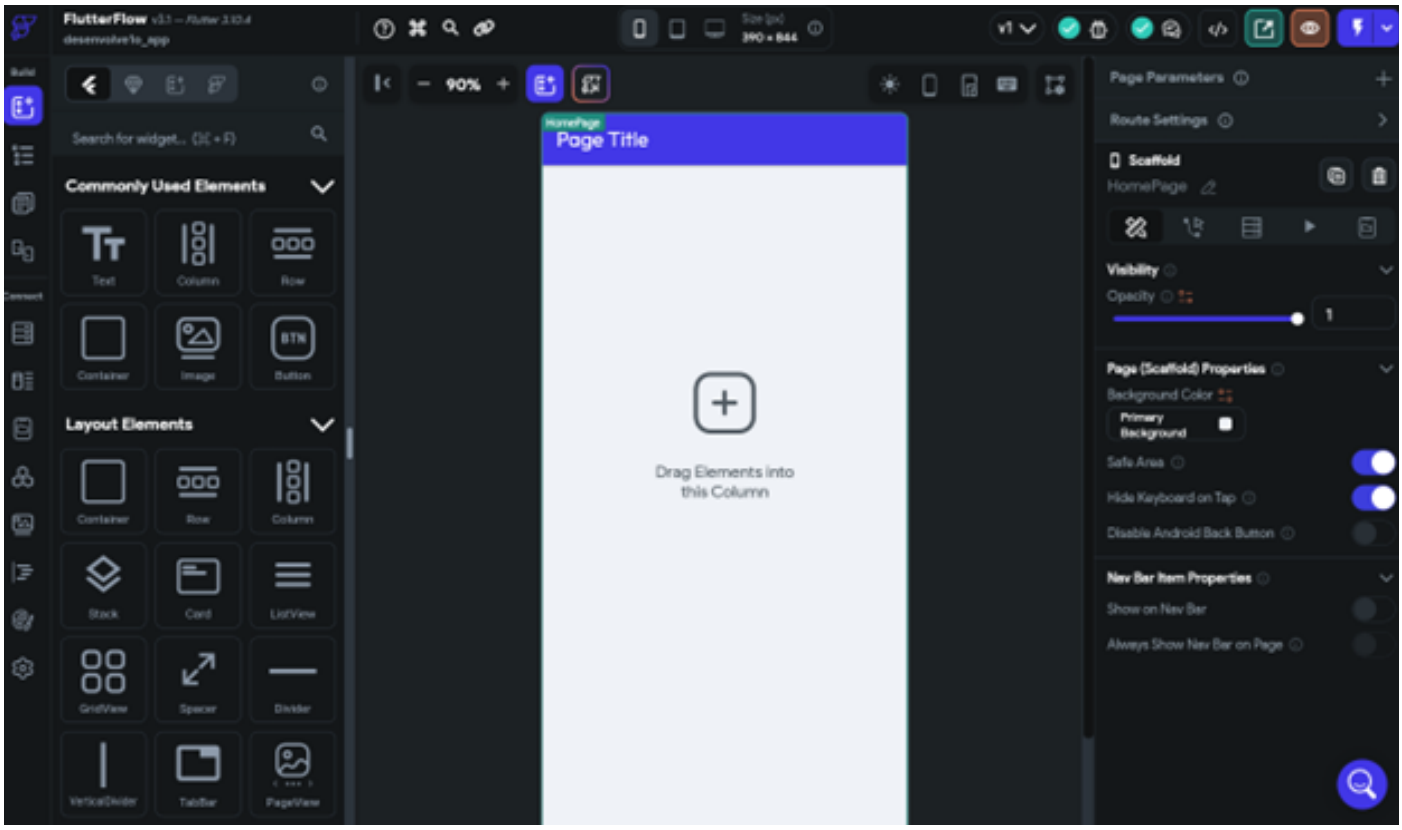
A tela de boas vindas irá aparecer.



Uma coisa muito interessante desta ferramenta é que ela possui diversos tutoriais gratuitos, distribuídos pela própria Google. Isso ajuda muito na criação de produtos com FlutterFlow.

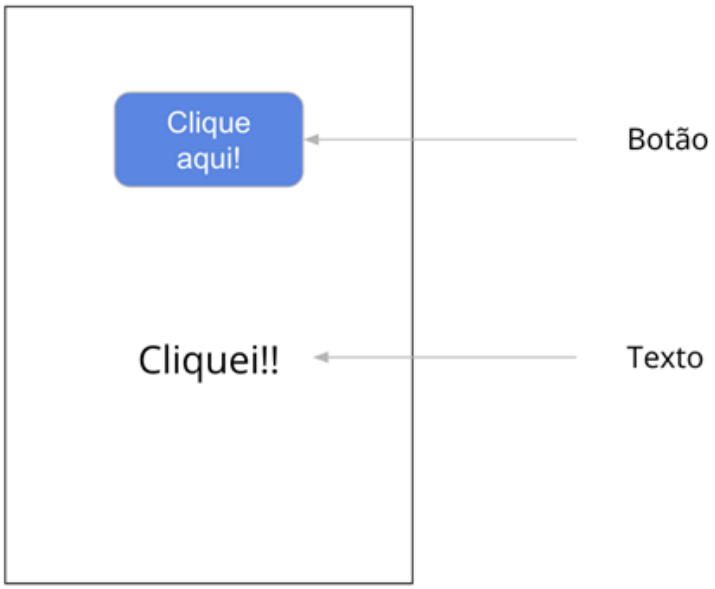
Tela principal do FlutterFlow

Enfim, chegamos na tela de trabalho do FlutterFlow. Explore ela toda para se familiarizar.



Exercício

Exercício prático 2 será criar um botão escrito "Clique aqui!" e um texto "Cliquei". (Sim, a mesma coisa que no Bubble. Faremos isso para comparar as duas ferramentas na próxima Seção)



Comparando as duas práticas

Vamos concluir esta introdução ao nosso curso de NoCode comparando duas ferramentas por meio de duas práticas simples que consistiram em adicionar um botão e um texto, tanto no Bubble quanto no Flutter Flow. Este primeiro aplicativo não realiza absolutamente nada, contendo apenas um botão e um pequeno texto. Não ajustamos sequer os textos presentes no botão ou no campo de texto, pois o objetivo era simplesmente posicionar os elementos na tela para uma análise comparativa. Agora, exploraremos as principais diferenças entre essas duas ferramentas, identificando as ênfases de cada uma no desenvolvimento do nosso aplicativo inicial.

No Bubble, observamos que podemos posicionar os elementos na tela, embora não mostre imediatamente como será a aparência no nosso sistema. Ao acessar o preview, localizado no canto superior direito, podemos ter uma ideia aproximada do visual do nosso sistema. Abaixo, há um debugger, mas perceba que ainda não apresenta o modelo de aplicativo de maneira completa. É necessário informar gradualmente ao sistema qual template estamos utilizando.

No entanto, o Bubble já realiza algumas ações automaticamente. Por exemplo, ele adiciona uma animação ao botão; ao passar o mouse sobre ele, a cor é alterada. Portanto, embora seja necessário ajustar alguns detalhes, o Bubble incorpora algumas características automaticamente.

Ao criar nossos elementos no Bubble, ao posicioná-los, o sistema nos informa que podemos ajustar sua aparência usando o menu lateral à direita. Podemos, por exemplo, clicar para editar, alterar informações, definir textos, entre outras opções. Dessa forma, o Bubble facilita a manipulação visual desses elementos, tornando mais simples movê-los de um lado para outro. Esses elementos são um pouco mais simples de trabalhar, e o sistema automaticamente acerta as informações dentro do nosso sistema.

Portanto, embora ainda tenhamos elementos de evento no workflow para trabalhar posteriormente, já temos uma representação visual simplificada do nosso sistema, o que facilita a criação do aplicativo no Bubble. No curso, abordaremos mais detalhadamente como criar eventos para esses botões, definindo fluxos com base nos elementos visuais.

Ao passarmos para o Flutter Flow, notamos uma ligeira mudança na dinâmica do "arrastar e soltar". Quando começamos a posicionar elementos, percebemos que eles não ficam exatamente onde foram soltos, e há outras opções para ajustá-los. Esse comportamento está relacionado à forma como o FlutterFlow gera código e encapsula elementos no sistema.

Uma característica interessante e divertida do FlutterFlow é que, ao adicionar o primeiro texto, ele automaticamente exibe "Hello World", um termo comum na programação e muitas vezes utilizado como o primeiro código a ser criado. Isso difere do Bubble, que simplesmente coloca "Edit Me" como o primeiro texto.

No FlutterFlow observamos que os elementos demandam um pouco mais de atenção e critério ao serem posicionados. Não basta apenas arrastar para qual-

quer lugar da tela, pois eles podem não se alinhar precisamente. É necessário considerar a estrutura do layout, como a criação de colunas, e ajustar os eixos para garantir um posicionamento adequado. Na barra lateral direita, encontramos informações relevantes para alinhar os elementos, como o eixo de encaixe e outras configurações.

Essa abordagem mais detalhada na disposição dos elementos reflete a natureza do Flutter, que é orientado à programação, e muitas das instruções geradas já seguem uma estrutura lógica, indicando a hierarquia dos componentes, como uma coluna contendo um botão, que, por sua vez, contém um texto.

Em resumo, embora possa exigir um pouco mais de esforço para posicionar os elementos no FlutterFlow, isso se deve à necessidade de considerar critérios mais específicos, como alinhamentos e espaçamentos. No entanto, ao se familiarizar com esses termos e ajustes, é possível alcançar o resultado desejado. Por outro lado, no Bubble, a abordagem é mais intuitiva, bastando arrastar os elementos para a posição desejada, resultando em menos ajustes e configurações adicionais.

A diferença de abordagem entre o Bubble, que é totalmente no-code, e o FlutterFlow, que adota uma abordagem híbrida, é evidente. No Bubble, a experiência permanece exclusivamente no-code, enquanto no FlutterFlow, a possibilidade de gerar código oferece flexibilidade adicional, permitindo manipulações mais avançadas após a criação inicial da interface visual.

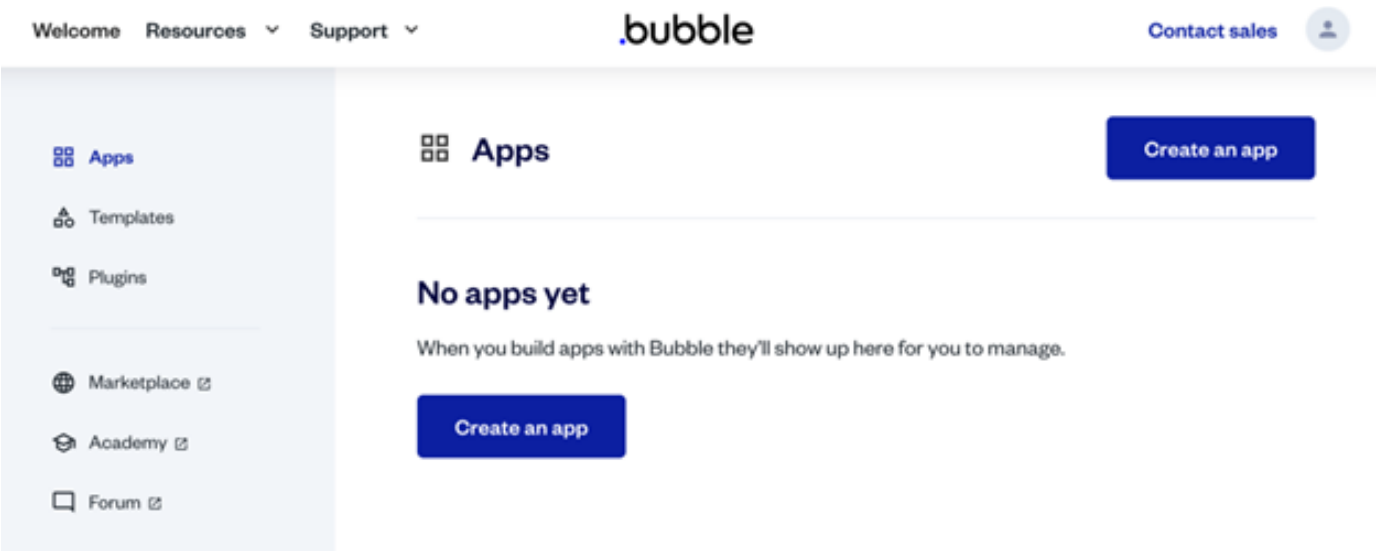
A escolha entre as plataformas dependerá, portanto, da preferência do desenvolvedor e dos requisitos específicos do projeto, considerando a abordagem totalmente no-code do Bubble e a capacidade híbrida do FlutterFlow.

Durante o curso, exploraremos a criação de um aplicativo completo no Bubble, uma plataforma completamente no-code, e no FlutterFlow, uma plataforma híbrida low-code. Essa abordagem nos permitirá comparar essas duas plata-

formas de maneira abrangente, explorando elementos semelhantes e recursos avançados, como plugins, APIs e integração de bancos de dados. Ao final, teremos uma compreensão aprofundada das diferenças e semelhanças entre essas ferramentas, auxiliando na escolha da mais adequada para diferentes necessidades de desenvolvimento.

Parte II Bubble

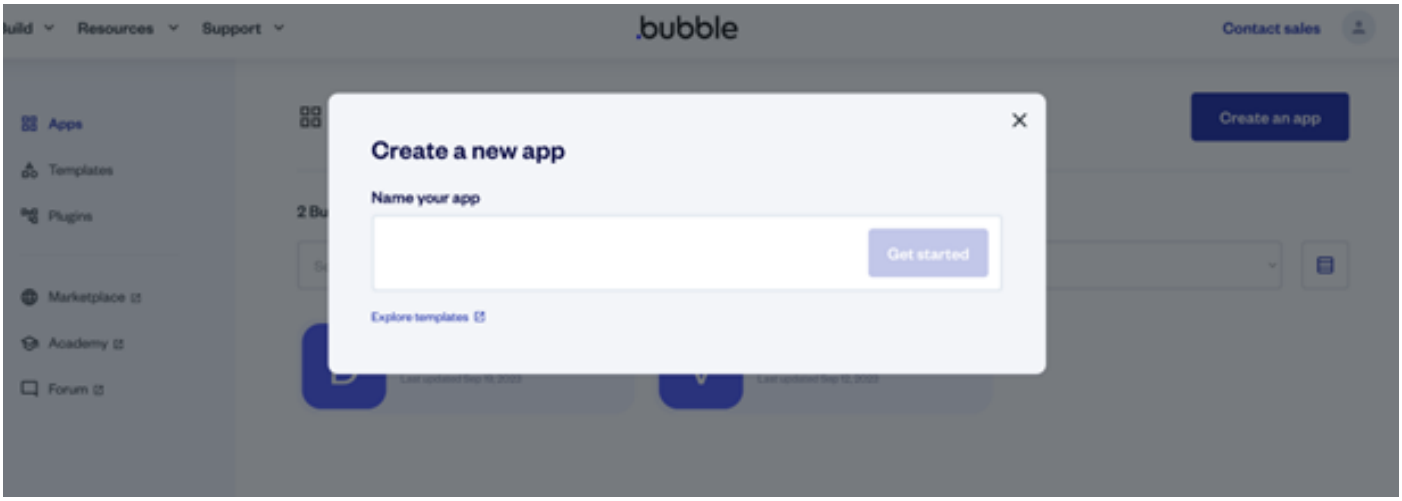
Nesta parte do curso, vamos trabalhar com a ferramenta Bubble. Você já fez a sua conta, então a primeira tela que irá aparecer para você será em:
https://bubble.io/home/apps



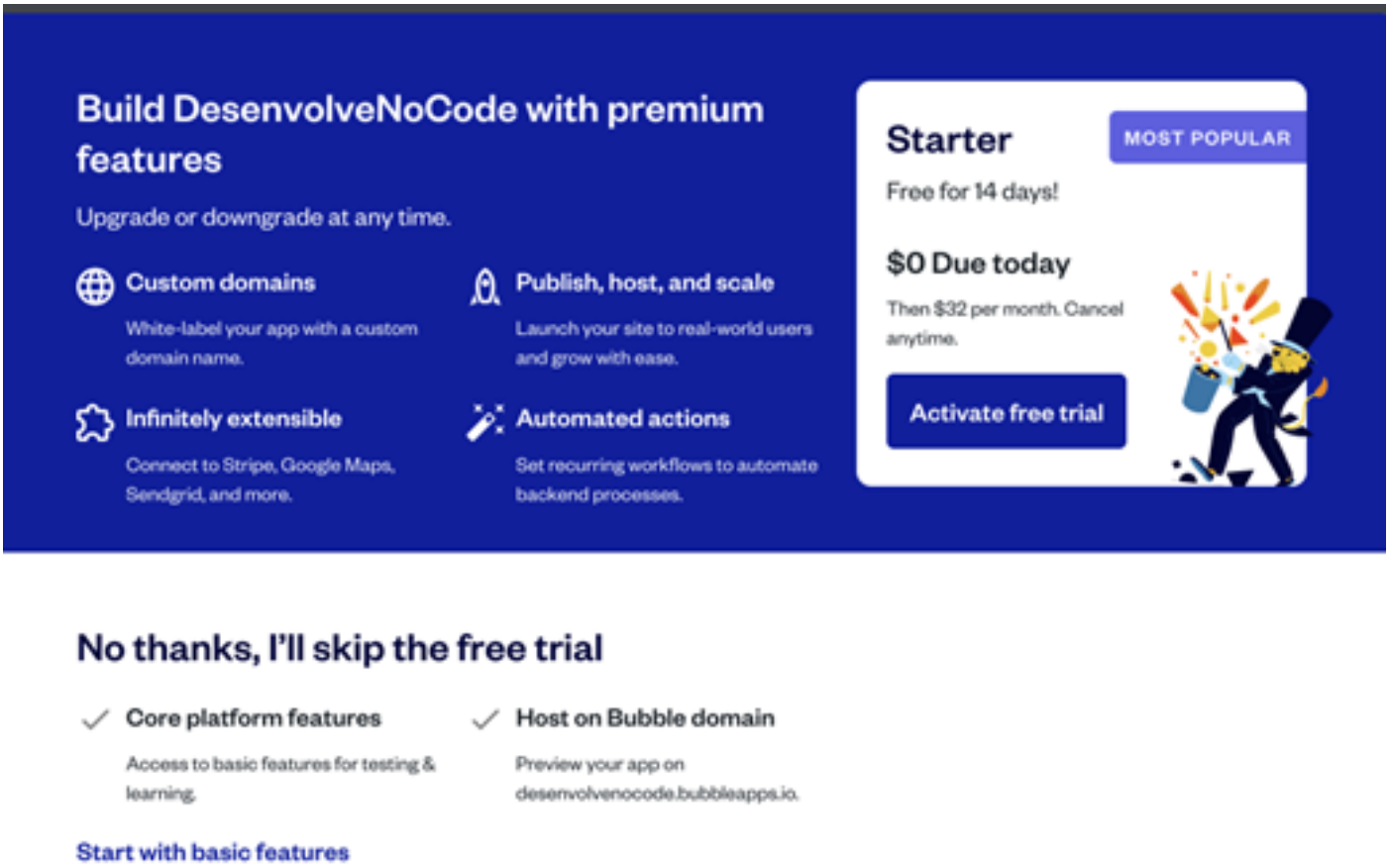
Todos os aplicativos que você criar, irão aparecer nesse Dashboard da plataforma.

O editor do Bubble

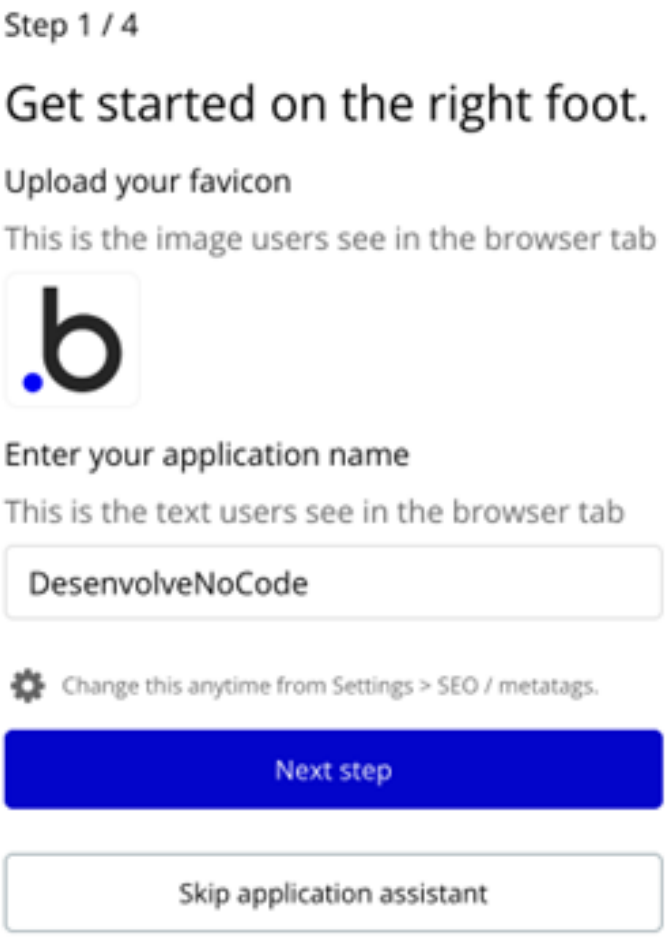
Vamos começar criando nosso novo projeto. Já fizemos isso no capítulo anterior, mas agora vamos explicar cada etapa desta criação. No Dashboard principal, clique em **Create an app** (Crie um aplicativo).



Lembre-se sempre de marcar **Start with basic features** quando a tela a seguir aparecer!



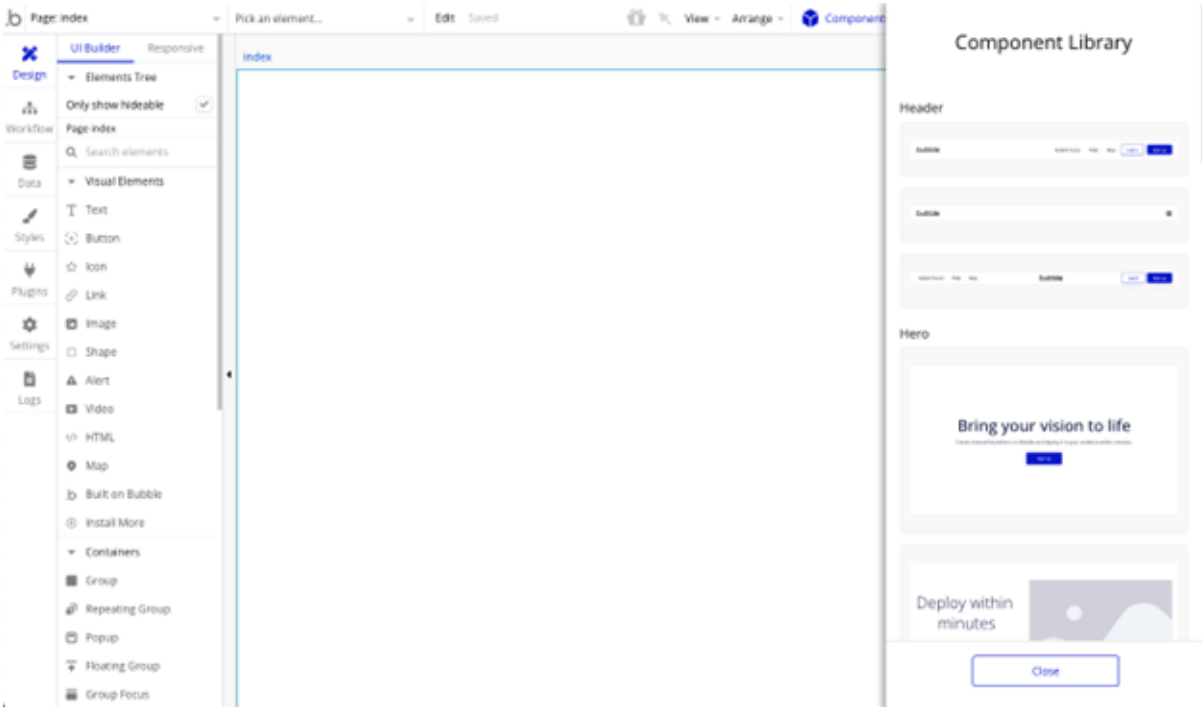
Vamos pular o assistente de configuração, clicando em **Skip application assistant.**



Pronto, agora podemos conhecer o editor do Bubble. O editor Bubble refere-se ao editor conectado a um aplicativo específico (em oposição ao painel (dashboard), onde você navega entre seus diferentes aplicativos e gerencia sua conta).

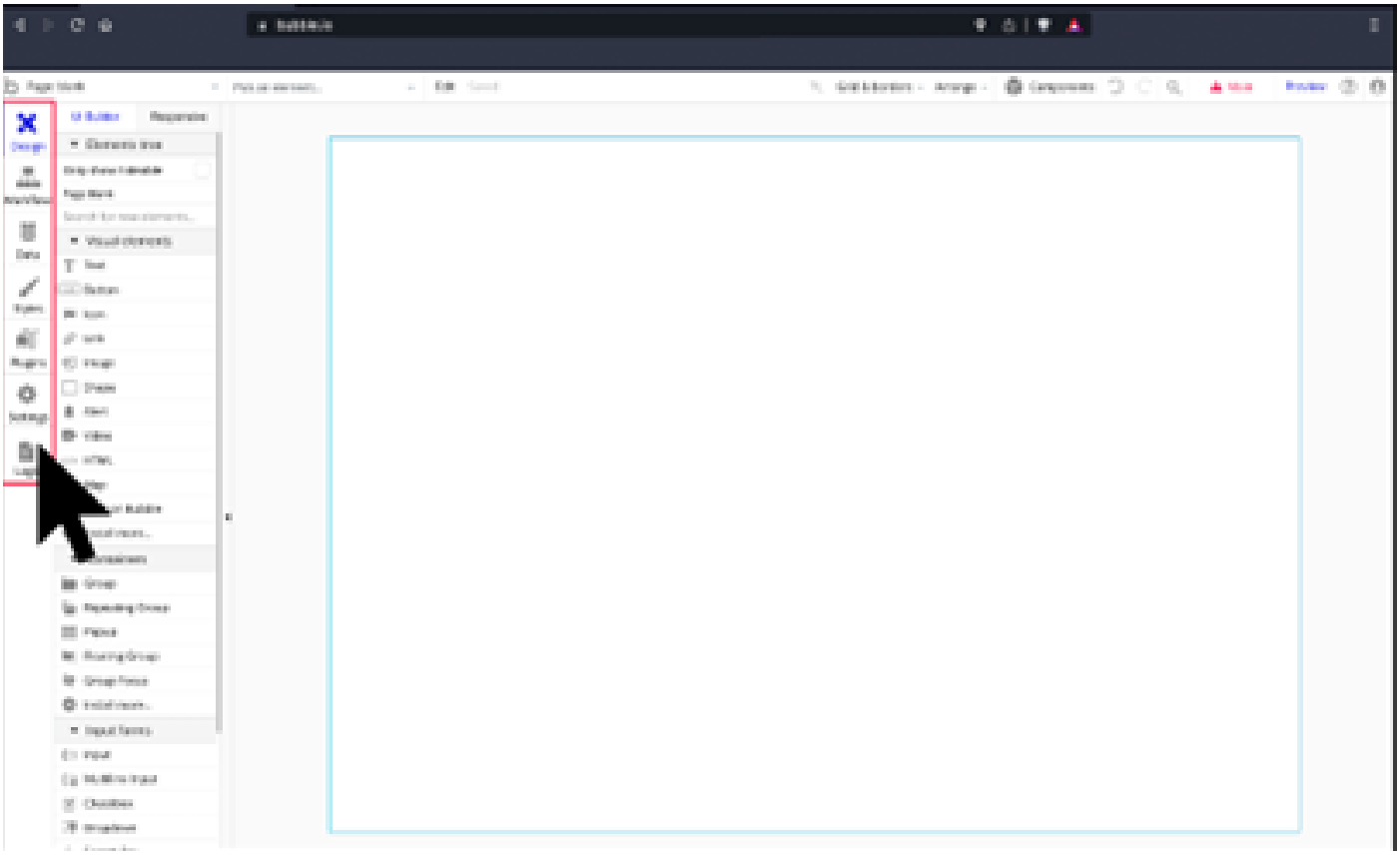
Aqui vale um aviso: Evite múltiplas guias do navegador editando o mesmo aplicativo! Abrir o editor Bubble para o mesmo aplicativo em várias guias do navegador pode resultar em comportamento inesperado. Mantenha o editor de um aplicativo limitado a apenas uma guia por vez.

O editor Bubble é como uma oficina digital onde você constrói e projeta seu aplicativo. Pense nele como uma bancada de trabalho onde você organiza, ajusta e conecta diferentes partes para que seu aplicativo tenha a aparência e funcionalidade desejadas. Vamos abordar os detalhes técnicos de design, dados e lógica nos capítulos seguintes, mas por enquanto, abordaremos a estrutura do editor e as diferentes ferramentas disponíveis.

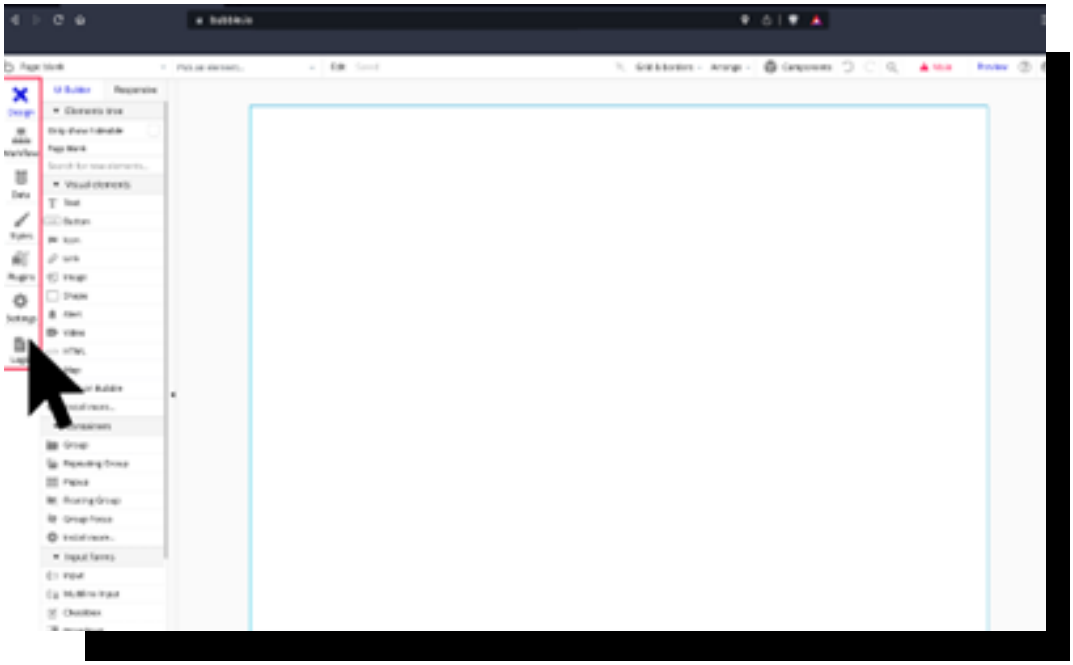


Abas e seções

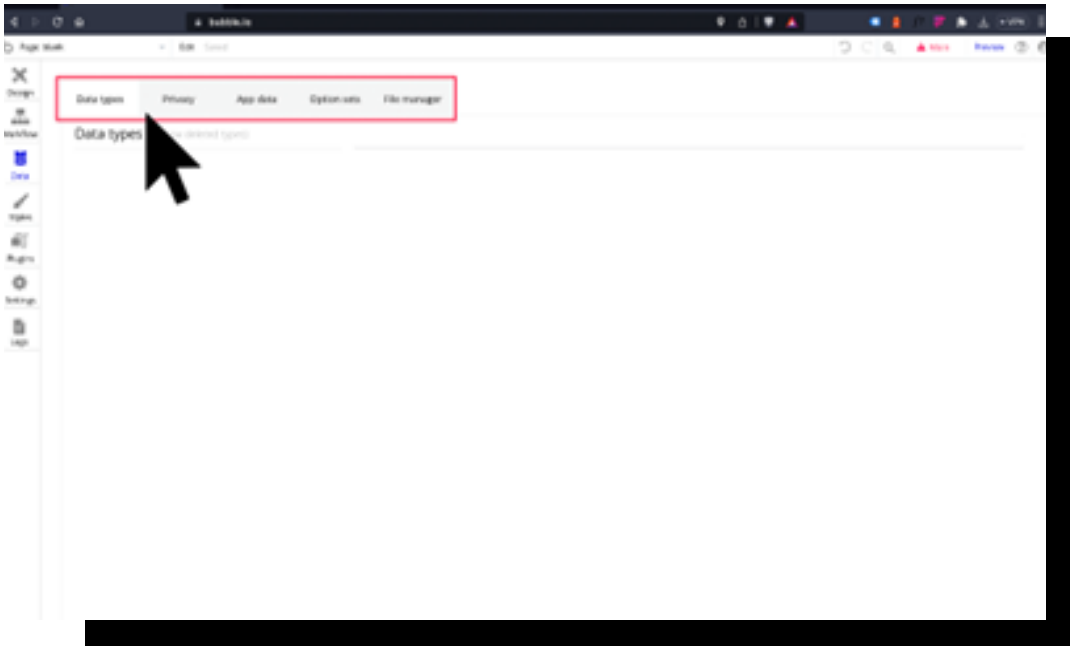
O editor do Bubble é organizado em diferentes abas **(tabs)** e seções **(sections)**, proporcionando um ambiente estruturado para construir e projetar seu aplicativo.



Abas



Seções



• **Aba Design:** Aqui é onde você estrutura a aparência visual do seu aplicativo. Você pode adicionar elementos, organizá-los na página e personalizar sua aparência. É como a tela em que você projeta a interface do usuário.

• **Aba Workflow:** Aqui, você define a lógica e a funcionalidade do seu aplicativo.

Você configura fluxos de trabalho, que são uma série de eventos e ações que ocorrem com base nas interações do usuário. Aqui é onde você dá vida ao seu aplicativo.

· **Aba Data:** Nesta aba, você gerencia a estrutura de dados do seu aplicativo. Você pode definir tipos de dados, criar campos e configurar o banco de dados do seu aplicativo. Isso é crucial para armazenar e recuperar informações.

· **Aba Privacy:** Nesta aba, você gerencia as configurações de privacidade do seu aplicativo. Você controla quem pode ver e modificar dados dentro do seu aplicativo. Configurações de privacidade são essenciais para proteger as informações do usuário.

· **Aba SEO/Metatags:** Aqui, você pode otimizar seu aplicativo para motores de busca personalizando configurações de SEO e meta tags. Isso ajuda a melhorar a descoberta do seu aplicativo na web.

· **Aba Plugins:** Se você deseja estender a funcionalidade do seu aplicativo, pode usar plugins. A aba Plugins permite adicionar, configurar e gerenciar plugins, que são componentes pré-construídos que adicionam recursos específicos ao seu aplicativo.

· **Aba Settings:** Esta aba contém configurações gerais para o seu aplicativo. Você pode configurar coisas como o nome do aplicativo, domínio, responsividade e outras configurações globais.

· Visualização e Versões Ao Vivo: Você pode visualizar como seu aplicativo se parece e funciona usando o modo de visualização. Além disso, você pode implantar **(deploy)** seu aplicativo ao vivo quando estiver pronto para acesso público.

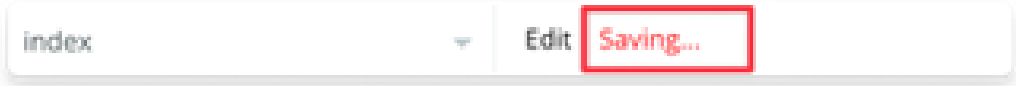
Lembre-se, essas são categorias amplas, e cada aba contém várias seções e opções para ajustar detalhes específicos do seu aplicativo. Conforme avança, você mergulhará mais fundo em cada aba para construir aspectos específicos da sua aplicação.

Principais recursos

Alguns recursos importantes dentro da ferramenta que iremos utilizar são:

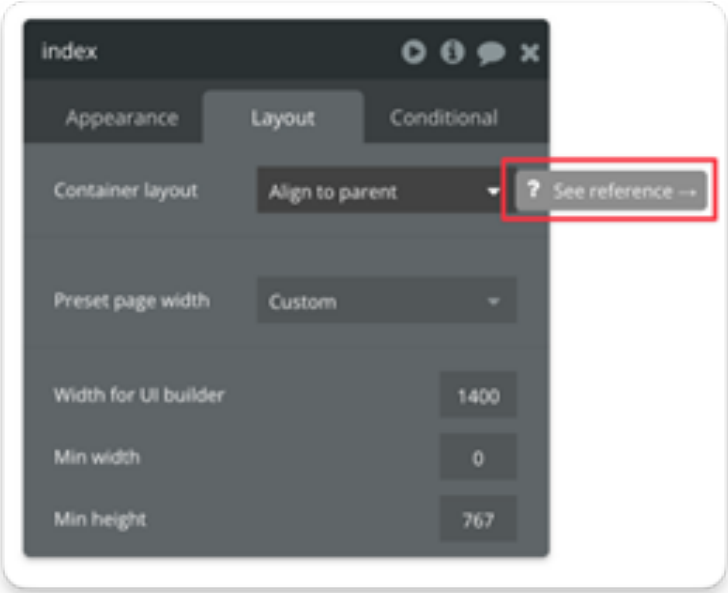
• **Salvamento Automático em Tempo Real:** O Bubble salva automaticamente seu aplicativo em tempo real. Qualquer modificação em um elemento, ação, configuração, etc., será salva instantaneamente. Você verá uma notificação de 'salvando' **(saving)** na barra superior por um instante, que mudará para 'salvo' **(saved)** assim que o processo de salvamento for concluído.

• **Erros de Salvamento:** Ocasionalmente, você pode encontrar um erro ao tentar salvar as alterações mais recentes. Isso geralmente ocorre devido a uma conexão fraca e será resolvido assim que seu dispositivo estiver online novamente. Se a notificação "Salvando..." na barra de ferramentas do Bubble ficar vermelha, significa que não pode se comunicar com sucesso com o servidor. Nesse caso, pause seu trabalho para reduzir o risco de perda de dados.

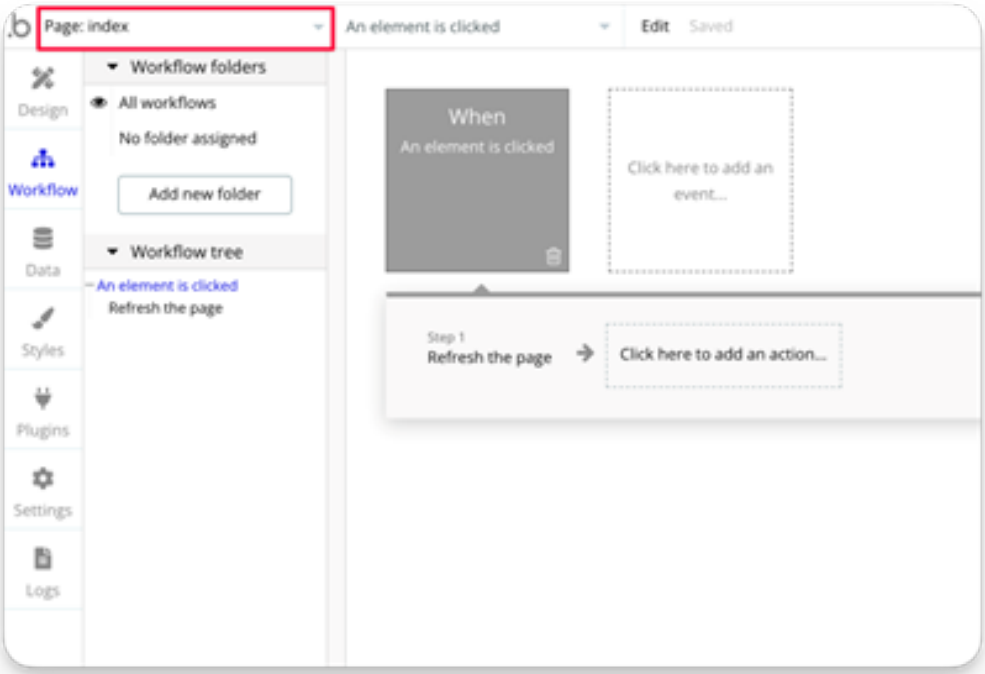


• **Desfazer e Refazer:** Você pode desfazer ou refazer quaisquer alterações feitas em seu aplicativo. Use os ícones correspondentes na barra superior para reverter ou reaplicar modificações. Se estiver fazendo alterações significativas ou experimentando com seu aplicativo e quiser reverter para uma versão específica, recomendamos usar o recurso de versões.

• **Atalhos para o Manual de Referência:** Você pode acessar a documentação de referência central a partir do editor através da dica de ferramenta 'Ver referência' **(see reference)**. Quando você paira sobre um ícone ou campo por alguns segundos, a dica de ferramenta aparecerá. Clicar na dica de ferramenta abrirá uma nova guia e o levará ao local certo na documentação. Se você já tiver a documentação aberta, será redirecionado instantaneamente para o local certo.



• **Seletor de Página:** No canto superior esquerdo do editor do Bubble, você encontrará o seletor de página. Aqui, você navega entre as diferentes páginas e elementos reutilizáveis do seu aplicativo, além de acessar a documentação de referência.

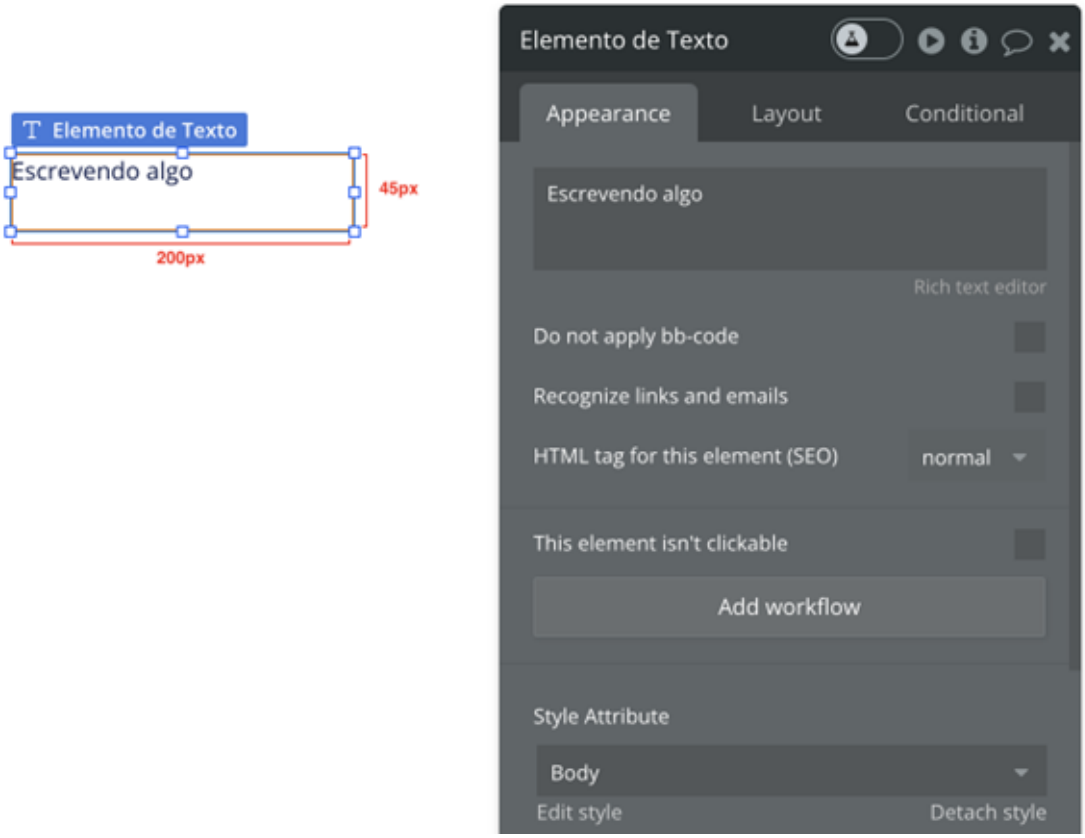


• **Menus Editar, Visualizar e Organizar:** Os três menus na barra de ferramentas superior oferecem ferramentas e configurações contextuais relevantes para o trabalho que você está fazendo.

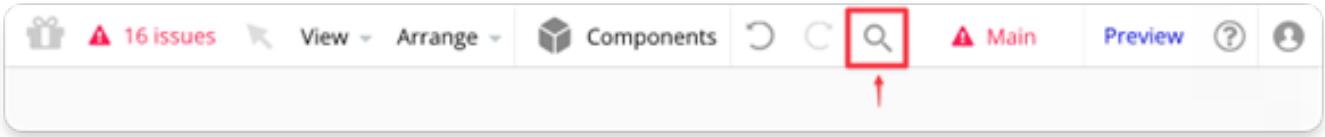
Ferramentas

Vamos dar uma olhada nas ferramentas mais amplamente utilizadas no editor.

O editor de propriedades **(property editor)** é usado para alterar as propriedades dos elementos na página.

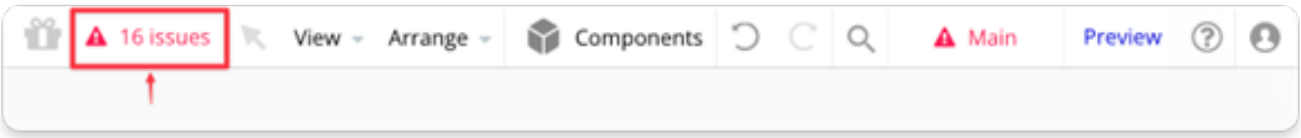


A ferramenta de pesquisa **(search tool)** é uma ferramenta avançada para buscar uma ampla variedade de objetos diferentes em seu aplicativo.



O rastreador de problemas **(issue tracker)** ajuda a identificar e corrigir problemas. Ele monitora dinamicamente seu aplicativo em busca de possíveis erros ou proble-

mas. Conforme você faz modificações, ele se atualiza automaticamente para destacar novos problemas ou limpar aqueles que foram resolvidos. Esse feedback em tempo real garante que você possa lidar imediatamente com quaisquer problemas potenciais, ajudando a manter a integridade e funcionalidade de sua aplicação.



À medida que você for avançando na criação de aplicativos, você irá explorar a fundo cada uma dessas funcionalidades.

Pré-visualização e Implantação

O Bubble vem com uma ferramenta integrada **(built-in tool)** para visualizar **(preview)** seu aplicativo. Este ambiente de desenvolvimento possui um banco de dados completamente independente e permite que você experimente seu aplicativo e todas as suas funcionalidades antes de implantá-lo para usuários ao vivo.

O Bubble é um editor "WYSIWYG" **(What you see is what you get)**, o que significa que o que você vê no editor geralmente é o que você verá no aplicativo final. Toda alteração que você faz dentro do Bubble é salva instantaneamente, permitindo que você visualize seu aplicativo em evolução conforme necessário.

Para visualizar seu aplicativo, clique no botão **Preview** no canto superior direito da barra de ferramentas do Bubble. Isso abrirá uma nova aba onde seu aplicativo será carregado.

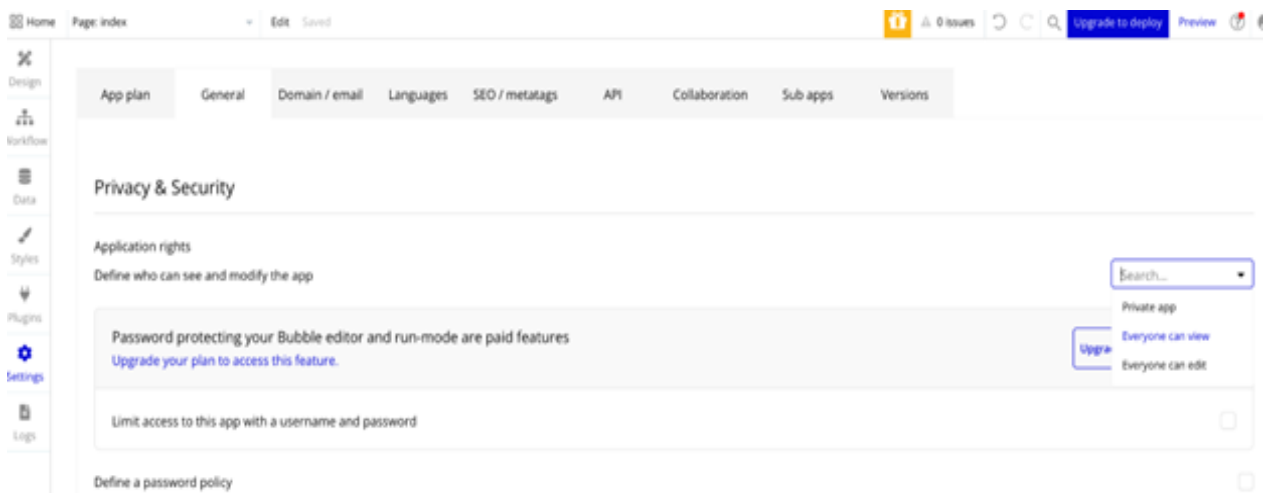
Implantar **(deploy)** seu aplicativo significa transferir ou atualizar a versão atual de sua aplicação do ambiente de desenvolvimento para o ambiente ao vivo (às vezes chamado de produção), tornando-a acessível aos usuários finais. Em outras palavras, este é o ponto em que você publica seu aplicativo para usuários que realmente vão usá-lo.

Embora você tenha a liberdade de implantar seu aplicativo sempre que quiser, o Bubble exige que o rastreador de problemas não esteja indicando nenhum problema. Se o rastreador de problemas estiver mostrando algum problema, eles precisam ser resolvidos primeiro para permitir a implantação.

Implantar seu aplicativo é um marco significativo, marcando o momento em que suas últimas alterações entram em vigor para seu público. É um evento que merece um planejamento cuidadoso, pois traz suas atualizações e melhorias diretamente para seus usuários finais. Trataremos disso em breve.

Liberando para visualização

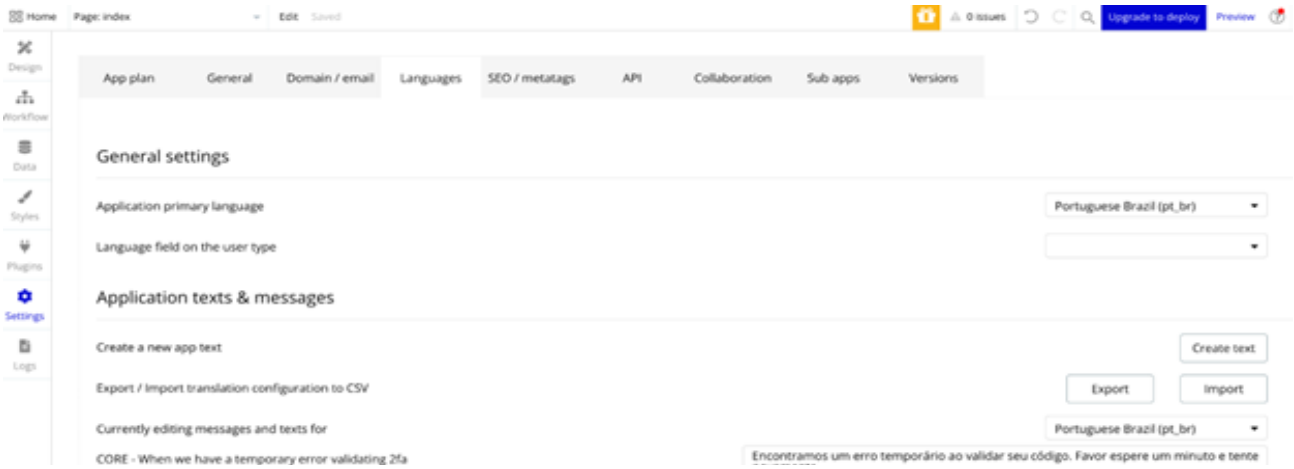
Para liberar seu link do editor para visualização: Settings - General - Application Rights - Everyone can view.



Feito isto, basta copiar o link da aba que você está trabalhando e enviar, seja para entregar um exercício, seja para mostrar para alguém, ou até para pedir ajuda.

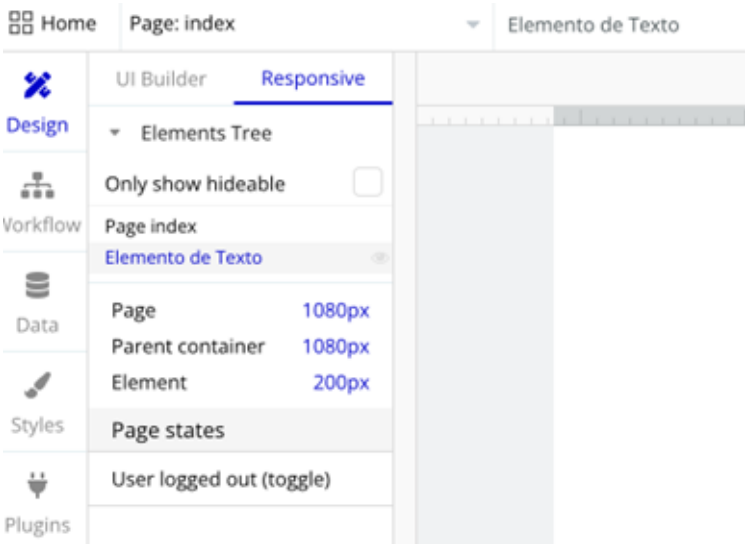
Traduzindo para Português

Para utilizar português como linguagem principal do seu aplicativo, vá em **Settings - Languages** e mude os campos **Application primary language** e **Currently editing messages and texts** para Portuguese Brazil (pt_br). Isso NÃO irá traduzir a ferramenta Bubble para português, mas vai traduzir as mensagens padrão que o usuário irá ver ao utilizar o aplicativo.



Configuração de responsividade

Ter um design responsivo nada mais é que um site ou aplicativo onde seu conteúdo permanece igual, independente do dispositivo que está sendo acessado. Assim, você consegue com que o que está sendo desenvolvido funcione bem para qualquer tamanho de tela.



Ao lado de **UI Builder**, temos a seção **Responsive**. Clique nela. Sua tela irá mudar e aparecerá um botão **Make responsive**.

Quando estivermos trabalhando nosso primeiro aplicativo, vamos configurar a responsividade a partir desta tela.

Construindo seu primeiro aplicativo

Iniciar sua jornada no desenvolvimento de aplicativos pode parecer como aprender uma nova linguagem, especialmente com a linguagem intrincada e terminologias técnicas usadas na programação tradicional. O Bubble utiliza terminologia intuitiva e autoexplicativa, como "things" (coisas), "workflows" (fluxos de trabalho) e "conditions" (condições), permitindo que você se concentre em dar vida às suas ideias. Principalmente pensando em um dos pontos importantes do no-code: fazer protótipos rápidos para provas de conceito.

O primeiro passo é PLANEJAR o aplicativo. À primeira vista, o processo de planejá-lo pode parecer assustador, mas o design intuitivo e os mecanismos de fluxo de trabalho do Bubble são projetados para guiar iniciantes pelo processo de criação de maneira tranquila.

Começar com uma visão clara é essencial. Divida sua ideia em funcionalidades principais e nas interações que você espera que os usuários tenham com seu aplicativo. Esboçar um wireframe básico (protótipo da interface) no papel ou usando uma ferramenta digital pode ajudar a visualizar o layout e a jornada do usuário. Mesmo que sejam apenas caixas e setas ásperas, isso lhe dará um roteiro a seguir.

Lembre-se, a chave para um aplicativo bem-sucedido não está apenas em sua funcionalidade, mas também em sua experiência do usuário. Considere o usuário final em cada estágio do desenvolvimento.

Planejando os recursos (features)

Antes de começar a construir, você precisa decidir que tipo de recursos **(features)** seu aplicativo precisa. A maioria das ideias de aplicativos começa como uma visão geral do que o aplicativo deveria fazer. Você então pega essa visão e a divide nos recursos necessários para torná-la realidade. Quando você decide sobre os recursos necessários, pode começar a planejar como esses recursos devem funcionar em uma interface real.

Ao criar um aplicativo, pense sempre em versões. É útil pensar no seu aplicativo em versões para saber exatamente o que você está construindo. É muito fácil ter novas ideias à medida que você trabalha e se desviar do seu plano inicial. Ao planejar versões, você pode manter o foco na entrega de um conjunto central de recursos primeiro e, em seguida, aprimorar e expandir progressivamente seu aplicativo com base no feedback e nas necessidades reais dos usuários.

Por exemplo, você pode pensar nas suas versões da seguinte forma:

1. MVP (Produto Mínimo Viável): Esta é a versão mais básica do seu aplicativo, contendo apenas os recursos essenciais necessários para torná-lo funcional. O objetivo aqui é lançar algo no mercado rapidamente para testar o conceito, reunir feedback e identificar melhorias e até mesmo pivôs potenciais.
2. Versão 1.0: Uma vez que você validou a ideia com seu MVP, é hora de refinar. Esta versão terá uma experiência do usuário (UX/UI) aprimorada e possivelmente alguns novos recursos com base no feedback do seu MVP. Você também resolverá quaisquer bugs ou problemas significativos identificados durante a fase do MVP.
3. Versão 2.0 (e além): À medida que sua base de usuários cresce e você coleta mais feedback, começará a identificar mais áreas para aprimoramento e expan-

são. Isso pode envolver a adição de novos recursos, melhoria dos existentes ou expansão para novos mercados ou plataformas.

Lembre-se, é tentador querer adicionar todos os recursos que você puder pensar desde o início, mas a contenção pode ajudar a levar seu aplicativo ao mercado conforme o cronograma. Toda vez que uma boa ideia surgir, você pode, é claro, anotá-la e planejá-la para uma versão futura.

Além disso, adotar uma abordagem passo a passo permite que você se adapte e pivote com base no feedback do mundo real, em vez de suposições. Os usuários muitas vezes usarão seu aplicativo de maneiras que você não havia antecipado ou expressarão necessidades que você não havia considerado. Ouvindo-os e iterando sobre seu produto, você aumenta as chances de sucesso do seu aplicativo a longo prazo.

MVP

A primeira versão de um aplicativo é frequentemente chamada de Produto Mínimo Viável (MVP). O objetivo é criar algo funcional que aborde efetivamente a necessidade ou problema principal que você identificou, sem se perder em recursos ou detalhes secundários.
Uma forma para abordar isso pode ser:

• **Identifique o problema central:** Qual é o problema principal que seu aplicativo visa resolver? Compreender isso ajudará a priorizar os recursos absolutamente necessários para o seu MVP.
• **Histórias do usuário:** Crie histórias do usuário para visualizar como os usuários interagirão com seu aplicativo. Por exemplo: "Como usuário, quero poder criar novos contatos para poder acompanhar meus clientes."
• **Priorize os recursos:** Liste todos os recursos que você imagina para o seu aplicativo e, em seguida, classifique-os com base em sua importância para o problema central. Concentre-se nas principais prioridades para o seu MVP.
• **Esboce ou crie wireframes:** Use esboços ou wireframes para mapear a jornada

do usuário e a interface. Essa ajuda visual ajudará você a entender como os recursos se integram e fluem juntos.

• **Ciclo de feedback:** Compartilhe sua lista de recursos priorizados e wireframes com potenciais usuários, partes interessadas ou colegas de equipe. Coleta feedback e ajuste seu plano conforme necessário.

• **Evite a expansão de recursos:** É tentador adicionar mais recursos conforme você planeja, mas mantenha o foco no objetivo principal. Recursos adicionais podem sempre ser adicionados em versões posteriores, após o teste e validação do MVP.

• **Planeje a coleta de feedback:** Como o propósito do MVP é testar e validar sua ideia, tenha mecanismos (como formulários de feedback, ferramentas de análise, etc.) para coletar feedback dos usuários após o lançamento.

Não tenha medo de deixar de fora recursos que não são absolutamente necessários para validar sua ideia. Você sempre pode adicioná-los posteriormente.

Design - UX e UI

Basicamente, o design é a aparência do seu aplicativo! O processo de design de um aplicativo pode ser categorizado em duas disciplinas principais: **UI (user interface) e UX (user experience).**

A interface do usuário (frequentemente chamada de UI) é a aparência geral de um aplicativo, incluindo layout de página, cores, tipografia, ícones, imagens e outros elementos. Projetar uma interface do usuário diz respeito a tornar o aplicativo visualmente atraente e alinhado com a identidade da marca.

O design de experiência do usuário (UX) concentra-se em como o usuário percebe o aplicativo. É fácil navegar e usar? É lógico em sua estrutura? Resolve problemas de forma eficiente? O design de UX está enraizado na compreensão e otimização da jornada do usuário, garantindo que a interface seja intuitiva, eficiente e amigável ao usuário. Além disso, também deve-se levar em consideração questões de acessibilidade para usuários (leitores de tela, contraste entre cores, etc).

Então, a partir disso, podemos elaborar alguns pontos nos quais você pode se concentrar no seu processo de design:

1 - Um aplicativo deve ter uma aparência agradável e um design consistente para representar sua marca.
2- Deve ser fácil de entender, navegar e usar.

O design é, é claro, uma disciplina ampla e profissional. Se você não tem formação em design, você pode estudar o trabalho de grandes designers para obter ideias e inspiração para o seu próprio trabalho.

Uma forma também de facilitar é começar pelo básico. Foque primeiro nos componentes básicos:

• **Botões:** Estes vêm em vários tamanhos e estados. Projete para ações primárias, secundárias e terciárias. Um botão "Cancelar" muitas vezes é menos proeminente do que um botão "OK", por exemplo.

• **Campos de Entrada:** Pense em campos de texto, menus suspensos, caixas de seleção, botões de rádio e controles deslizantes.

• **Navegação:** Projete barras de navegação, barras laterais e barras de abas.

• **Cabeçalhos e Texto do Corpo:** Elementos de texto formatados corretamente para cabeçalhos e parágrafos.

Depois de ter os componentes básicos, é hora de lidar com os mais complexos:

• **Cards:** Comumente usados para exibir trechos de informações de maneira organizada (como os produtos em uma loja de comércio eletrônico).

• **Modais:** Popups que chamam a atenção do usuário (alertas, confirmações, etc.).

• **Tabelas:** Essenciais para exibir dados. Considere paginação, ordenação e filtros.

Os usuários interagirão com a sua interface, então considere vários estados:

• **Hover:** Como um elemento parece quando um usuário paira sobre ele?
• **Ativo:** O que acontece quando um elemento é clicado?
• **Desativado:** Certifique-se de que os usuários possam reconhecer componentes não clicáveis ou não interativos (por exemplo, um botão ou formulário de entrada desativado)

Seguindo esses passos, podemos começar a esboçar nosso primeiro aplicativo.

Estudo de caso: Lista de supermercado

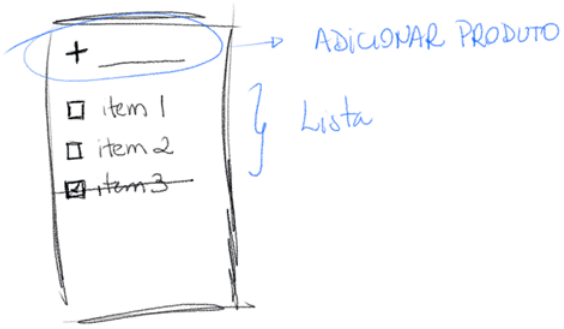
Vamos criar então nosso primeiro aplicativo: uma lista de supermercado. Pensando que nossa primeira versão é um MVP, vamos definir:

• **O problema central:** Queremos criar um aplicativo para armazenar nossa lista de compras. Ele consiste em uma lista de itens que podem ser adicionados e removidos. Você pode buscar outros aplicativos que fazem isso para poder te inspirar.

• **Histórias do usuário:** O usuário deve ser capaz de criar um item (dando um nome a ele) e adicioná-lo à lista. Quando clicar no item, o mesmo deve ser riscado simbolizando que foi retirado da lista. Se clicar em um item riscado, ele volta para a lista.

• **Priorize os recursos:** Aqui vamos pensar somente na funcionalidade de inserir e retirar o nome do item.

• **Esboce ou crie wireframes:**



Então o escopo do nosso projeto será uma página com:

- Um campo de formulário para preencher com o nome do produto e um botão adicionar;

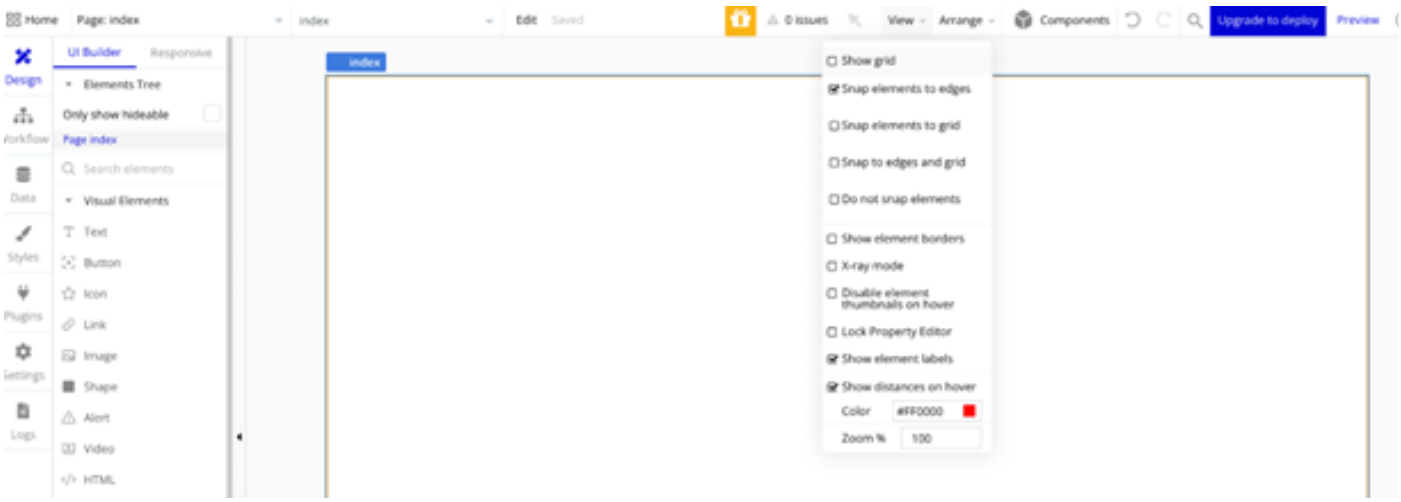
- Uma lista com todos os produtos já adicionados;

- Uma função para quando clicar no item, vai marcar como comprado;

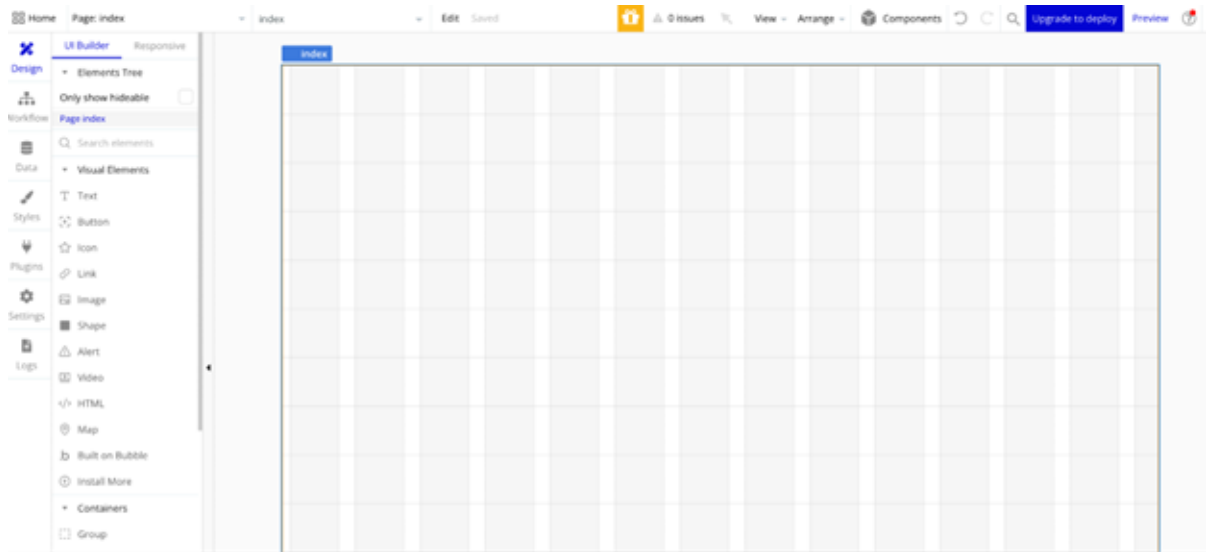
- Uma função para quando clicar no item já comprado, desmarcá-lo.

Configurando o layout

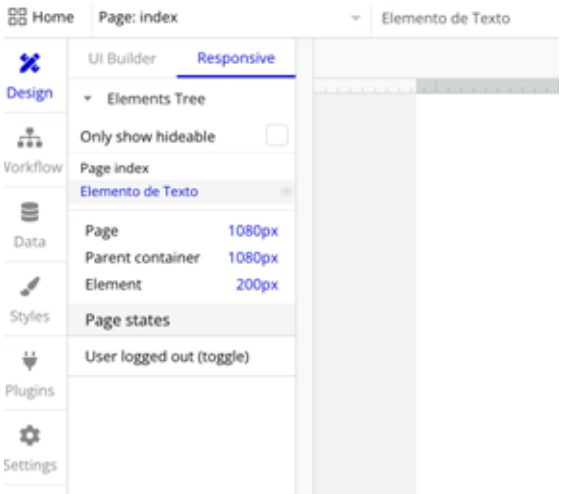
Vamos começar ativando a grade do nosso canva para que facilite a visualização dos elementos. No menu superior clique em **View** e selecione **Show Grid**.

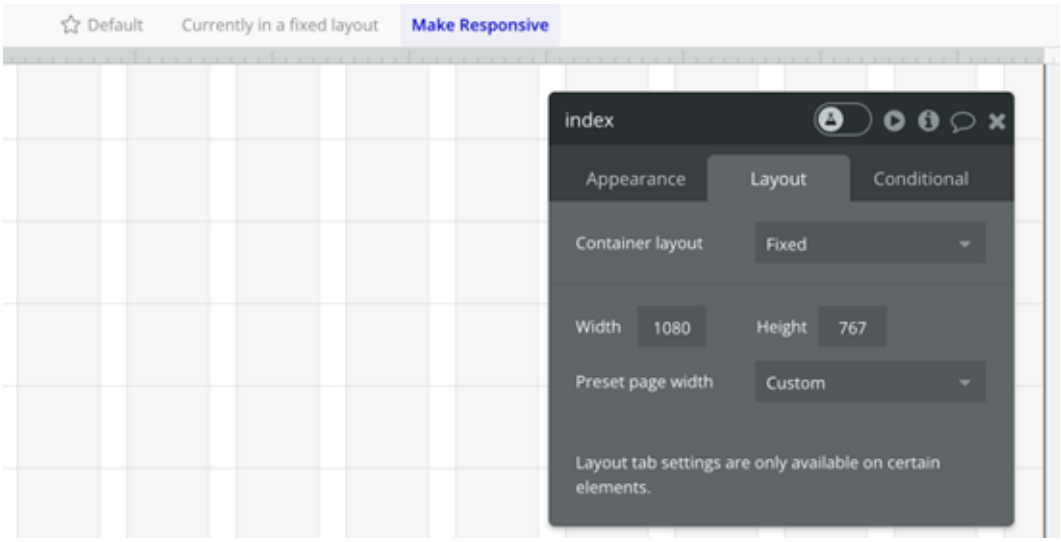


Assim, teremos essa visualização em grade:



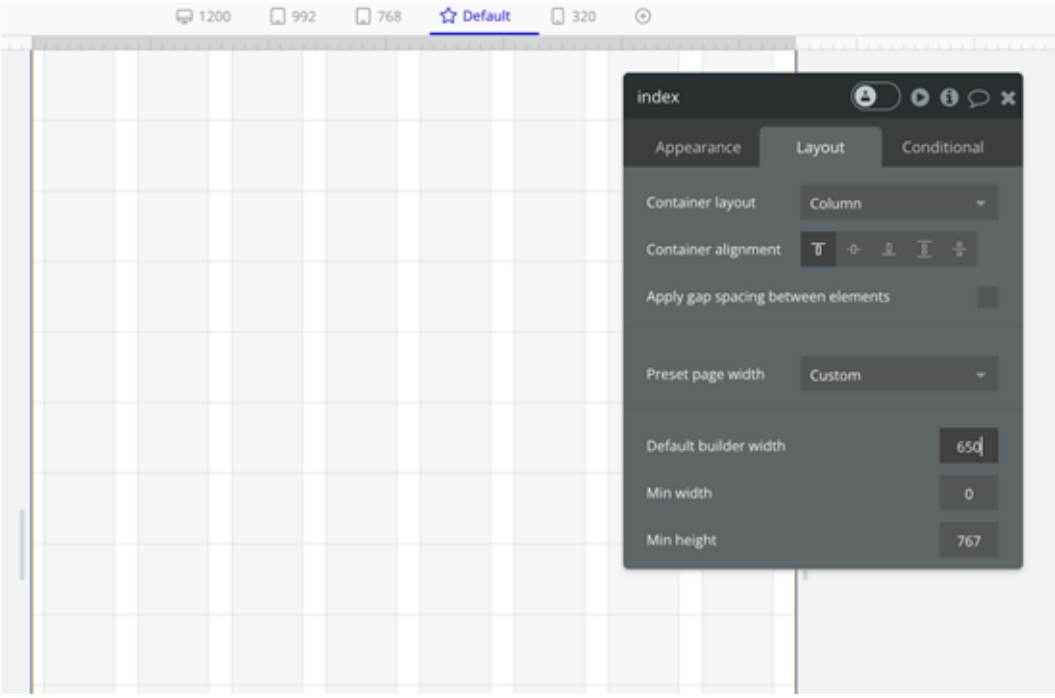
Agora vamos na parte de Responsividade **(Responsive)** e vamos preparar nosso layout.





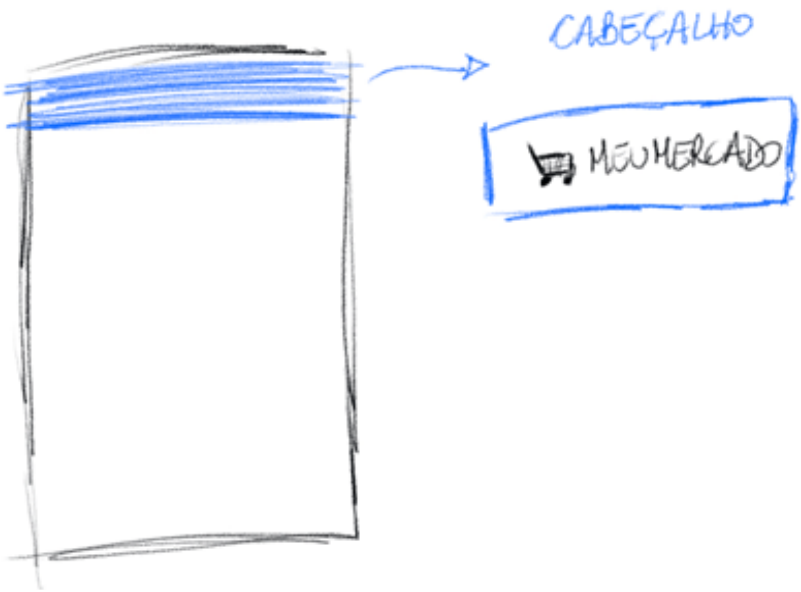
Ao clicar em **Make Responsive**, vamos trabalhar no menu **Layout**. Por padrão **(default)**, o **Container** layout estará como fixo **(Fixed)**. Vamos modificá-lo para **Column** (coluna). Isso significa que ao posicionar nossos elementos eles ficarão um em cima do outro.

Vamos alterar também a largura para 650. Isso vai dar uma margem legal para trabalharmos.

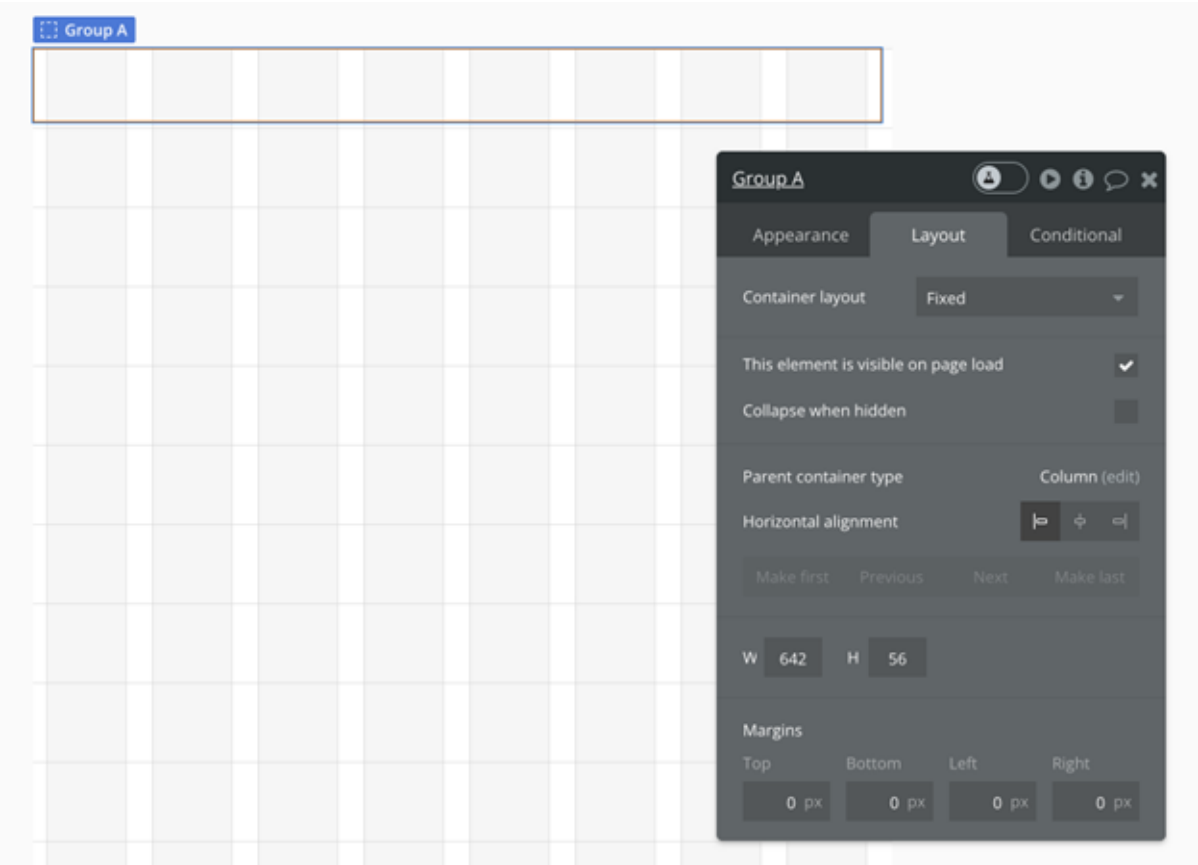


Cabeçalho

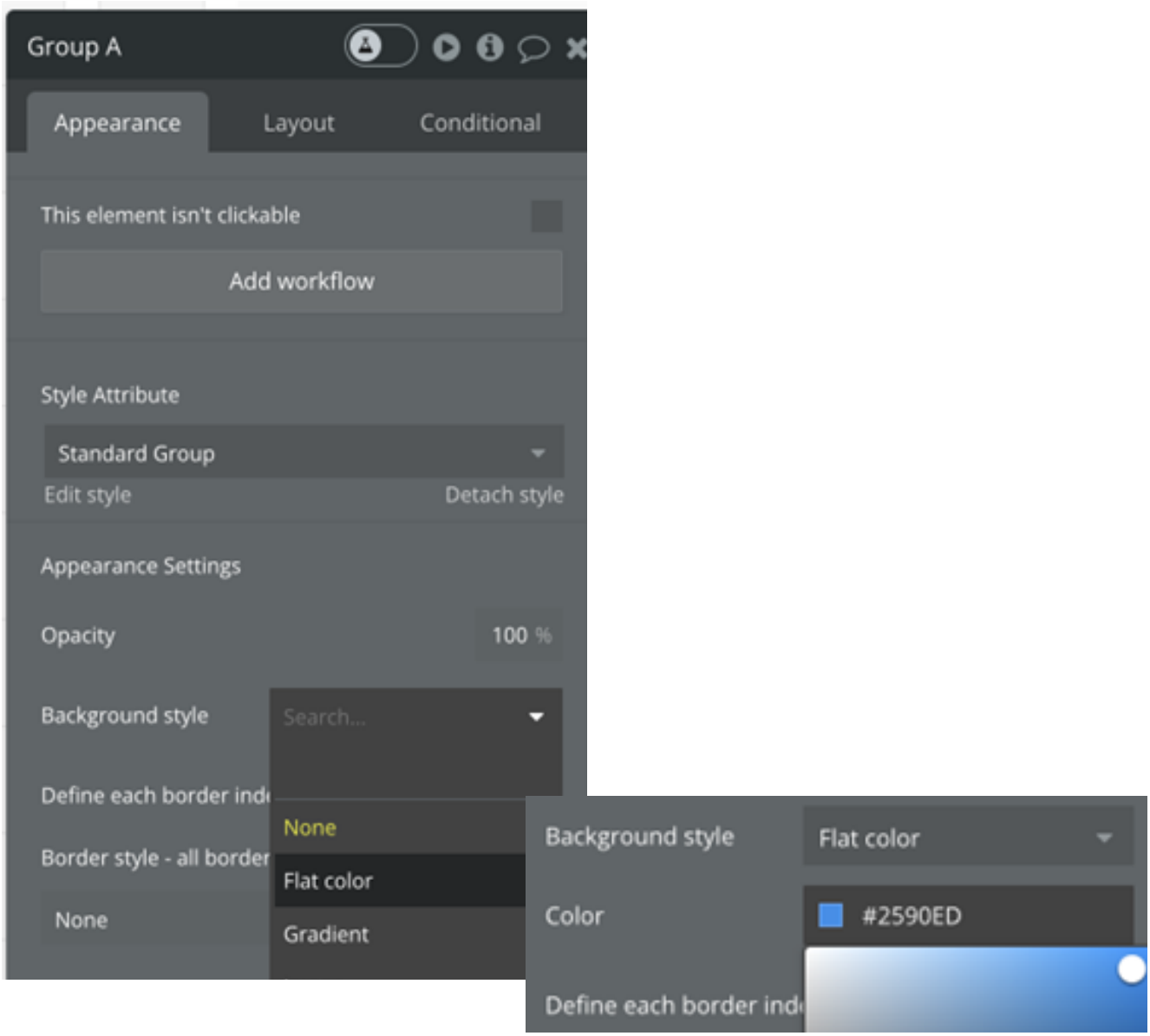
Vamos começar a desenhar nosso aplicativo pelo cabeçalho, isto é, a parte superior do nosso aplicativo. Pensando no nosso rascunho, ele vai ser uma barra superior com o nome do aplicativo e um ícone.



Para criar um cabeçalho no Bubble, vamos arrastar o elemento **Group**.



Agora vamos ajustar esse nosso grupo. Ele deve ter uma cor e ser composto por dois outros elementos: O nome do aplicativo e um ícone. Para ajustar a cor, na janela de propriedades (cinza escuro) vá em **Appearance** (Aparência) > **Background style > Flat Color**. Depois explore mais opções de fundo! Aqui eu escolhi a cor **#2590ED.**

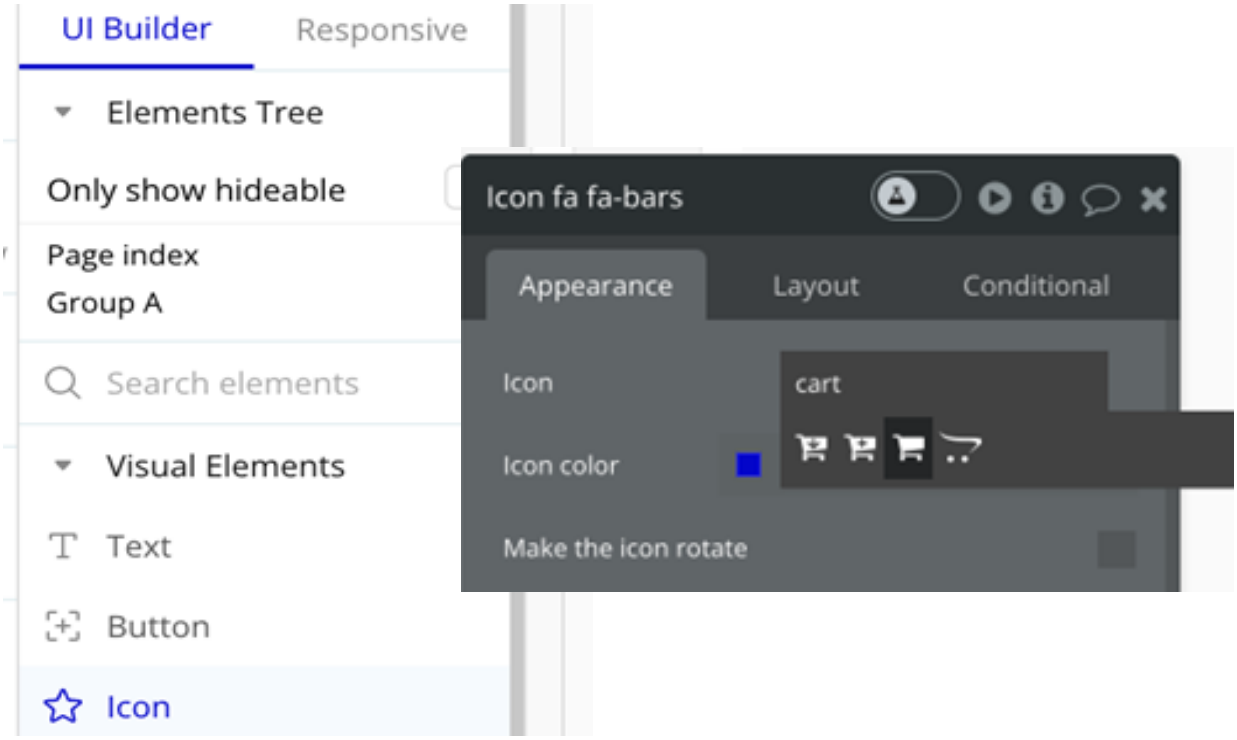


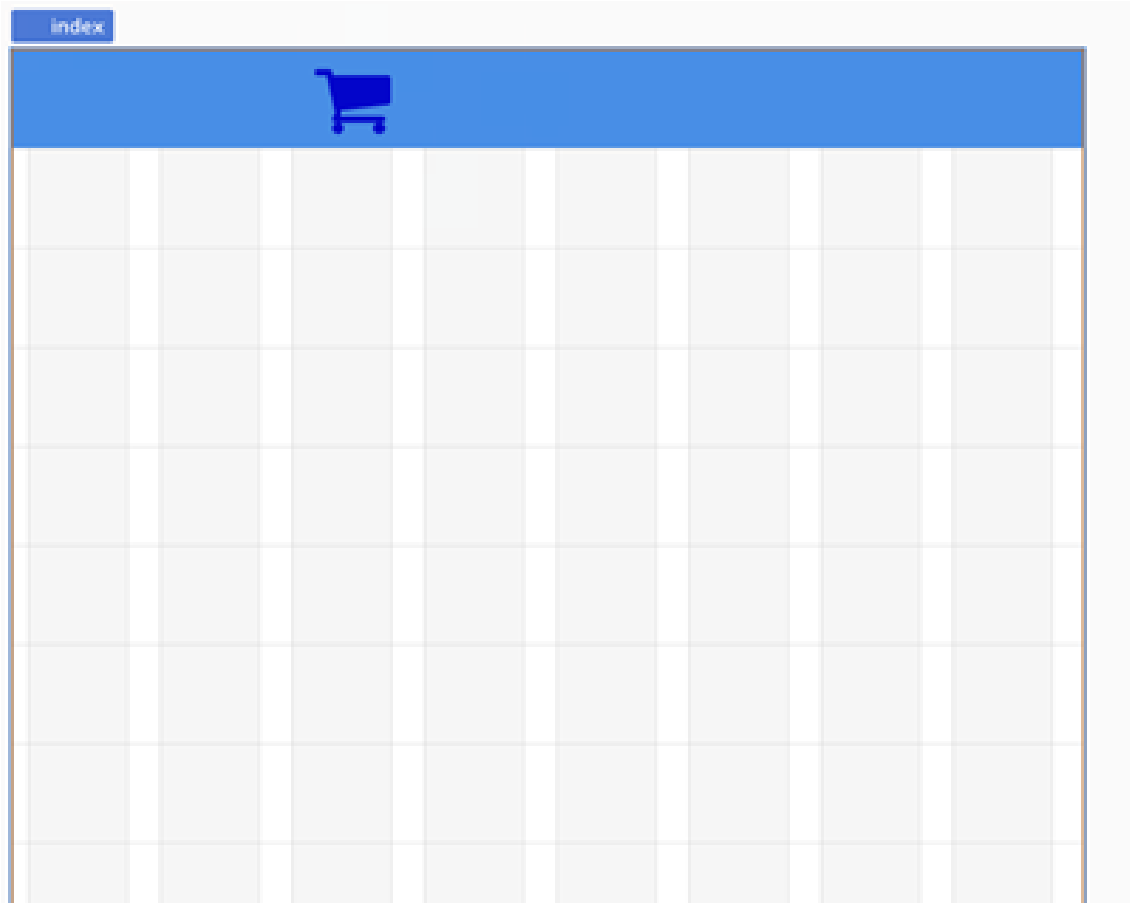
Para o tamanho, coloque no **Layout** os valores de **W** (witdh - largura) e **H** (height - altura) em 650 e 60. Você pode ir modificando isso para deixar o design com a sua cara.

Quer ver como está ficando? Clique em **Preview**.

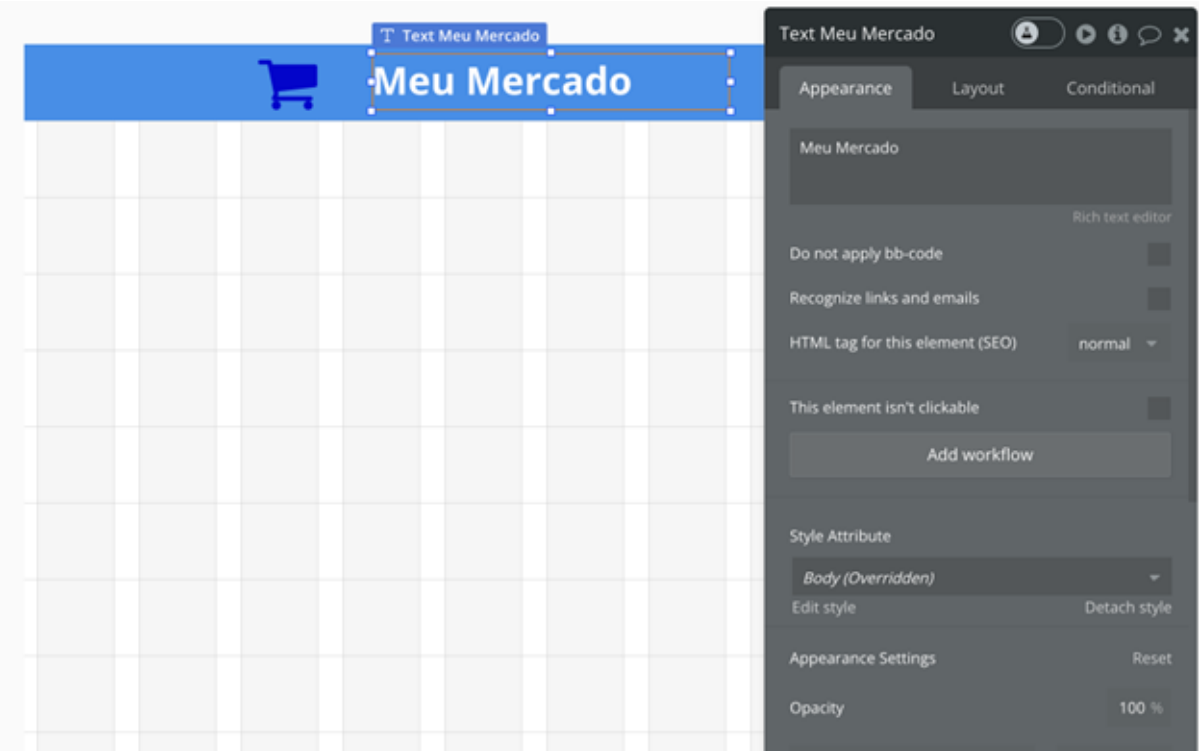
Vamos agora acrescentar o elemento ícone. O ícone pode tanto ser o elemento **Icon**, quanto o elemento **Image**. Aqui vamos usar um ícone da própria biblioteca, mas tenha em mente que você pode desenhar sua própria logo.

Clique em **Icon** e depois no seu **Group** para escolher a posição. Eu selecionei o ícone **Cart**, para simbolizar nossa lista de mercado.





Agora só falta colocar um texto para colocarmos o nome do aplicativo. Clique em **Text** e nas propriedades altere seu valor para Meu Mercado. Altere a fonte, o tamanho e a cor para combinar melhor com nossa interface. Aqui eu coloquei: 30px, Open Sans e Branco.



Outra opção para montarmos nosso cabeçalho é usar repositórios de imagem para nossa logo. Fica uma dica aqui: https://www.flaticon.com/br/. Basta baixar a imagem que você achou mais legal e colocar como elemento **Image** no nosso grupo.

A imagem escolhida pode ser encontrada [aqui.](#)

Pronto, nós temos um cabeçalho!

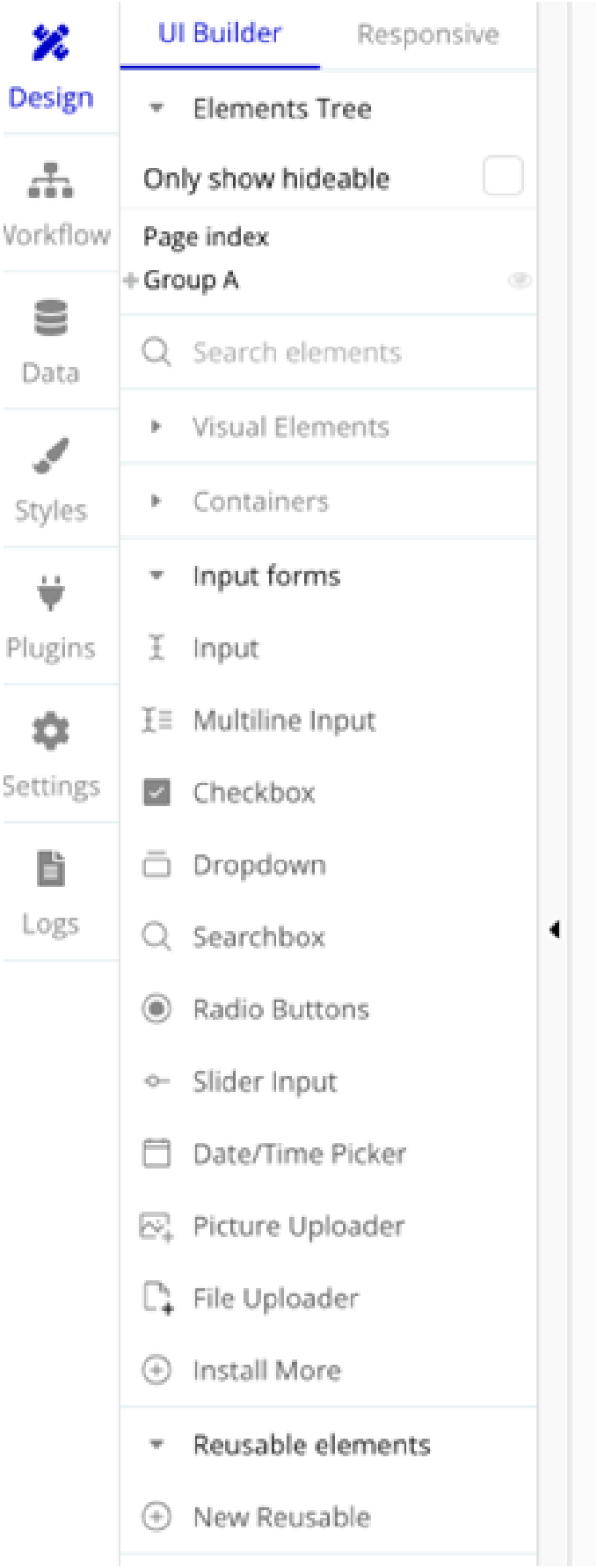
Cadastrando um item

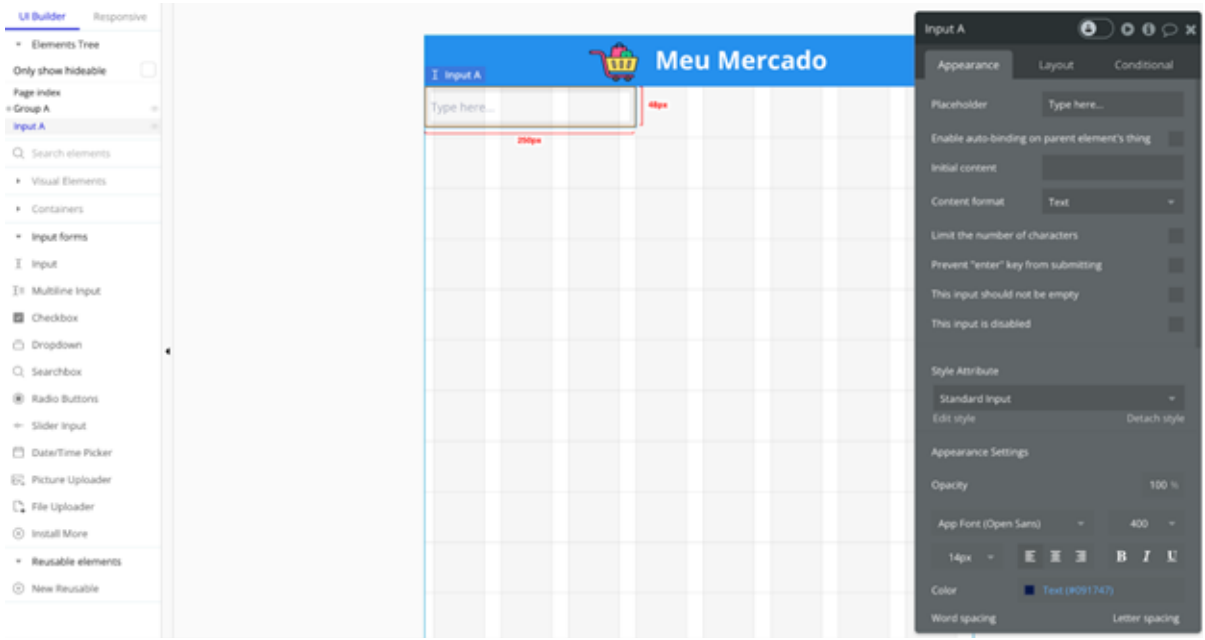
No nosso escopo de produto, indicamos que precisamos de um campo de formulário para preencher com o nome do produto e um botão adicionar. Os elementos de um formulário **(Form)** ficam na aba **Input Forms.**

Nela estão todos os elementos para criação de um formulário.

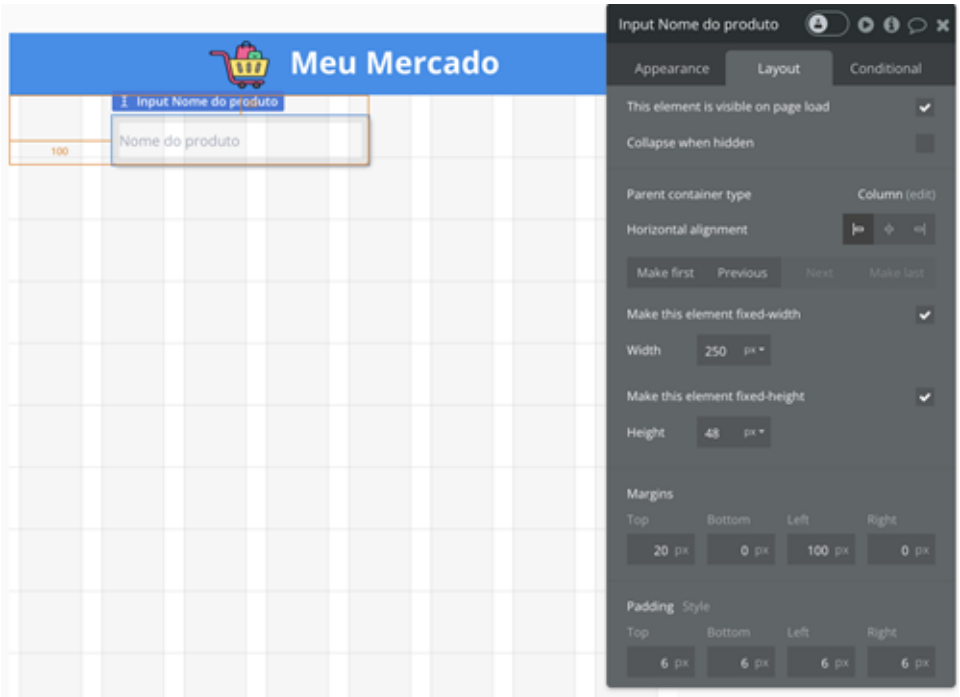
No nosso rascunho, colocamos um campo para escrita do nome do item e um botão para adicionar. Então aqui vamos utilizar o elemento de formulário **Input**, que representa uma entrada simples. Depois vamos colocar um elemento visual **Button** para ser o botão de adicionar.

Para o **Input**, vamos precisar mexer nas propriedades de aparência e também de posicionamento na tela. Ao colocá-lo no nosso canva, ele fica alinhado à esquerda, logo abaixo de nosso cabeçalho.

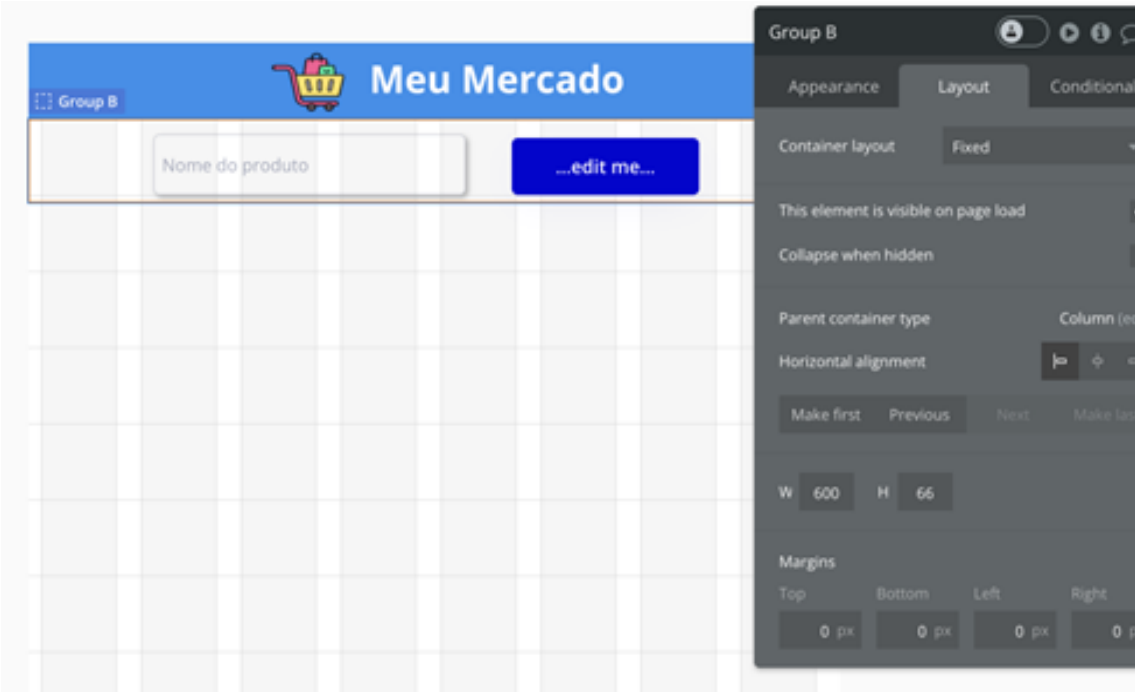




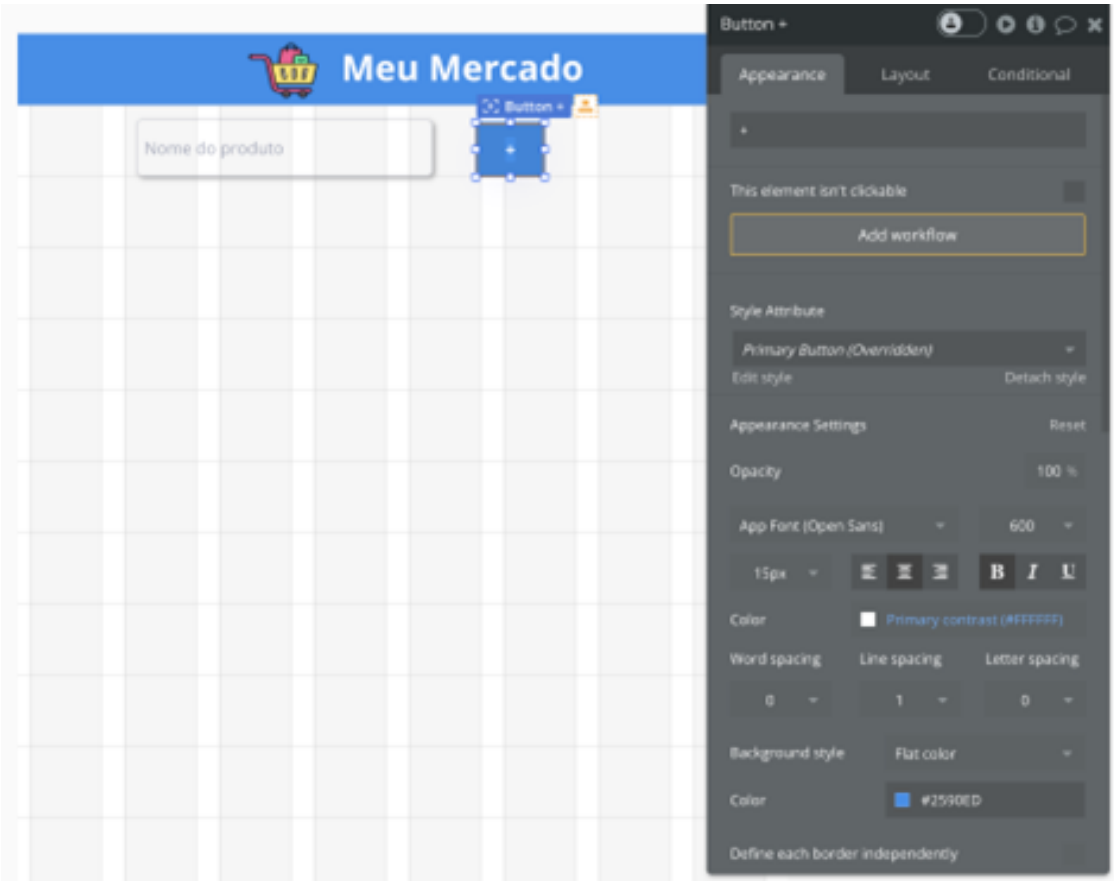
Aqui vamos alterar o **placeholder** para Nome do produto e vamos adicionar uma sombra **(Shadow Outset)** no elemento. Vamos também alterar o layout para colocar um espaçamento entre a margem e o cabeçalho. Explore sempre a janela de propriedades para ajustar o seu design.



Agora vamos colocar nosso botão ao lado do nosso texto. Para isso vamos precisar repensar o layout do nosso aplicativo. Nós colocamos ele todo como Coluna, isso significa que todos os elementos dentro dele ficarão um em cima do outro. Vamos então criar mais um grupo para conter os elementos **Input** e **Button** para trabalharmos com eles em linha. Coloque um **Group** abaixo do nosso cabeçalho e arraste nosso **Input** até ele. Depois, clique em **Button** e coloque nesse nosso novo grupo.



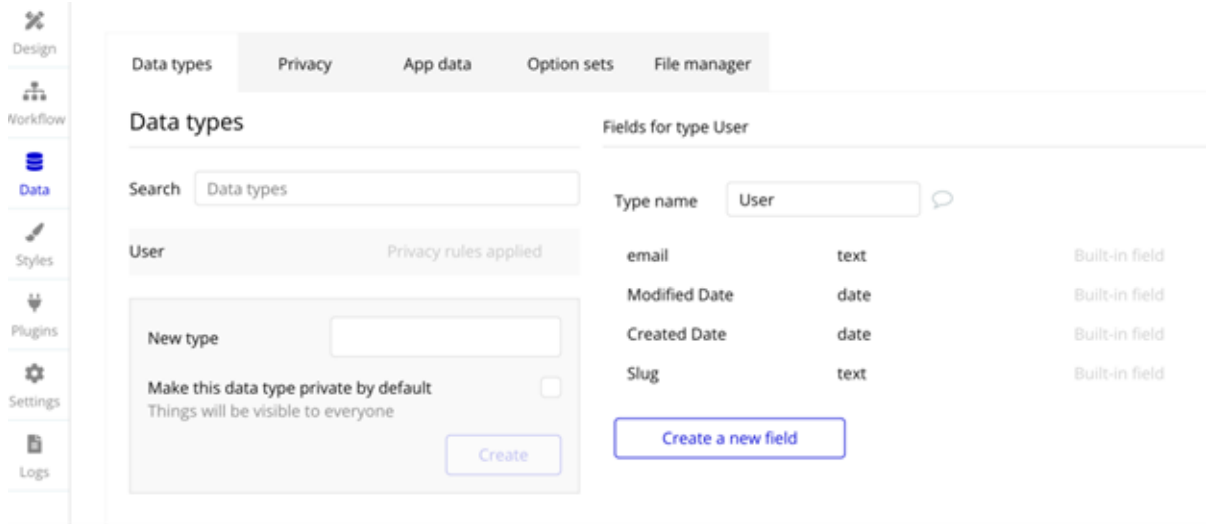
Agora vamos mexer nas propriedades do nosso botão para ele ficar mais legal. Eu coloquei o texto como sendo + e coloquei como cor a mesma do cabeçalho.



Agora podemos começar a fazer as funcionalidades do nosso aplicativo!

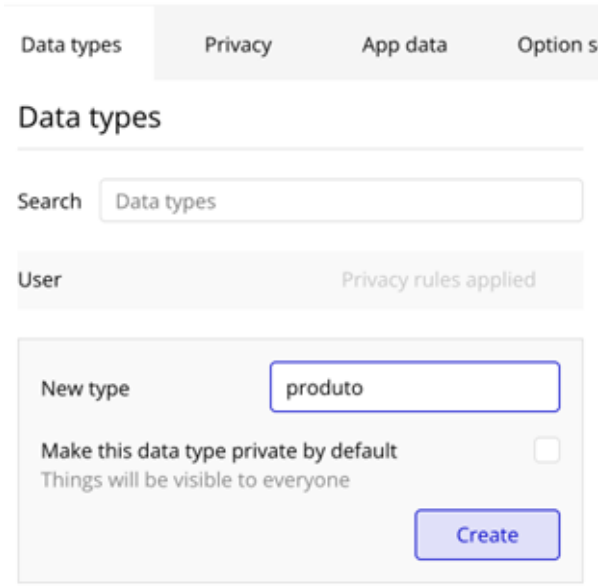
Criando e preparando um banco de dados

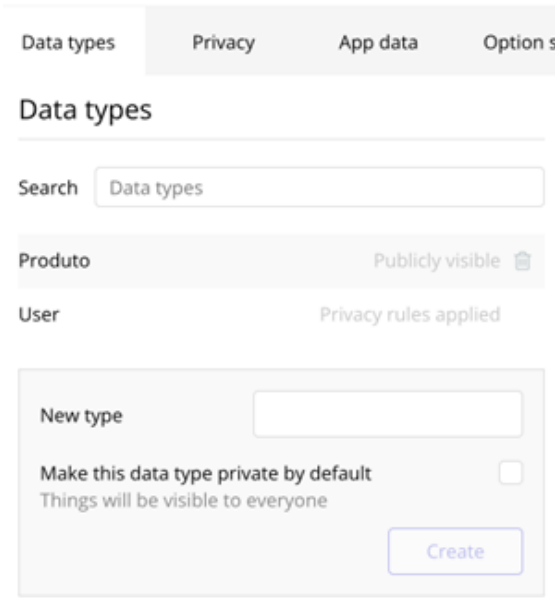
Para que possamos manter nossa lista de compras, precisamos montar um banco de dados simples. Para isso, precisamos estruturar a informação que desejamos armazenar. Nós podemos fazer isso na aba **Data** do Bubble.



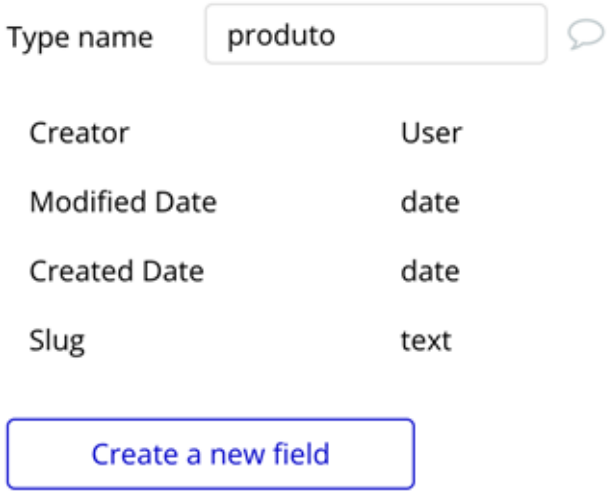
Aqui não vamos entrar nos detalhes de um banco de dados, isso veremos no próximo módulo. Mas já podemos estruturar um banco básico para nossas informações: nome do produto. Nós vamos focar nas 3 primeiras abas: D**ata types, Privacy e App data.**

Data types se refere aos tipos de dados ou tabela. Por padrão, o Bubble já começa com a tabela **User** (usuário) com campos configurados. Vamos deixar essa tabela quietinha por enquanto e criar nossa própria tabela: **produto**.



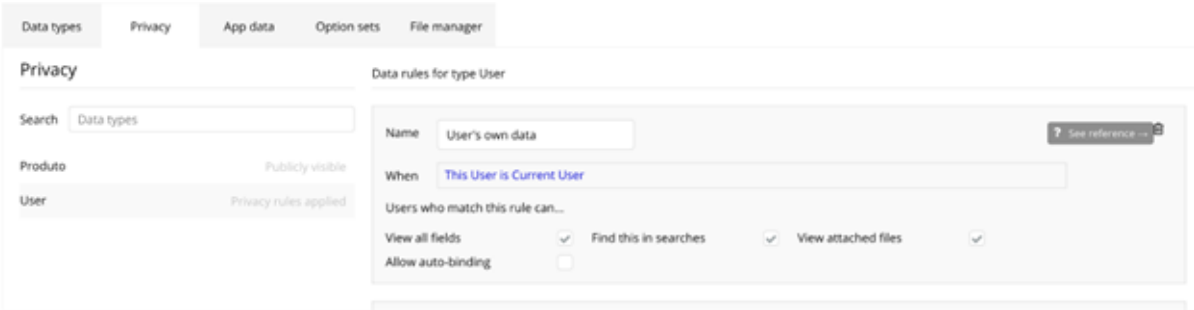


Ao criar a tabela **Produto**, o Bubble cria alguns campos automaticamente:



Vamos deixá-los por enquanto e criar um novo campo **Create a new field**: nome do produto. O tipo de dados que escolheremos é texto e não é uma lista, pois estamos tratando dos itens individualmente.

Agora vamos apagar uma configuração padrão que o Bubble coloca na aba **Privacy**, que trata da privacidade dos dados. Clique em **User** e clique no ícone lixeira. Assim apagamos essa regra de privacidade que não é interessante para nós agora.

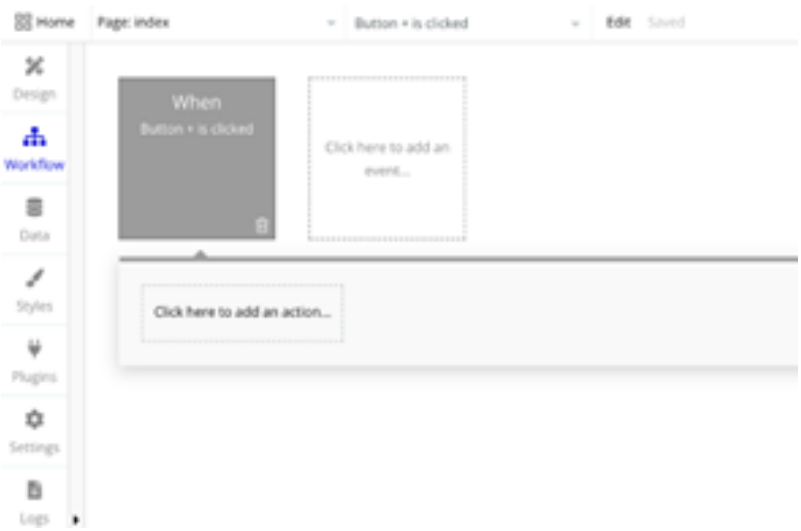


Na aba **App data** temos acesso em tempo real ao nosso banco de dados. Por enquanto, tudo estará em branco pois não cadastramos nenhum produto e nenhum usuário.

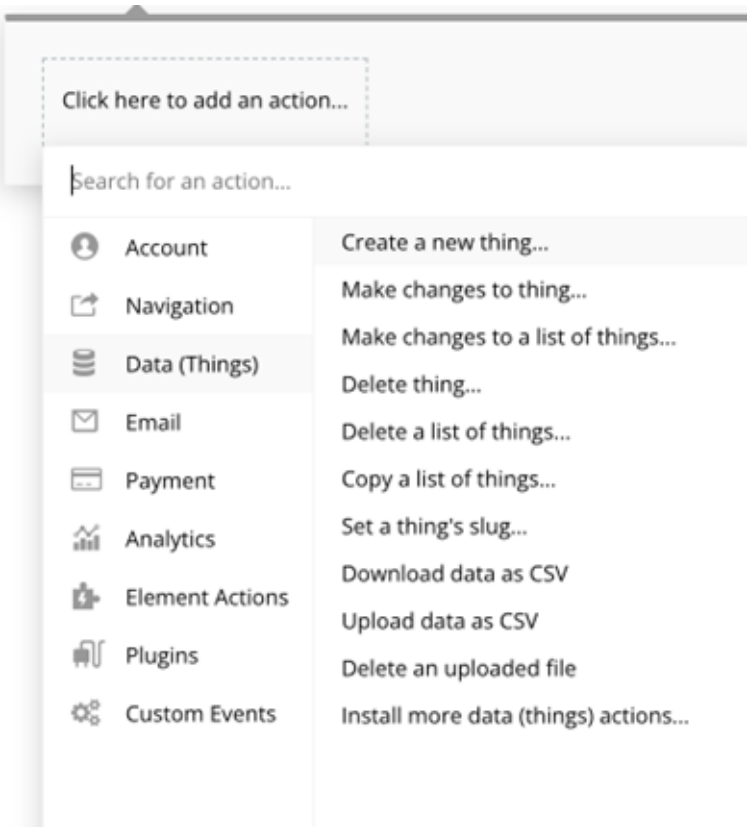
Agora é hora de fazer nosso botão funcionar e adicionar os produtos em nosso banco de dados! Nosso botão precisa de um fluxo de trabalho, um **Workflow**.

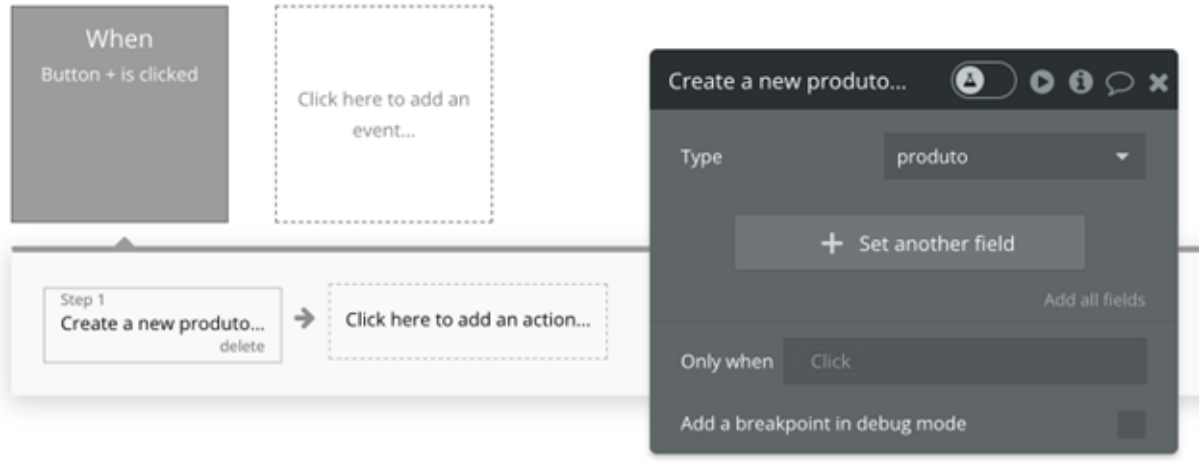
Clique em **Add workflow**. Lembre-se de sempre selecionar o elemento que você quer trabalhar para abrir a janela de propriedades correta.

Ao clicar em **Add workflow**, iremos para a aba **Workflow** na qual poderemos montar o evento associado a esse botão. Por padrão, o Bubble já cria o que chamamos de gatilho, que é o que deve ser feito para acionar o evento. No nosso caso, quando o botão for clicado, devemos adicionar o nome do produto no nosso banco **(action)**. Então vamos montar esse fluxo.

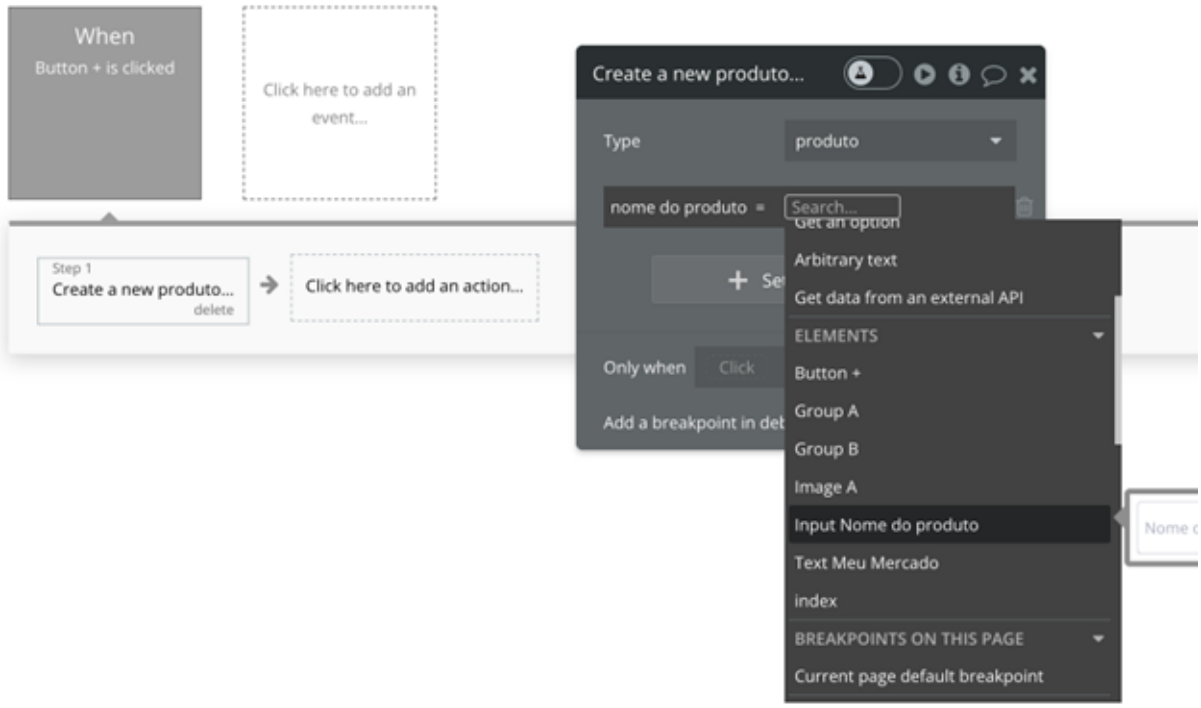


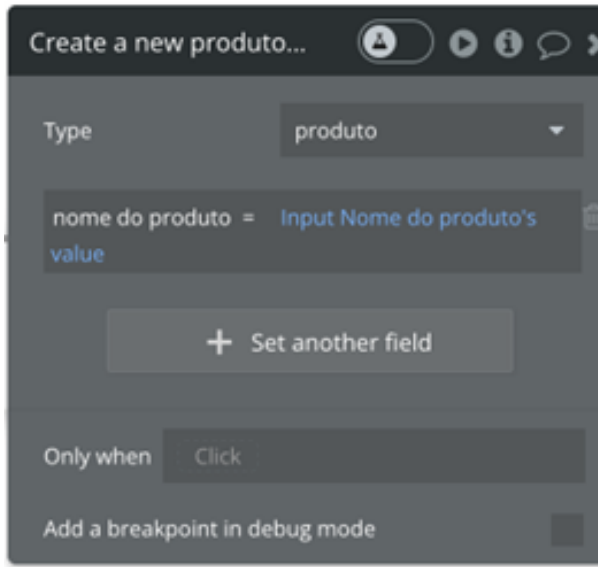
Clique em **Click here to add an action**, para criarmos a ação desse botão! Vamos criar uma ação do tipo **Data** para criar uma nova entrada no banco de dados **Create a new thing**. E qual coisa criaremos? O produto!





E qual é o dado da tabela que vamos inserir? Temos que inserir o nome do produto baseado no que o usuário digitou no nosso **Input**. Então vamos clicar em **Set another field**, selecionar "nome do produto" e associá-lo ao nosso Input. Aqui é bem importante que nossos elementos tenham um nome intuitivo, assim conseguiremos encontrá-los de maneira mais simples.

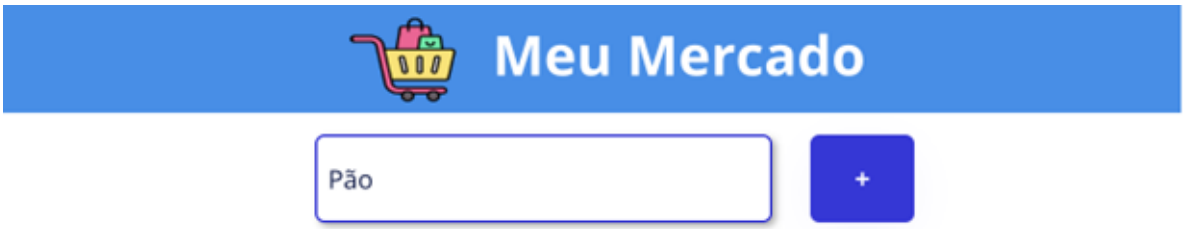




Agora dê um **Preview** e teste. Vá em **App data** para ver se foi adicionado corretamente no nosso banco.

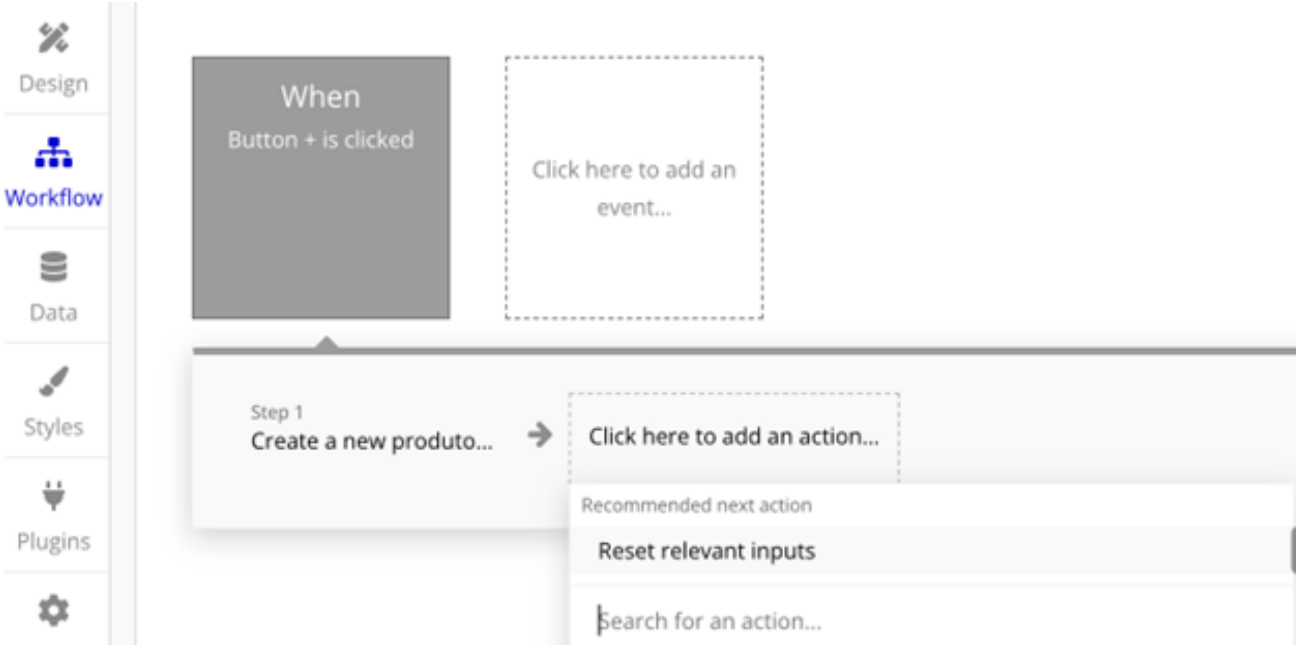
Quer saber exatamente o que está acontecendo no seu aplicativo? Você pode usar o seu Debugger na tela de Preview.

Antes de fecharmos essa parte do banco de dados, vamos analisar uma pequena melhoria. Sempre que adicionamos um item, o nome do produto continua no nosso Input. É bem mais elegante ele ser apagado, afinal o produto aparecerá na lista que vamos criar.



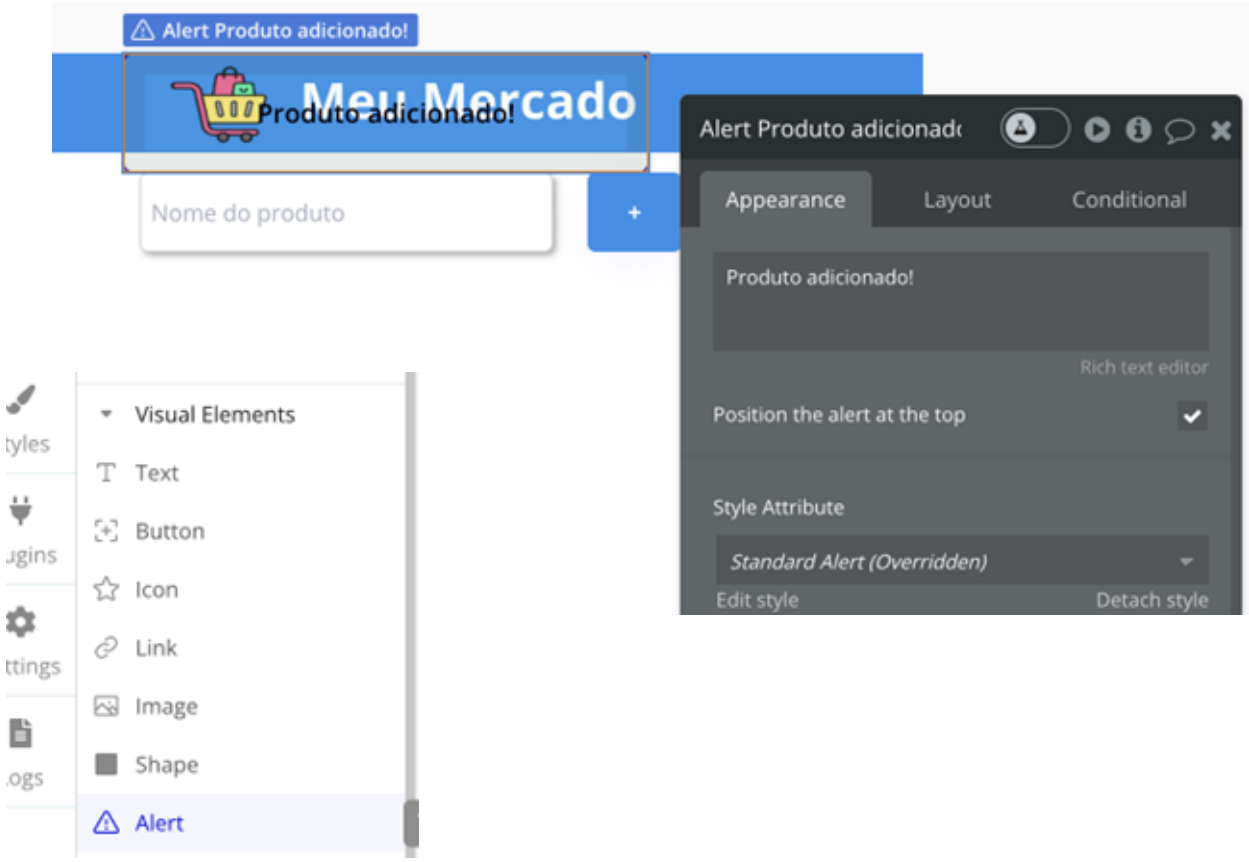
Também é interessante criar um alerta avisando que o produto foi adicionado. Isso tudo é importante para a melhoria da experiência do usuário.

Vamos então novamente no **workflow** do nosso botão para adicionar uma ação. Além de adicionar no banco, vamos apagar o texto da entrada. Uma coisa legal é que o Bubble já te recomenda exatamente a ação que queremos: Apagar o campo - **Reset relevant inputs.**

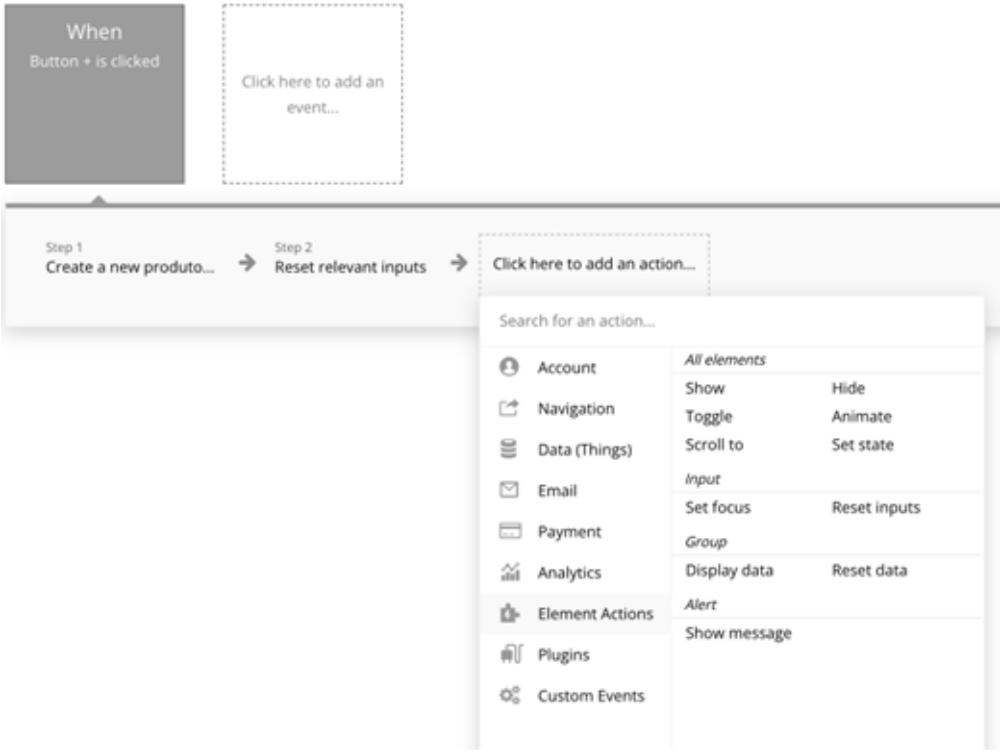


Essa mesma ação é encontrada em **Element Actions.**

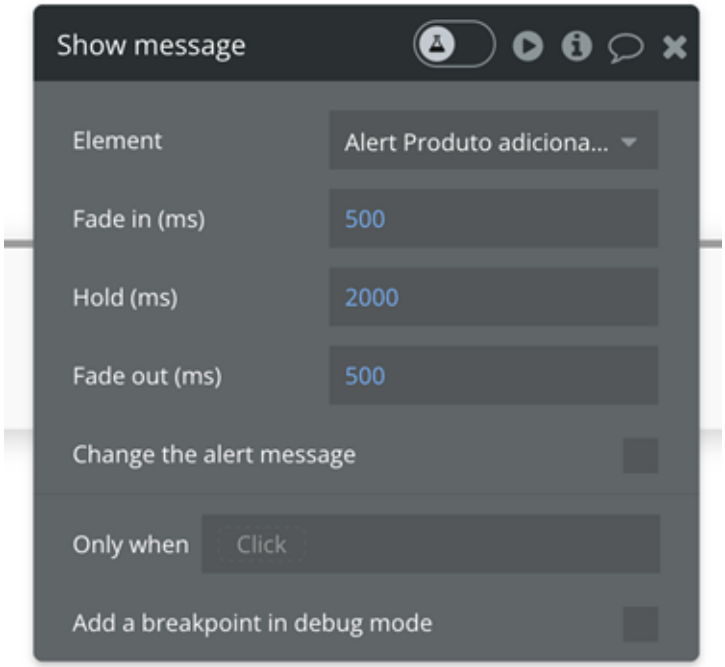
Agora vamos criar o alerta na tela. Para isso, vamos inserir o elemento **Alert** e posicioná-lo na nossa tela. Marque em sua propriedade **Position alert at the top**.



Agora voltando ao nosso workflow, nossa próxima ação é mostrar o alerta: **Element actions > Alert > Show Message.**



Depois podemos configurar quanto tempo ele fica na tela. Por padrão são 2000 milissegundos, isto é, 2 segundos.

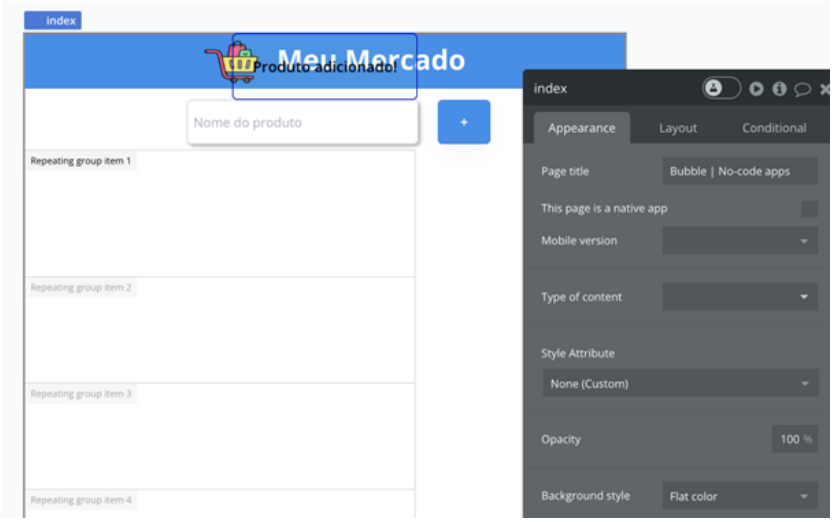


Agora sim, podemos fazer a funcionalidade de mostrar a nossa lista!

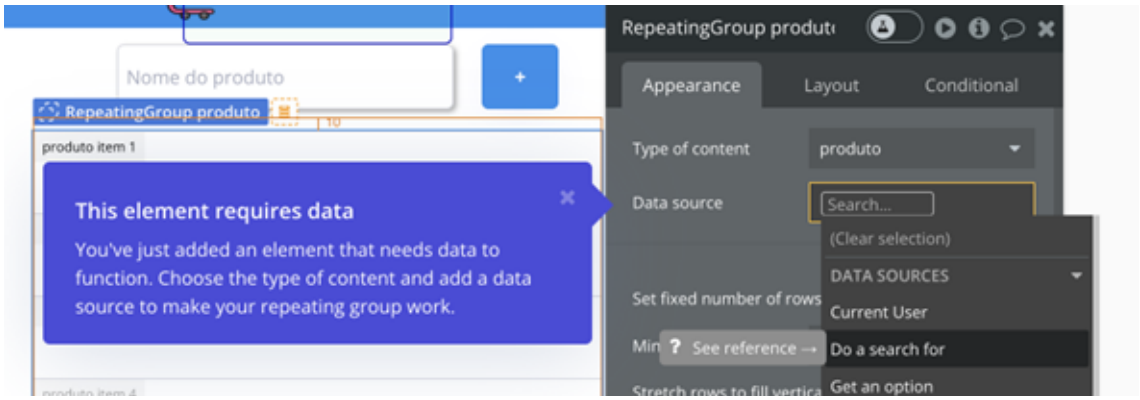
Mostrando a lista na tela

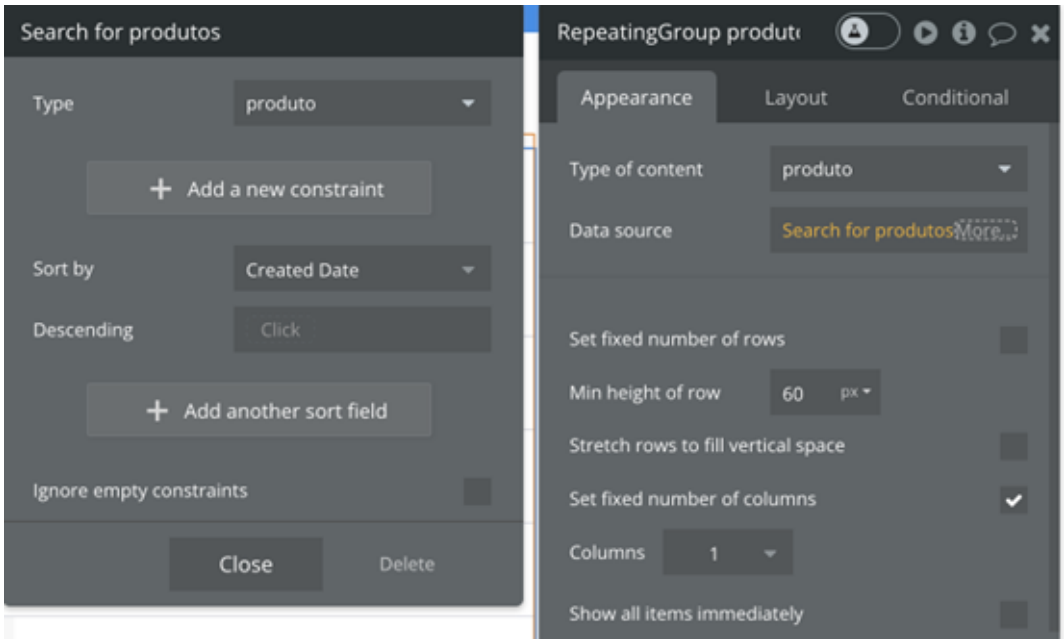
Voltando ao escopo do nosso projeto, precisamos mostrar na tela os itens que adicionamos. No nosso rascunho colocamos uma caixinha ao lado do nome do produto. Essa caixinha é o que chamamos de **checkbox**. Teremos que mostrar vários elementos deste tipo, afinal, estamos mostrando uma lista.

No Bubble, sempre que precisamos inserir listas, nós devemos colocar o elemento **Repeating Group,** o grupo repetidor. Então pegue esse elemento e arraste para a tela. A ideia deste elemento é que tudo o que você inserir na primeira célula, irá se repetir nas células seguintes.



A primeira informação que precisamos colocar nesse elemento é qual o tipo de conteúdo, **type of content** e qual a fonte do dado, **data source**. Nós queremos puxar as informações de produtos, então utilizaremos o **Do a search for**.

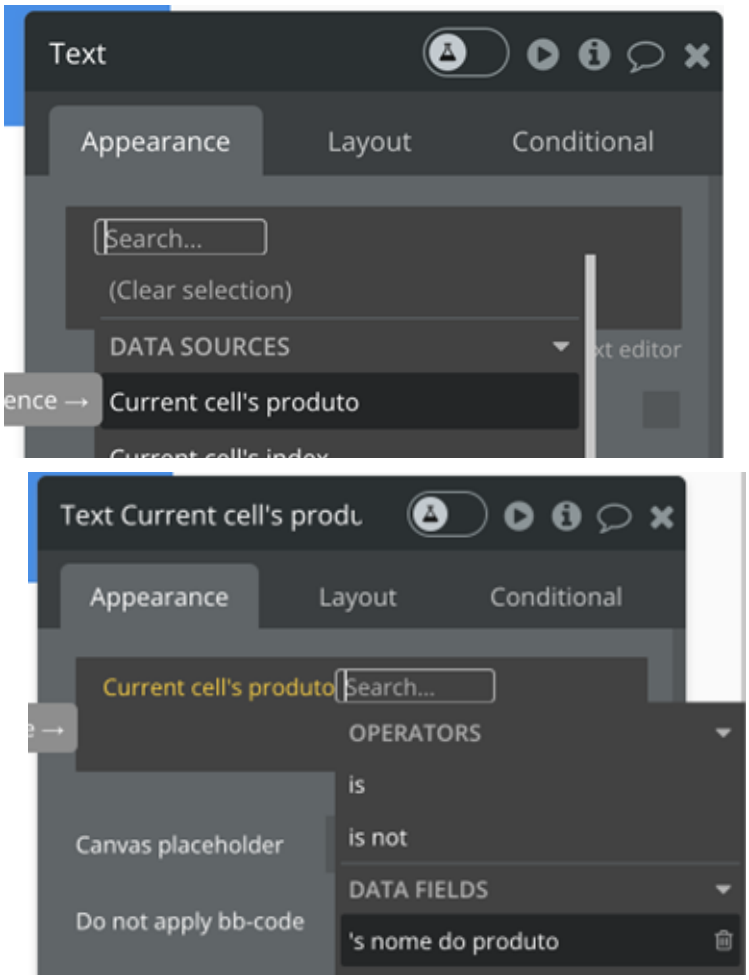




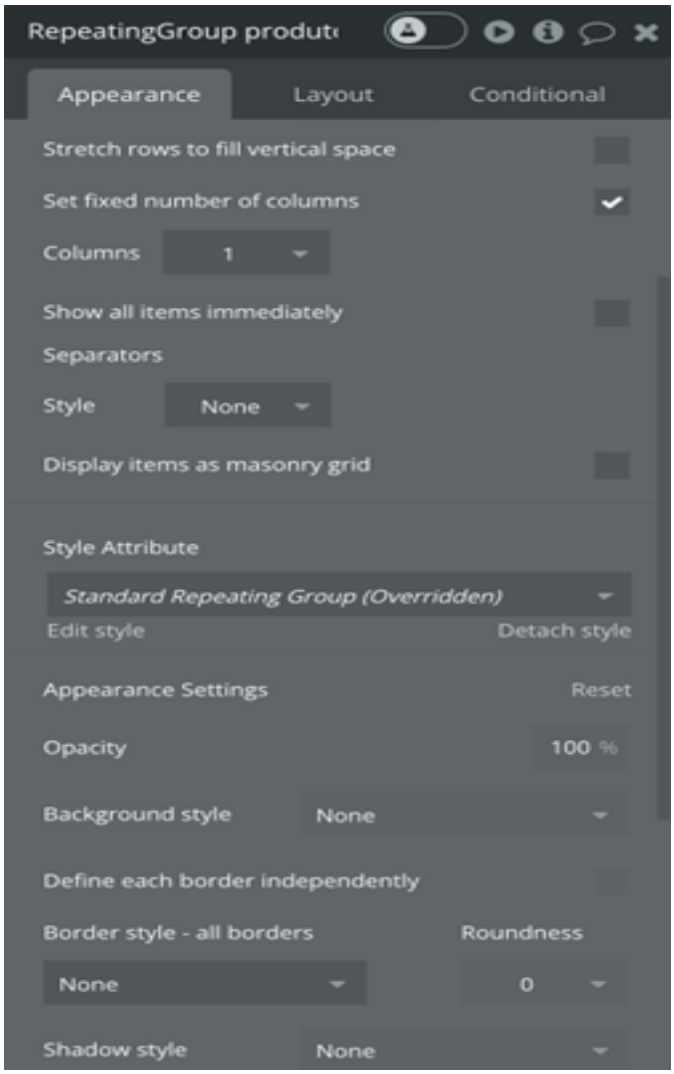
Assim já vamos conectar nosso banco à nossa tela. Agora precisamos efetivamente mostrar o nome do produto. Então vamos arrastar um **Text** para a primeira célula do nosso **Repeating Group**. Se deixarmos somente assim, no nosso **Preview** teremos apenas:

Isso acontece porque, apesar de eu ter inserido dois itens no banco de dados, o valor do campo texto está estático. Aqui nós precisamos puxar um dado dinâmico. Então na janela de propriedades desse texto, clique nela e selecione **Insert dynamic data.**

Vamos pegar então **Current cell's produto, nome do produto.**



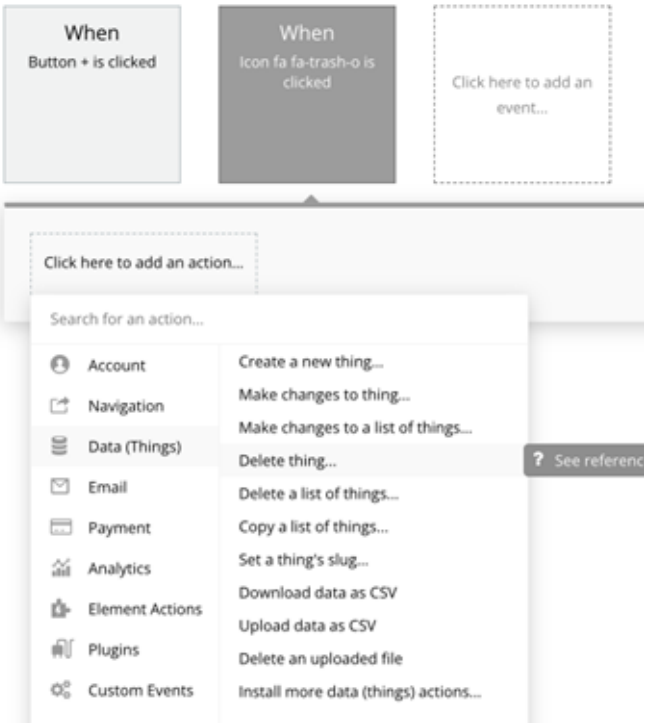
Prontinho! Agora podemos trabalhar um pouco no design do nosso grupo repetidor para ele aparecer um pouco mais legal. Como ainda vamos colocar mais coisas na nossa célula, tire a borda e o separador para deixar a interface mais limpa. Basta modificar as propriedades **Separator style none** e **Border style none**.



Pronto! Já conseguimos ver as informações na nossa lista. A próxima funcionalidade é apagar um item. Vamos fazer isso antes da funcionalidade de riscar o item, como fizemos no nosso rascunho.

Vamos colocar um ícone **(Icon)** de lixeira na célula do grupo repetidor.

Assim como fizemos com o botão adicionar, precisamos configurar a ação na nossa lixeirinha. Para fazer isso, basta adicionarmos um **Workflow** no nosso ícone de lixeira. A ação que vamos criar é **Delete things** para deletar o produto da célula.

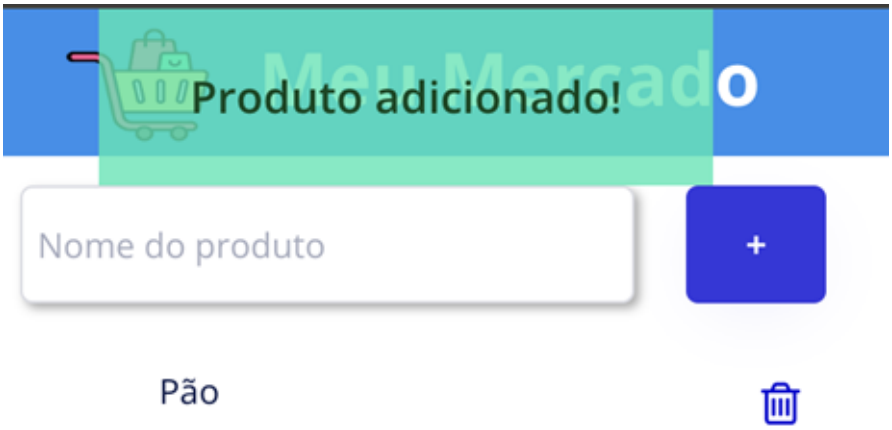


Vamos fazer agora um alerta para dizer que um item foi apagado. Lembrando que é sempre importante dar feedbacks para o usuário para melhorar a sua experiência.

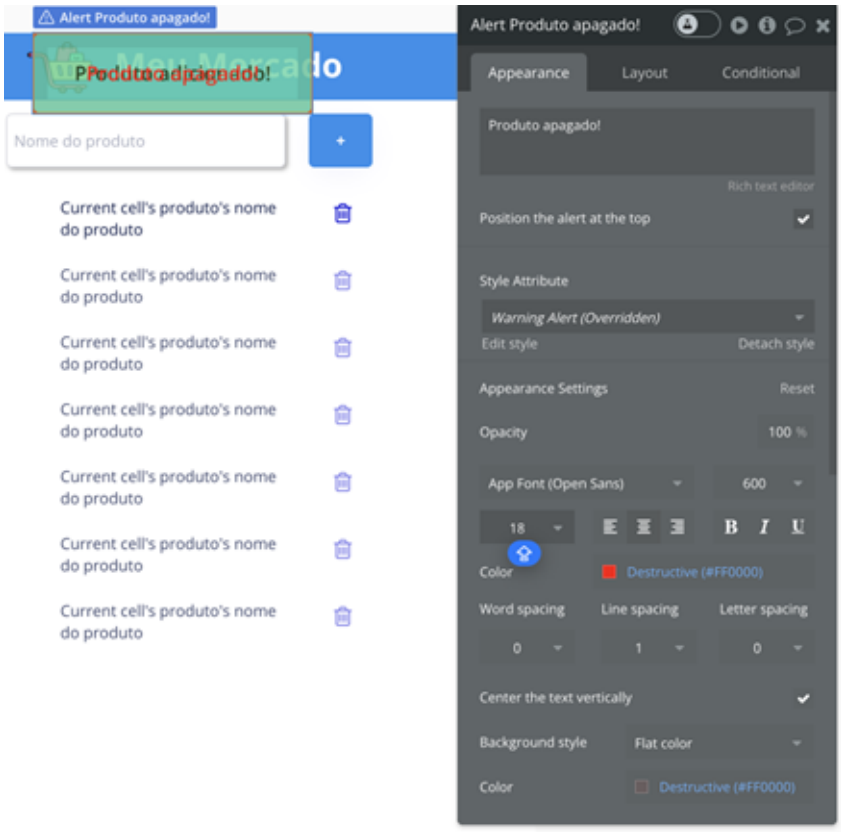
Vamos aproveitar esse momento para ajustar o design dos nossos alertas. Vamos colocar verde para quando o item for adicionado e vermelho para quando for apagado.

Vamos voltar no nosso alerta de produto adicionado e trocar algumas cores:

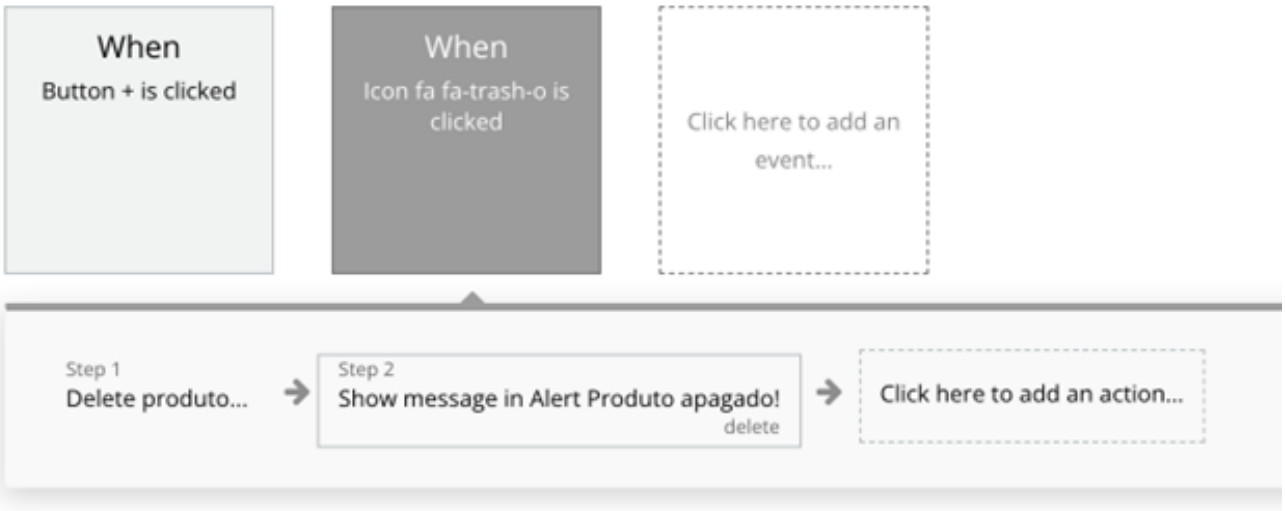
- Cor da fonte: #014721
- Fundo: Estilo **Flat Color** e cor #06E7B3
- Estilo da borda: **none**



Pronto! Agora está mais bonitinho! Vamos criar um novo alerta, seguindo os passos desse primeiro. Vamos aproveitar alguns padrões do Bubble e escolher o alerta com estilo **Warning.**

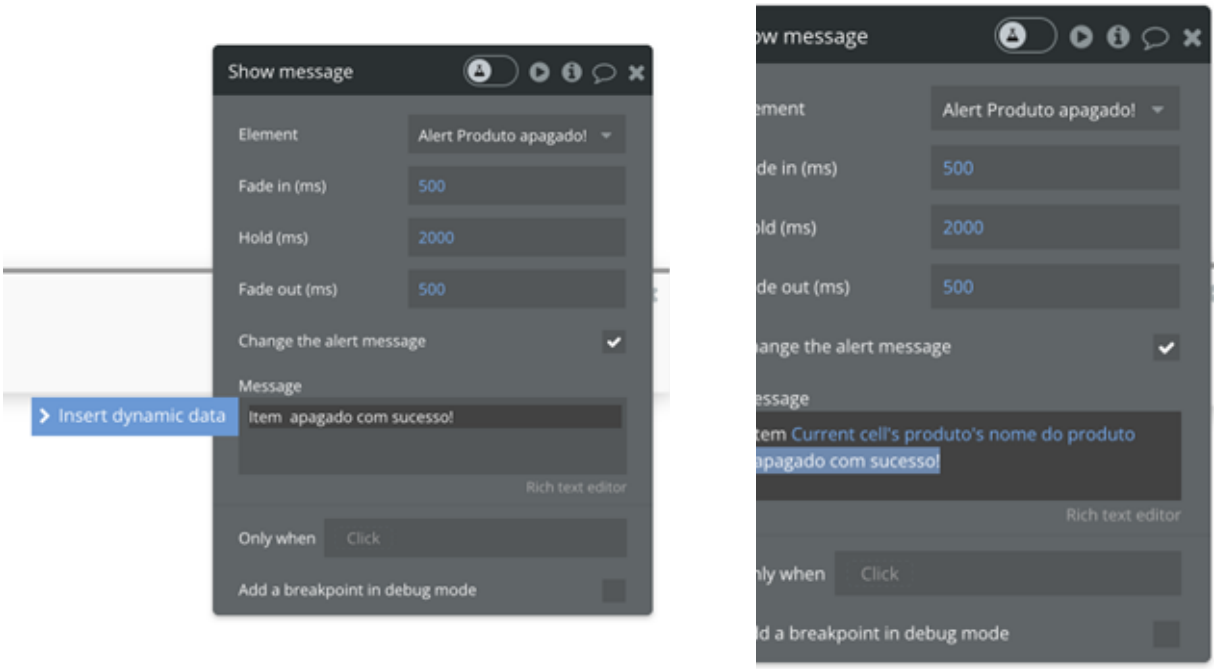


Agora basta ir no **Workflow** da nossa lixeira e adicionar a ação de mostrar esse novo alerta.

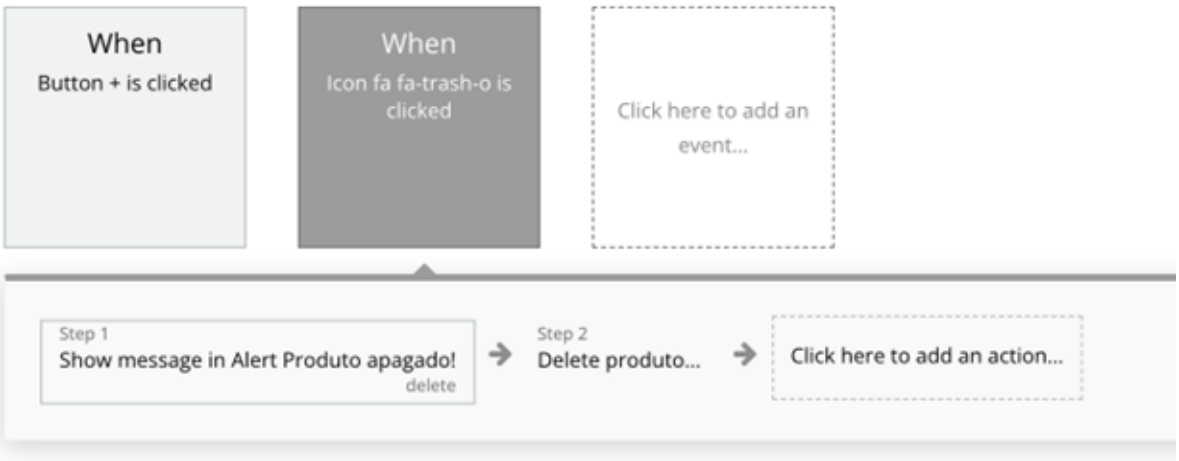


Para ficar um pouco melhor, vamos trabalhar na cor de fundo desse alerta. Fique a vontade para colocar o design do alerta do jeito que você achar mais interessante.

Uma dica interessante é que podemos personalizar a mensagem para indicar qual item foi deletado!

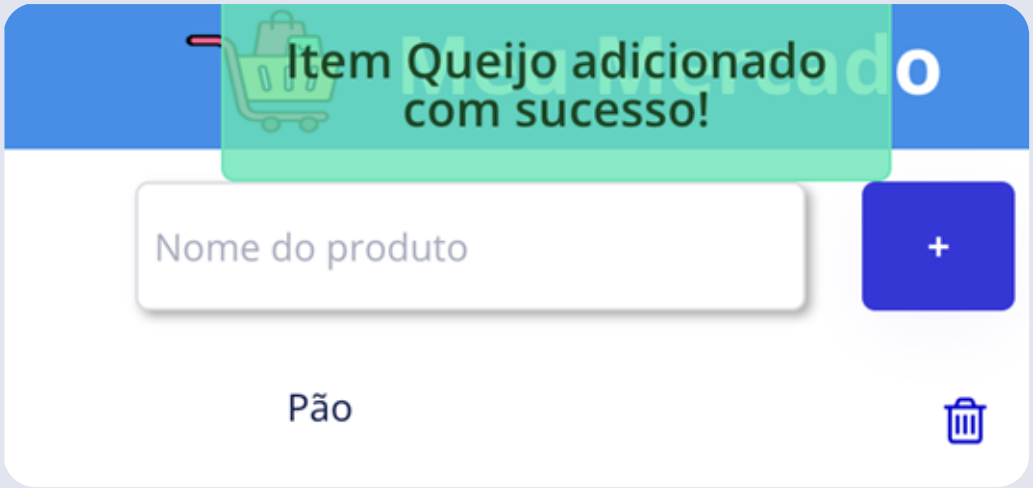


Só que veja só. Se mantivermos o fluxo dessa forma, estamos apagando o item e depois mostrando a mensagem. O que vai acontecer com nossa variável dinâmica? Ela estará vazia! A solução é trocarmos a ordem das ações.



Exercício

Agora é com você! Altere o alerta de Produto adicionado para ele também aparecer o nome do item que foi incluído em nossa lista.

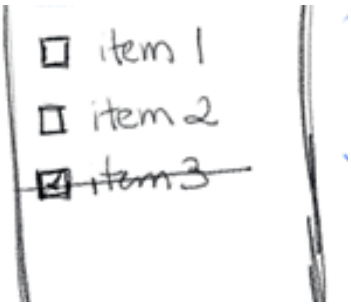


Modificar o estado de um item

Agora que já conseguimos colocar e retirar os itens da nossa lista, podemos passar para os próximos itens do nosso escopo de projeto.

- Uma função para quando clicar no item, vai marcar como comprado;
- Uma função para quando clicar no item já comprado, desmarcá-lo.

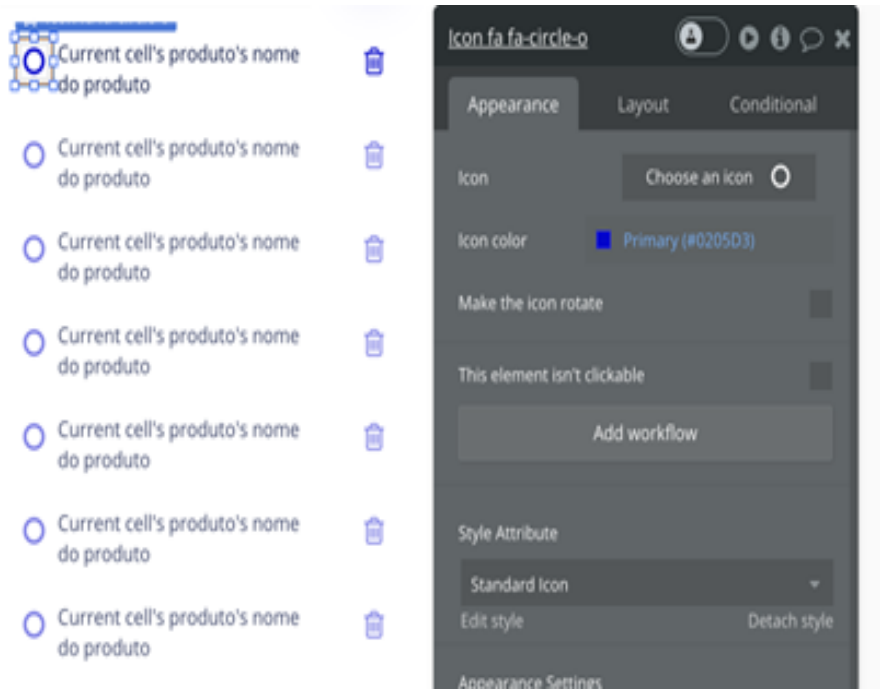
De acordo com o nosso rascunho, agora vamos riscar os itens já comprados.



Isso é interessante em uma lista de compras, pois às vezes repetimos muito alguns itens e não necessariamente precisamos apagá-lo do banco de dados. Vamos apenas marcar com um "comprado". Em uma próxima ida ao mercado, podemos clicar no item para ele voltar para nossa lista de compras.

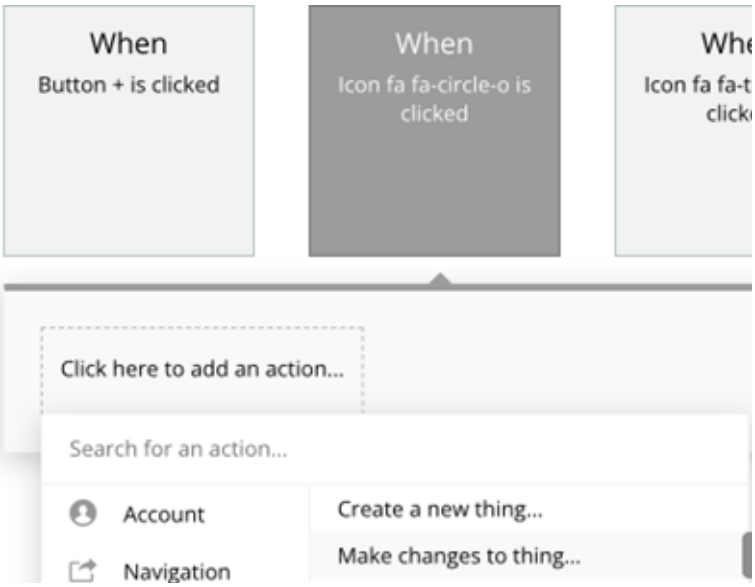
Para fazer isso teremos que acrescentar a informação de estado **(status)** no nosso produto. Logo, vamos estruturar nosso banco de dados. Vamos criar um novo campo **comprado** que será booleano, ou seja, verdadeiro ou falso. No Bubble, será um campo do tipo yes/no.

Pronto! Agora vamos para a interface. O que falta colocar do nosso rascunho é a checkbox que mencionamos lá atrás. Lembrando que basta colocá-la na primeira célula do grupo repetidor. Porém, não vamos usar o elemento **checkbox**. Por que isso? Pois a **checkbox** não se coloca workflow, e aqui nós queremos criar esse fluxo de trabalho. Então vamos colocar um **icon**, assim como fizemos com a lixeira.

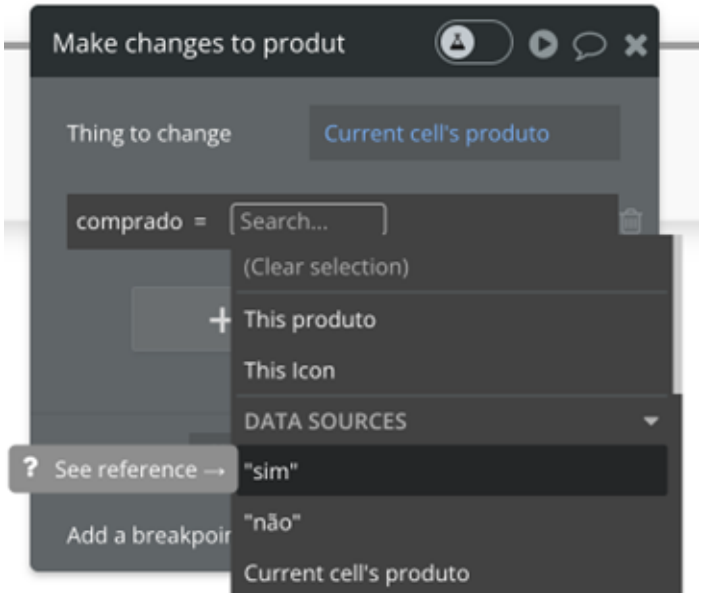


E vamos adicionar um workflow para ele! Aqui temos algo bem interessante para fazer. Em que situação nós devemos riscar nosso item? **Se** comprado for verdadeiro então risca o produto. Isso é uma **condicional**.

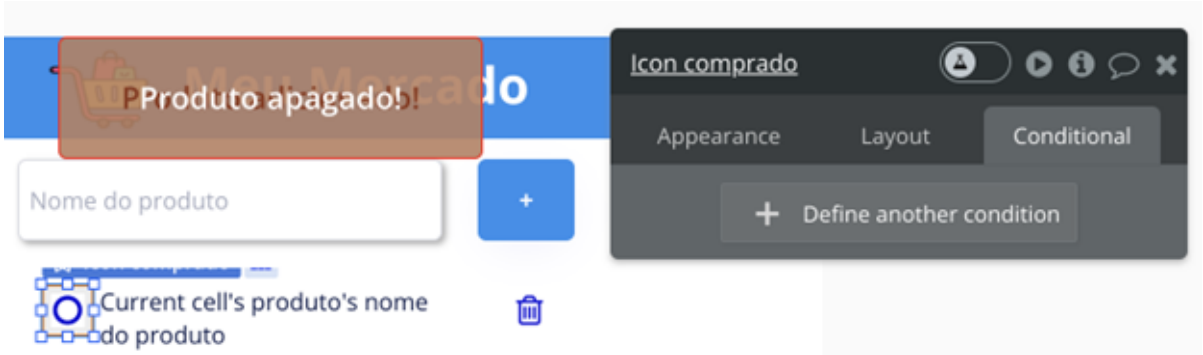
A primeira coisa que precisamos definir é a ação de clicar. Quando clicar, vamos alterar no nosso banco de dados. Nós vamos usar a ação de **Data > Make changes to thing**. Pois ao clicar queremos que nosso campo comprado vire SIM. E se clicarmos novamente ele virará NÃO.



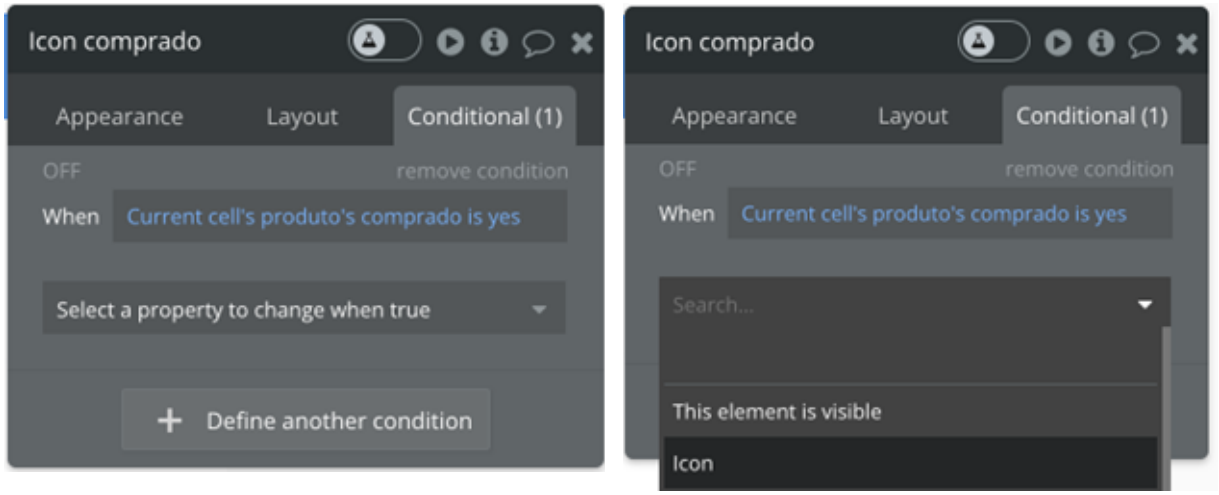
E agora precisamos definir a nossa condicional. Por que? Qual o valor de comprado? Vai depender do valor anterior dele. Se era sim, passa para não. Se era não, passar para sim.

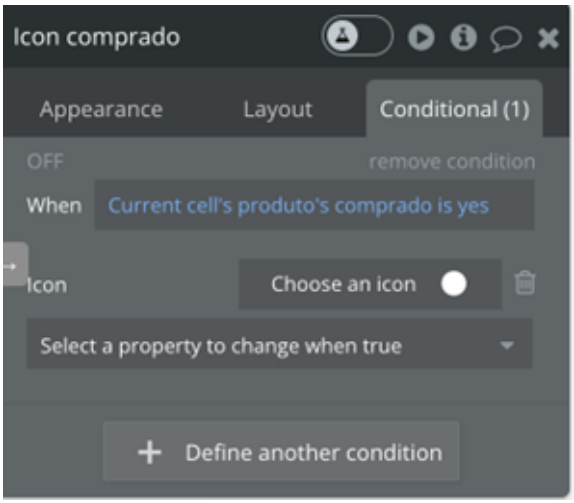


Então vamos aprender a usar a **condicional** no Bubble. As condicionais ficam sempre na janela de propriedades. Então clique no nosso ícone de marcação e vá em **Conditional** e clique em **Define another condition**.



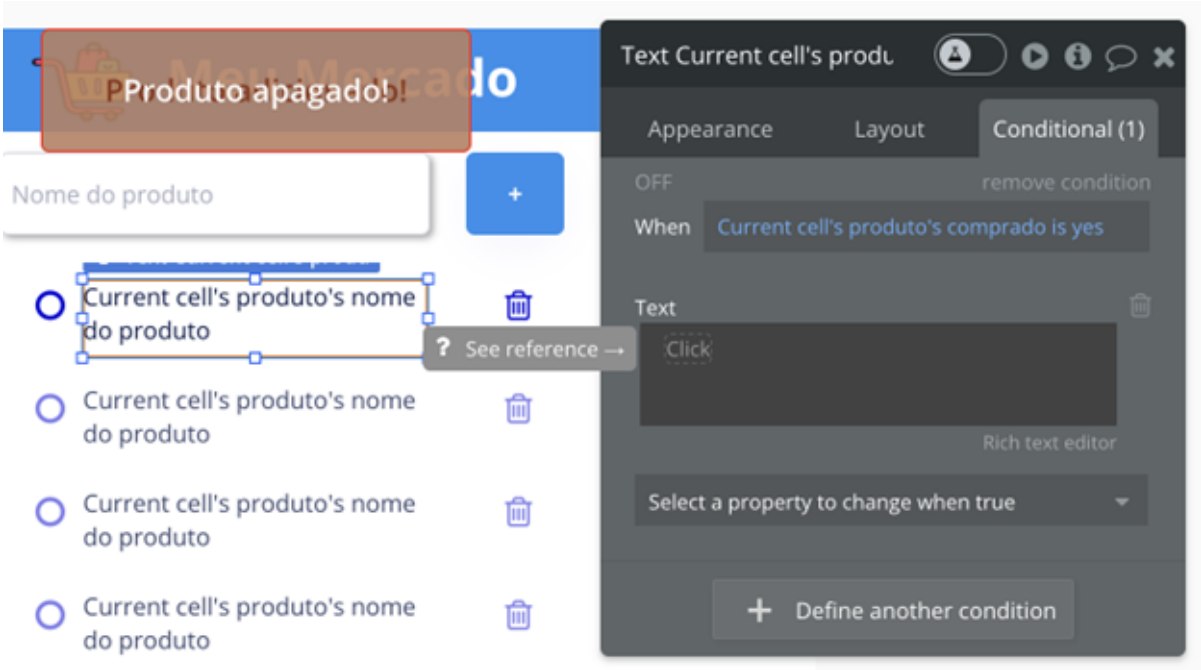
E vamos fazer com que quando o valor de comprado for sim, vamos mudar o ícone para a bolinha toda colorida.



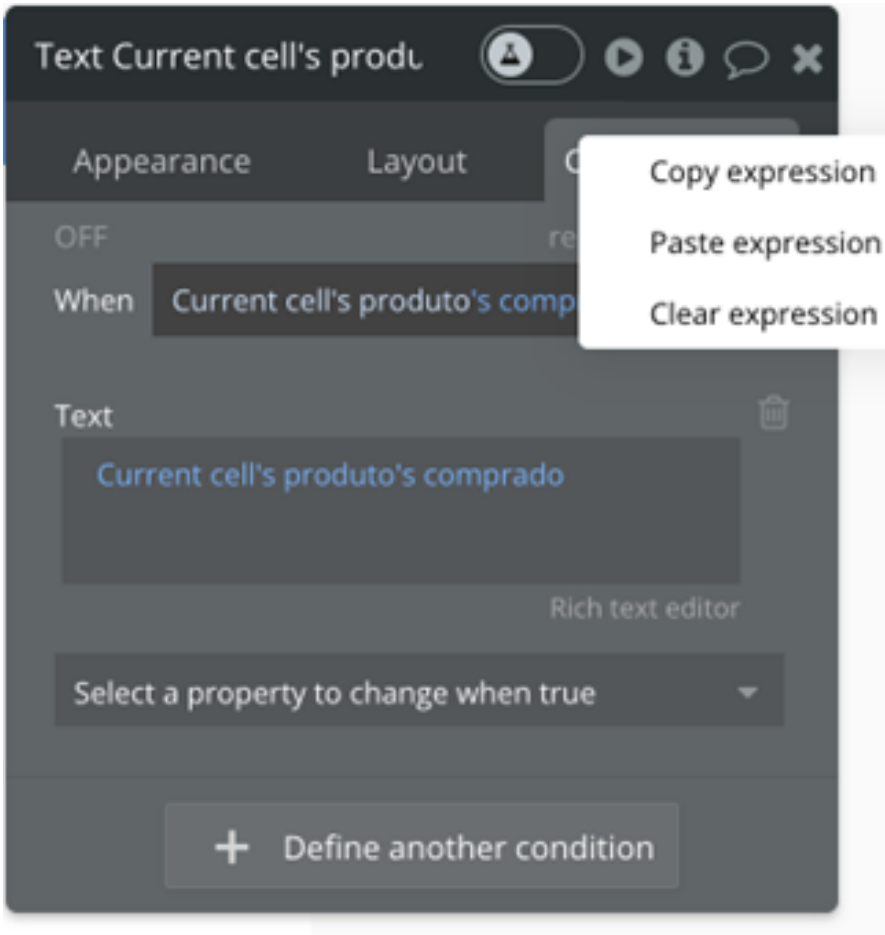


Então nosso aplicativo até o momento está assim:

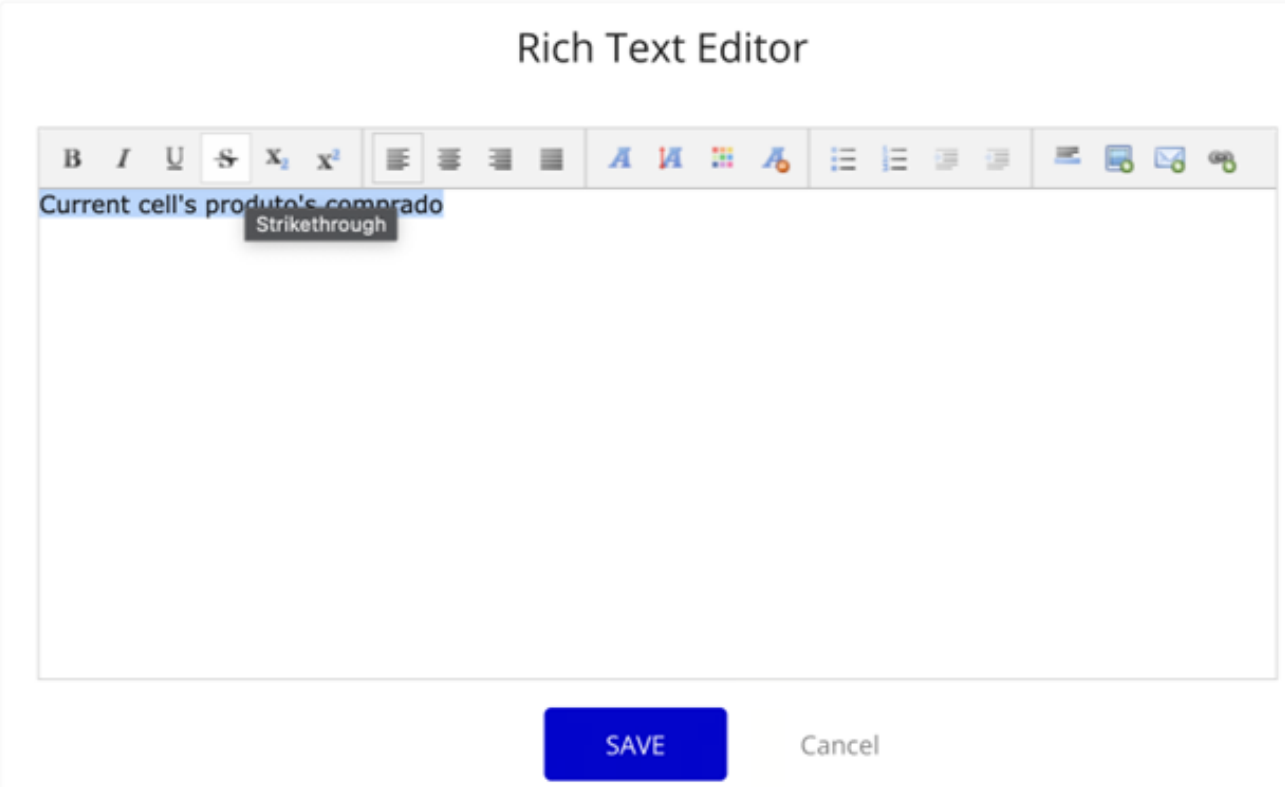
Agora vamos riscar o nome do produto quando comprado for sim. Então faremos o mesmo caminho. Clicamos no item do nome, selecionamos condicional e criamos a nossa nova condição. Normalmente, esse efeito de riscar o texto é chamado **Strikethrough**. Infelizmente no Bubble não tem como alterar a fonte diretamente assim na condicional. Então vamos criar um novo texto com essa característica.

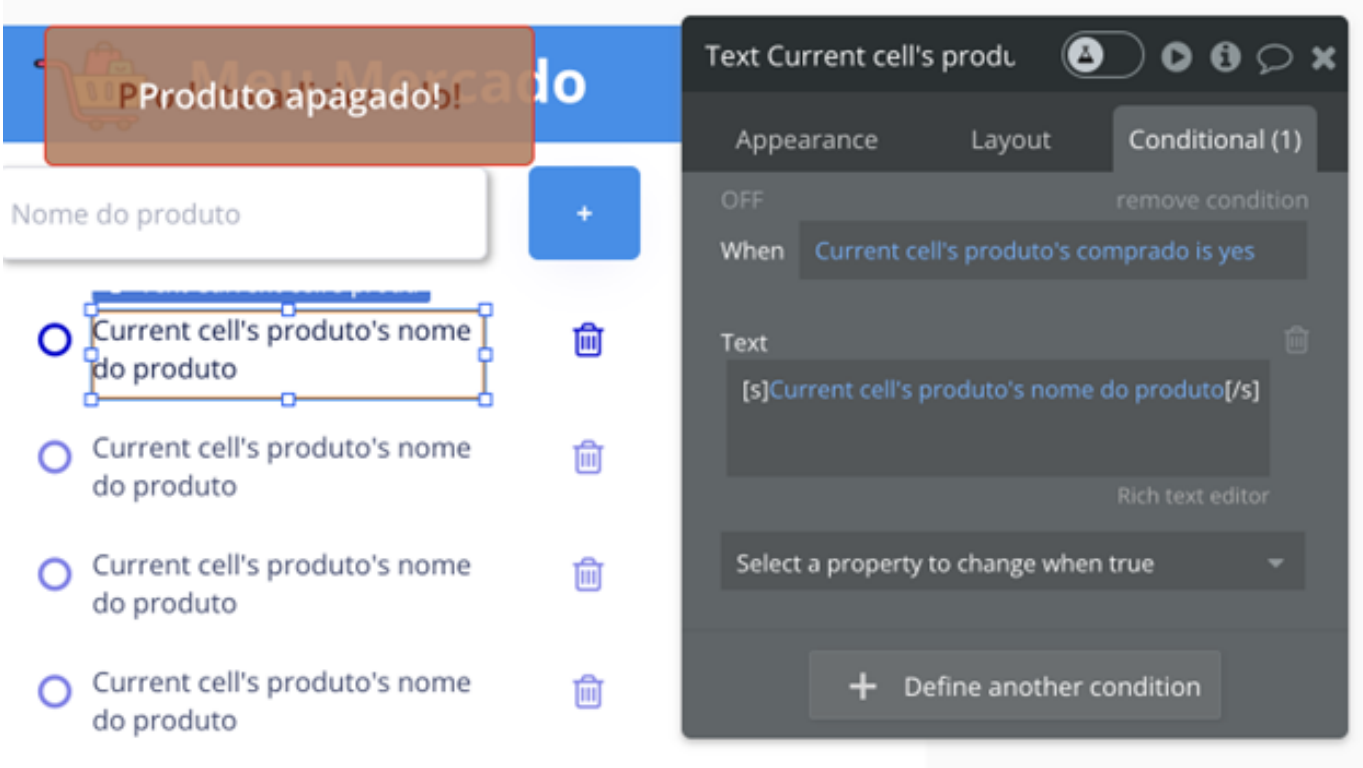


Clique com o botão direito em **Current cell's produto** e **copy expression**. Vamos colar neste campo **Text** logo abaixo. Lembre de trocar comprado para **nome do produto**.



Em seguida vamos clicar em **Rich text editor**. Nele vamos marcar o **strikethrough**.

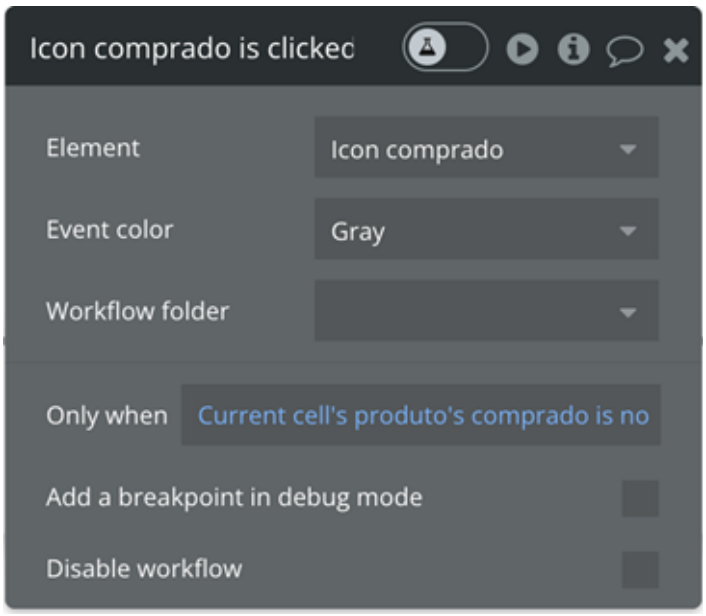




E agora nosso aplicativo está assim:

Agora precisamos **desmarcar**. Então vamos trabalhar no fluxo do nosso ícone comprado: Se era sim, passa para não. Se era não, passar para sim. Essa condicional que iremos criar não será mais na propriedade do elemento da tela, vamos lá no gatilho da nossa ação no **workflow.**

Clique no gatilho **When** no nosso ícone comprado. E vamos criar em sua janela de propriedades uma condicional **Only when**. Assim, somente quando o campo estiver com não, vamos mudá-lo para sim.



Agora vamos copiar esse gatilho e fazer o inverso. Quando for não, vire sim.

Prontinho! Vamos avaliar nosso escopo:

- Um campo de formulário para preencher com o nome do produto e um botão adicionar - FEITO
- Uma lista com todos os produtos já adicionados - FEITO
- Uma função para quando clicar no item, vai marcar como comprado - FEITO
- Uma função para quando clicar no item já comprado, desmarcá-lo - FEITO

Parece que finalizamos nossa primeira ideia, nosso MVP. Claro que podemos melhorar muitas coisas na nossa lista de supermercado, por exemplo: não aceitar itens repetidos ou vazios, colocar os itens comprados no final da fila, editar os itens e até acrescentar um login de usuário. Na próxima seção vamos tratar das melhorias do nosso aplicativo e avançar para a versão 1.0 da nossa lista de supermercado.

Exercício

Agora é com você! Envie o link da sua lista de supermercado até o momento.

Estudo de caso: Aprimorando a lista de supermercado

Vamos então aumentar o escopo do nosso projeto. Agora teremos as seguintes funcionalidades a mais:

- Quando um item for riscado, ir para o final da lista;
- Ordenar os itens por ordem alfabética;
- Editar um item;

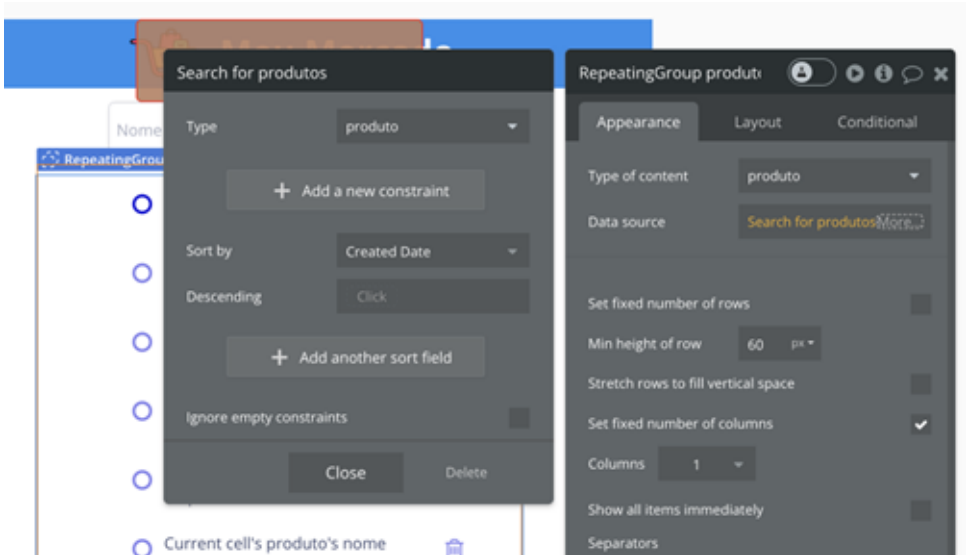
Ordenando a lista

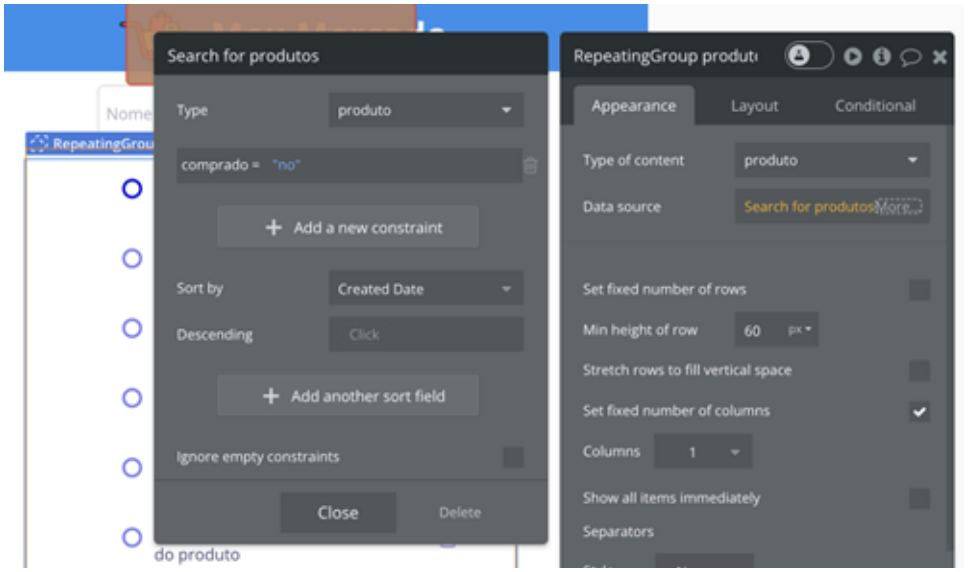
Nosso aplicativo está dessa forma até o momento:

Nosso primeiro aprimoramento será passar os itens riscados para a parte final da lista. Para isso vamos trabalhar com duas listas distintas: Uma lista de não comprados e uma de comprados. Logo, vamos criar mais um **Repeating group** para que possamos tratar de cada caso.

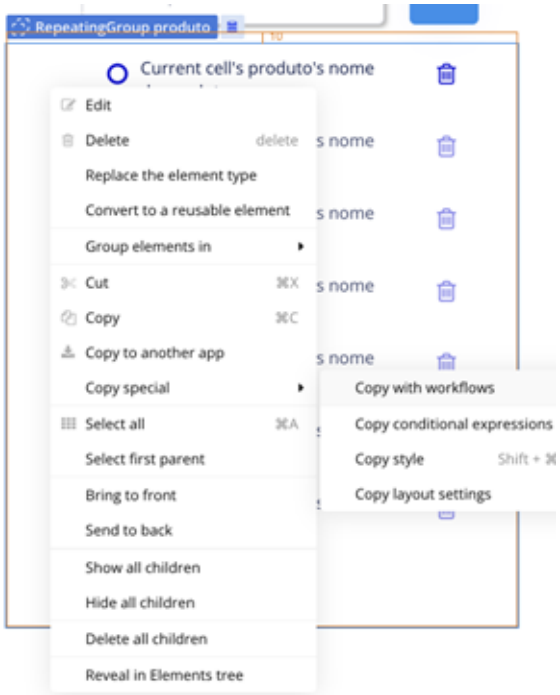
Então vamos ao Bubble e vamos clicar no nosso grupo repetidor. Inicialmente vamos **filtrar** os dados que são mostrados, afinal, queremos mostrar os itens separados pelo valor do campo comprado da nossa tabela produto.

Essa filtragem é realizada diretamente na busca, na propriedade **Data source > Search for produtos**. Na janela que abrir, vamos clicar em **Add a new constraint** (adicionar uma nova restrição). Aproveite e repare como já estamos ordenando os produtos por ordem de criação **(Sort by Created Date)**.

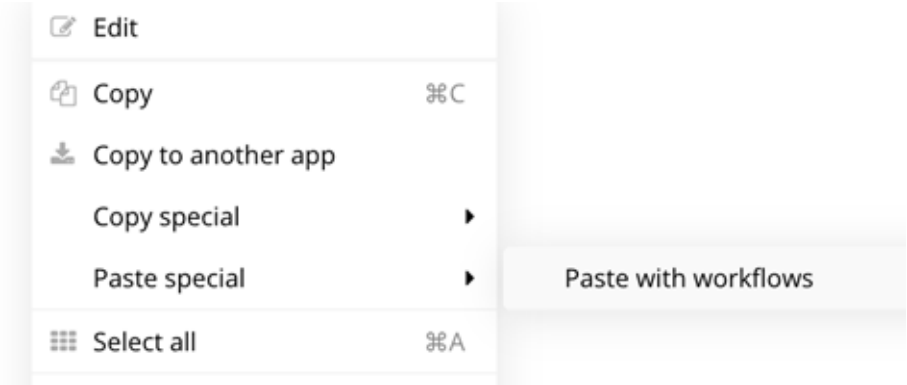




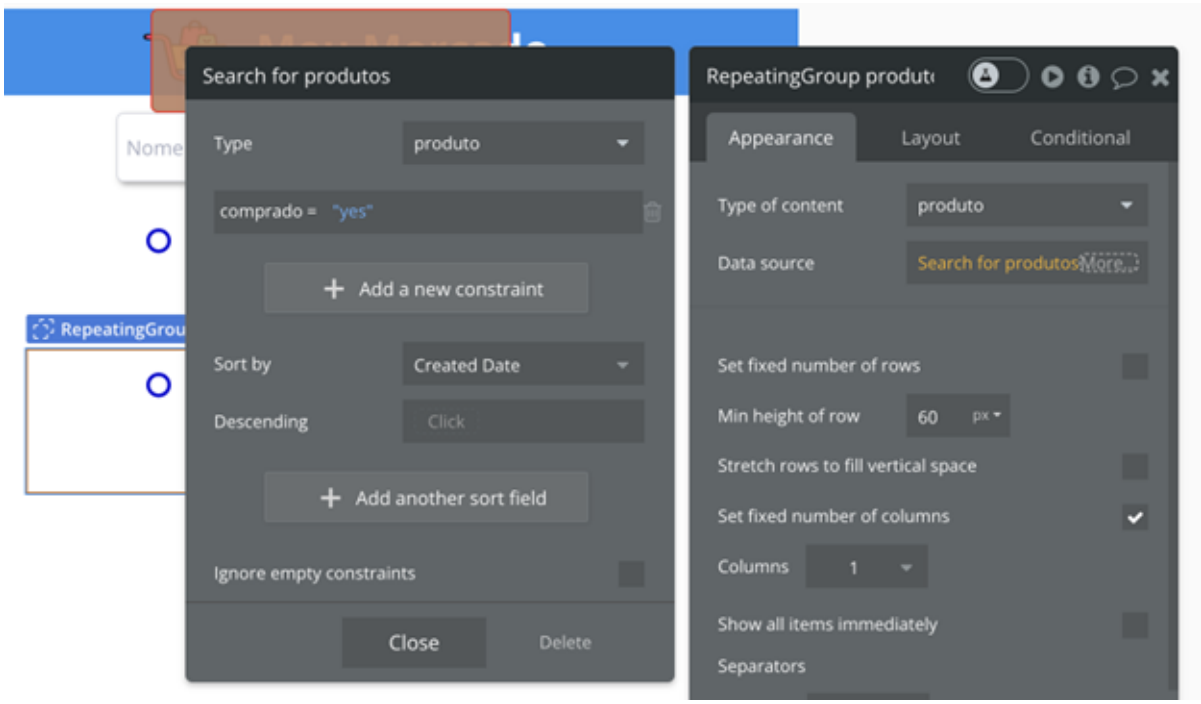
Atualizando nosso **Preview**, veremos que agora só são mostrados os produtos que ainda não foram comprados. Agora podemos criar o próximo grupo repetidor, que terá como filtro/restrição ser um produto comprado. Aqui podemos copiar o grupo já existente e alterar essa regra. IMPORTANTE: aqui precisamos copiar com os workflows! Logo, clique com o botão direito e selecione **Copy Special > Copy with workflows**.



Depois, basta colar abaixo do nosso grupo repetidor.



Vamos então alterar a restrição para comprador = yes.

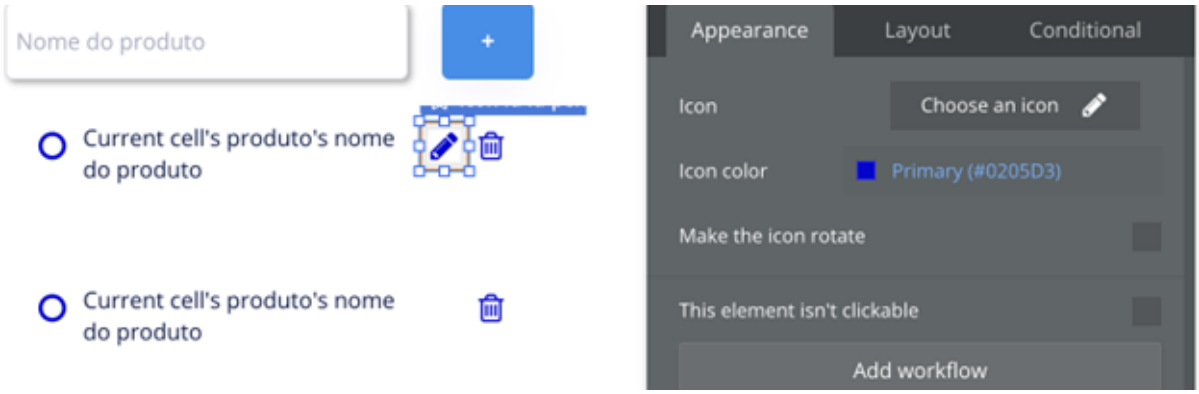


Então nossa nova visualização será:

Agora, para melhorar nossa visualização e ficar mais fácil achar um item na nossa lista de supermercado, vamos alterar o **Sort by** do nosso grupo repetidor para ordenar pelo nome do produto. Assim, veremos os itens em ordem alfabética.

Editando um item

Vamos agora fazer a ferramenta de edição do nosso item. Então a primeira coisa que vamos fazer é colocar o ícone de edição na célula do grupo repetidor. Vamos colocar apenas no grupo dos produtos não comprados.

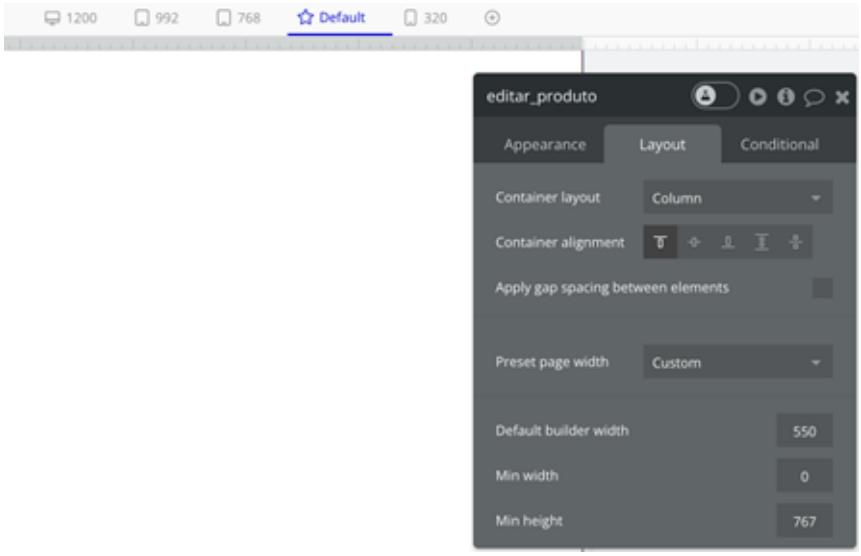


Agora precisamos pensar como vamos abrir nosso produto para edição: Vamos mudar na própria tela? Vamos abrir uma tela própria para isso? No primeiro caso nós trabalhamos com o conceito de **Página Dinâmica** e no segundo caso trabalhamos com o conceito de **Popup**.

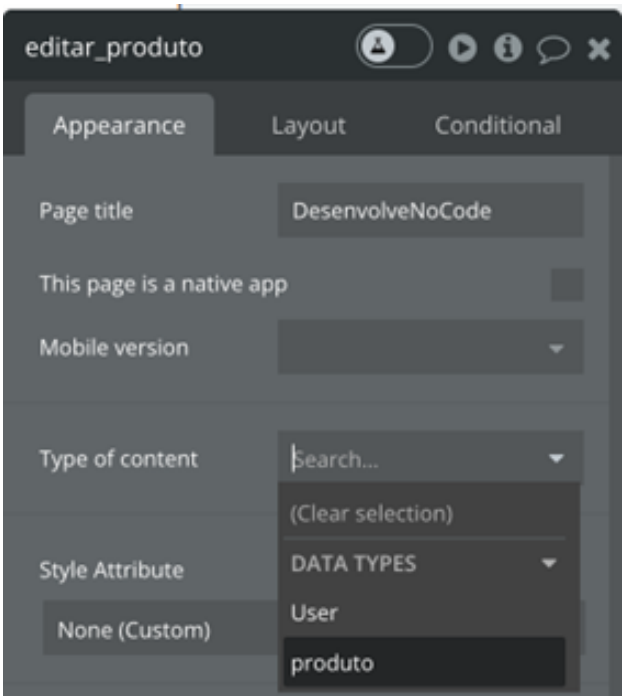
Página Dinâmica

Até o momento temos apenas uma página no nosso projeto, a index. Para entendermos o conceito de página dinâmica, vamos criar uma nova no Bubble.

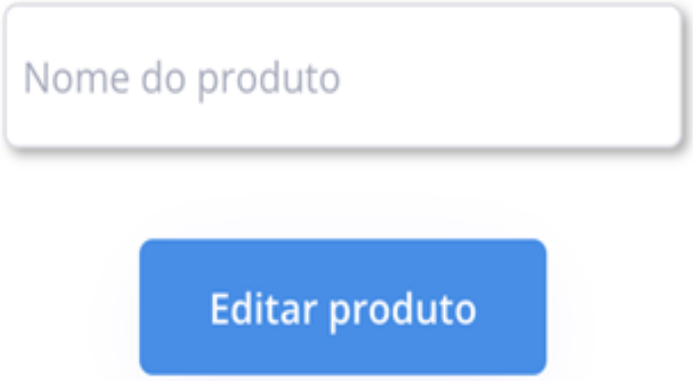
Vamos configurar o layout dessa nova página igual fizemos com a primeira: colocando responsividade, layout de coluna e largura de 550px.



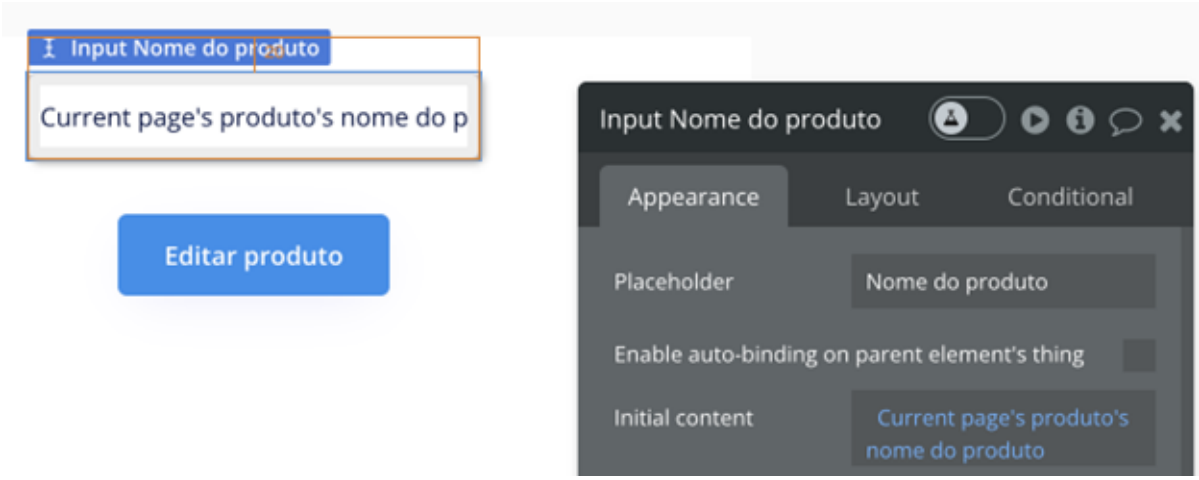
Quando nós vamos para a aparência da página **(Appearance)**, temos uma propriedade chamada **Type of content**, tipo do conteúdo. Ao definir o tipo de conteúdo você transforma a página em uma página dinâmica: Ela é uma página que vive em função de um determinado item do banco de dados. Nossa página será do tipo **produto**.



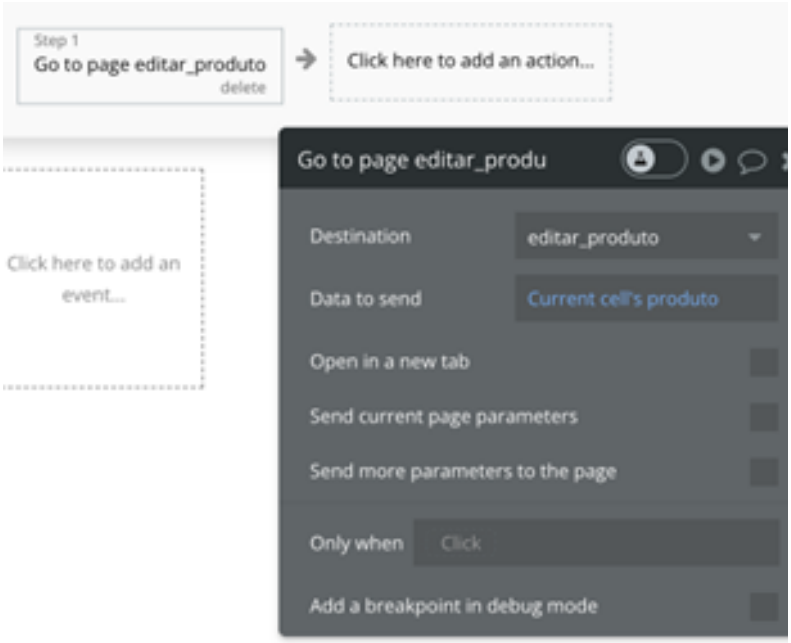
Essa página será formada por um campo e um botão: o campo para alterar o nome do produto e o botão para salvar. Aqui podemos copiar o **Input** da nossa index para ficar tudo padronizado, assim como o botão. Lembre de copiar de maneira simples, pois faremos novos workflows.



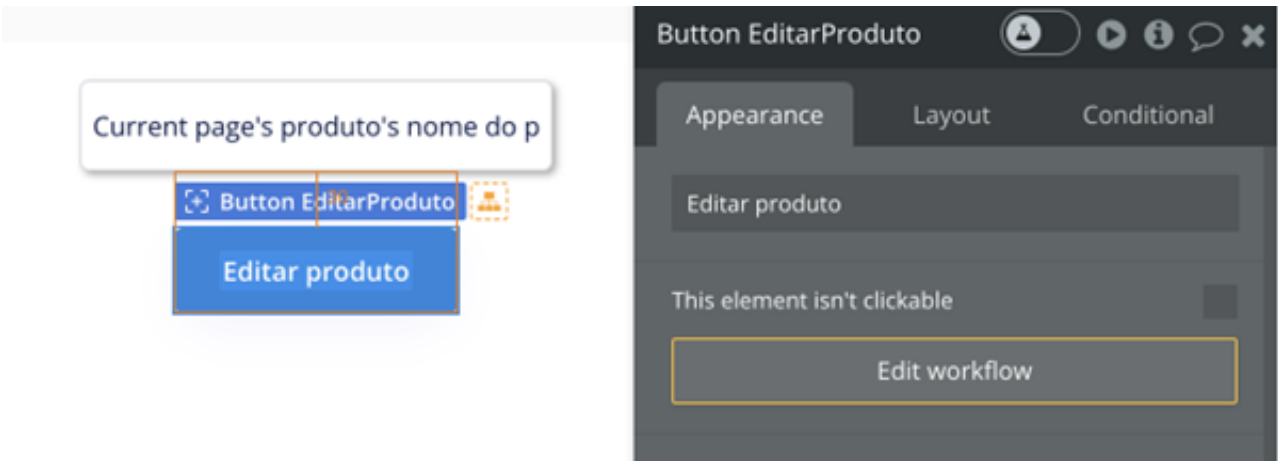
Como nós transformamos esta página em dinâmica, nós podemos manipular outros tipos de propriedades. O campo **input** agora tem a propriedade de **initial content**, ou seja, um conteúdo inicial. Assim, ele já vai mostrar o valor do nome do produto que queremos editar: **Current page's produto's nome do produto.**



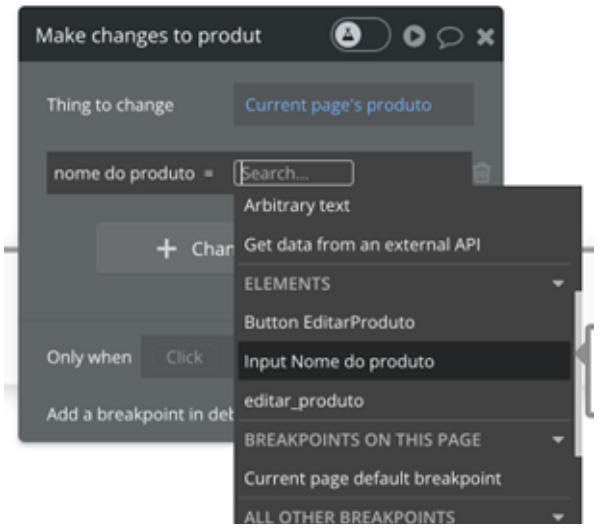
Então agora podemos ajustar o **workflow** no nosso ícone de edição. Quando clicarmos no ícone de editar produto, queremos que essa nova página seja aberta. Para abrir essa nova janela, nós vamos enviar o dado produto para ela. Assim, **Data to send** se torna **Current cell's produto**.

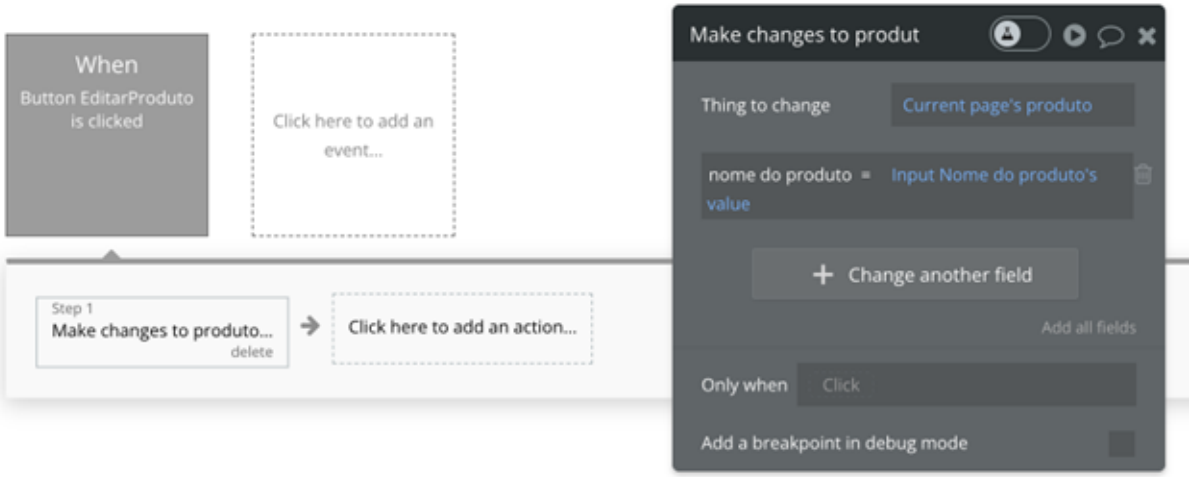


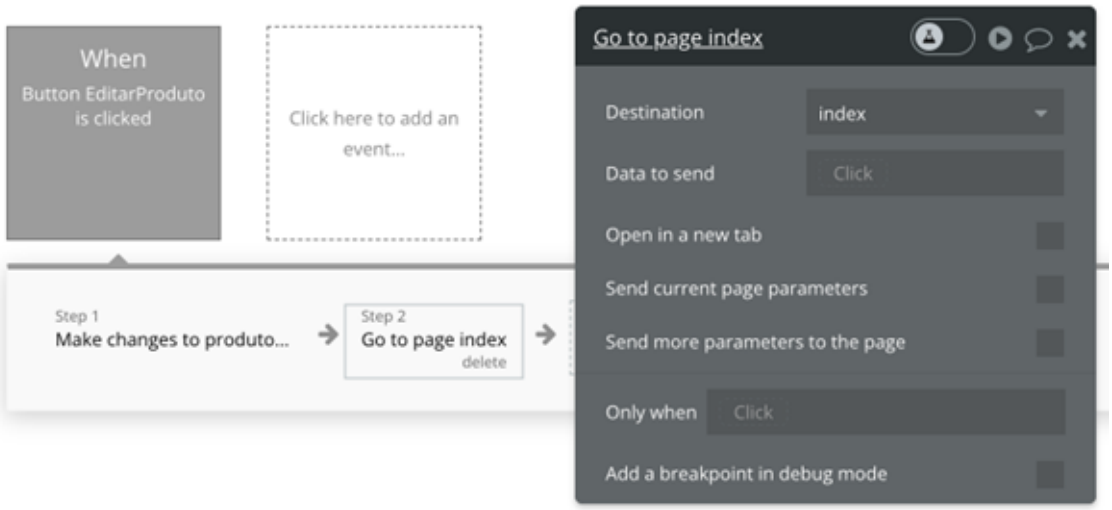
Pronto! Agora ao clicar no ícone de edição, a nova página será aberta com o produto carregado no input. Agora podemos fazer o workflow do botão desta página.



Nós vamos montar duas ações: modificar o dado e voltar para a página inicial.







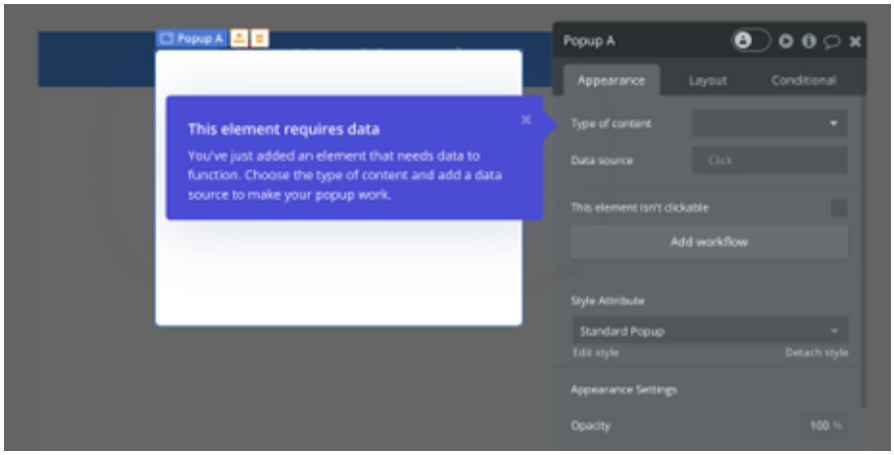
Esse tipo de navegação para páginas dinâmicas é muito útil na criação de aplicativos. Porém, para aplicativos simples como essa nossa lista, onde só queremos modificar um item, existe uma forma também mais simples de trabalhar.

Popup

Popup é um elemento que fica em **Containers**. Como o próprio nome indica, ele é um elemento que salta da tela, mas não é necessariamente uma página nova cheia de informações. Para alterações simples e mensagens curtas, ele é muito indicado.

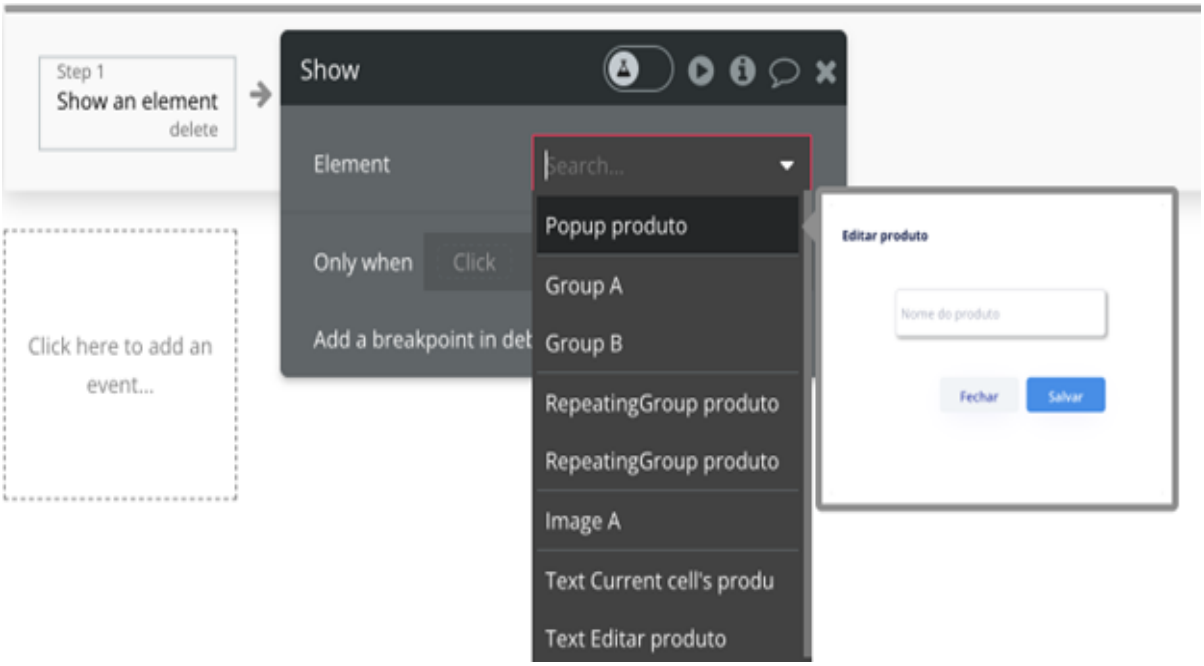
Por ser um **container**, sua forma de trabalho é igual a todas as outras: basta arrastar os elementos para ele.

Ele será colocado na nossa própria index e repare como ele já vem como uma página dinâmica, ou seja, precisa de um tipo de conteúdo - que será o produto. Não precisamos definir o **Data Source** pois ele será enviado da página index.

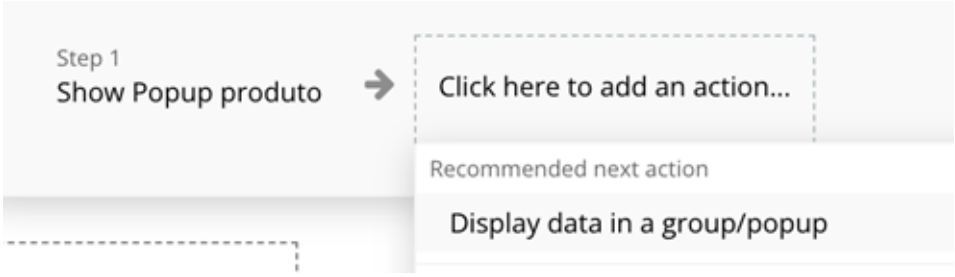


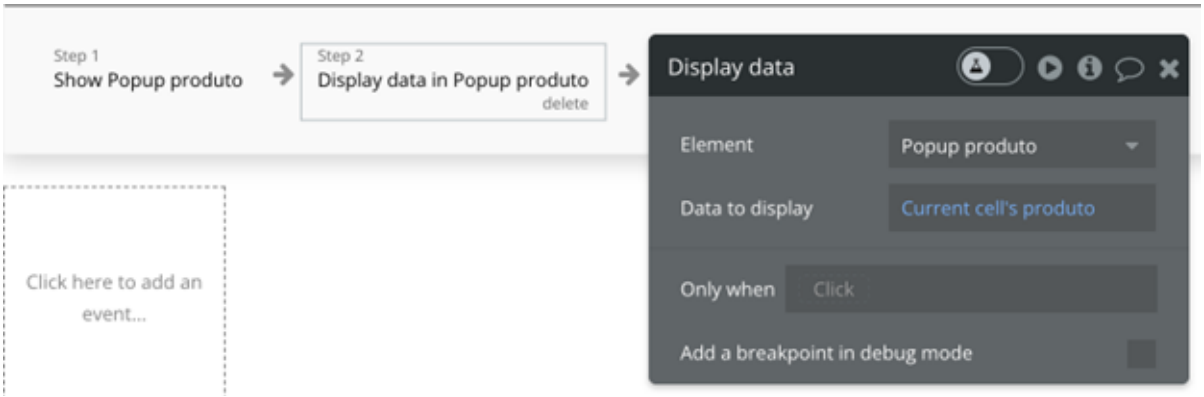
Vamos então montar a interface do nosso popup.

Agora vamos alterar o **workflow** do nosso ícone de edição para que ele abra o **popup**.

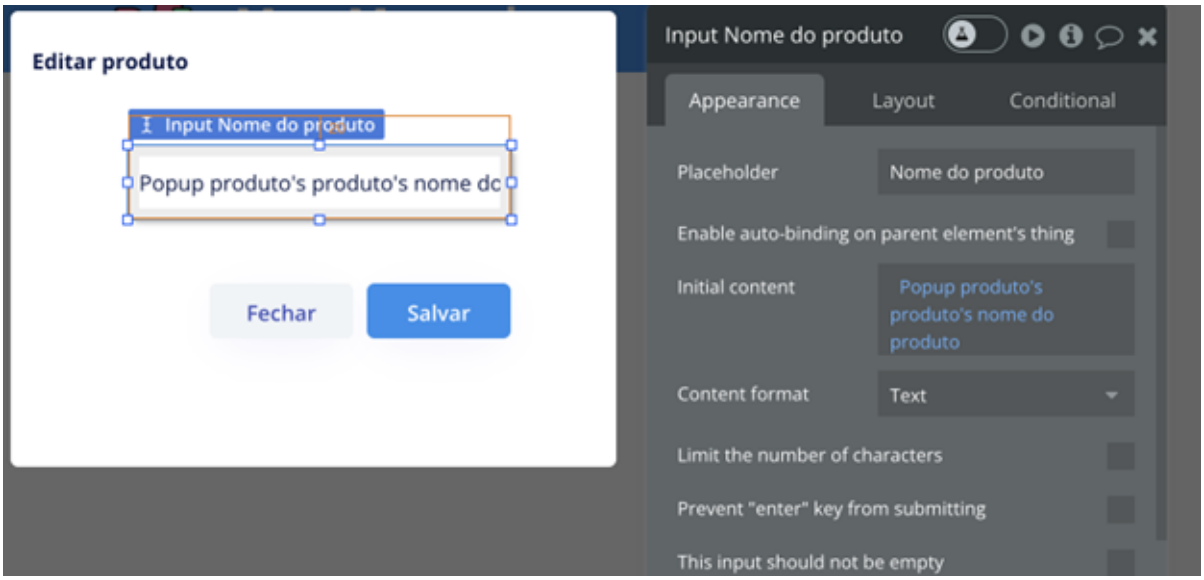


A segunda ação será **Display data**, para passar o dado para o popup. Essa ação aparece como recomendação do Bubble.



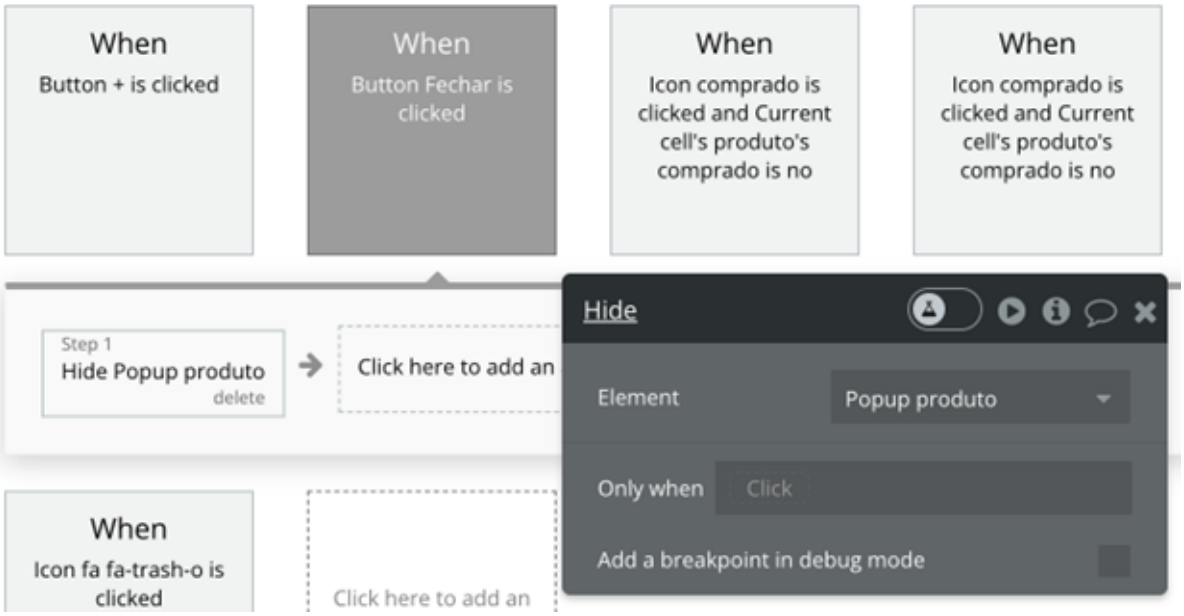


Agora no **Input** do **Popup** podemos pegar o **Initial content**.

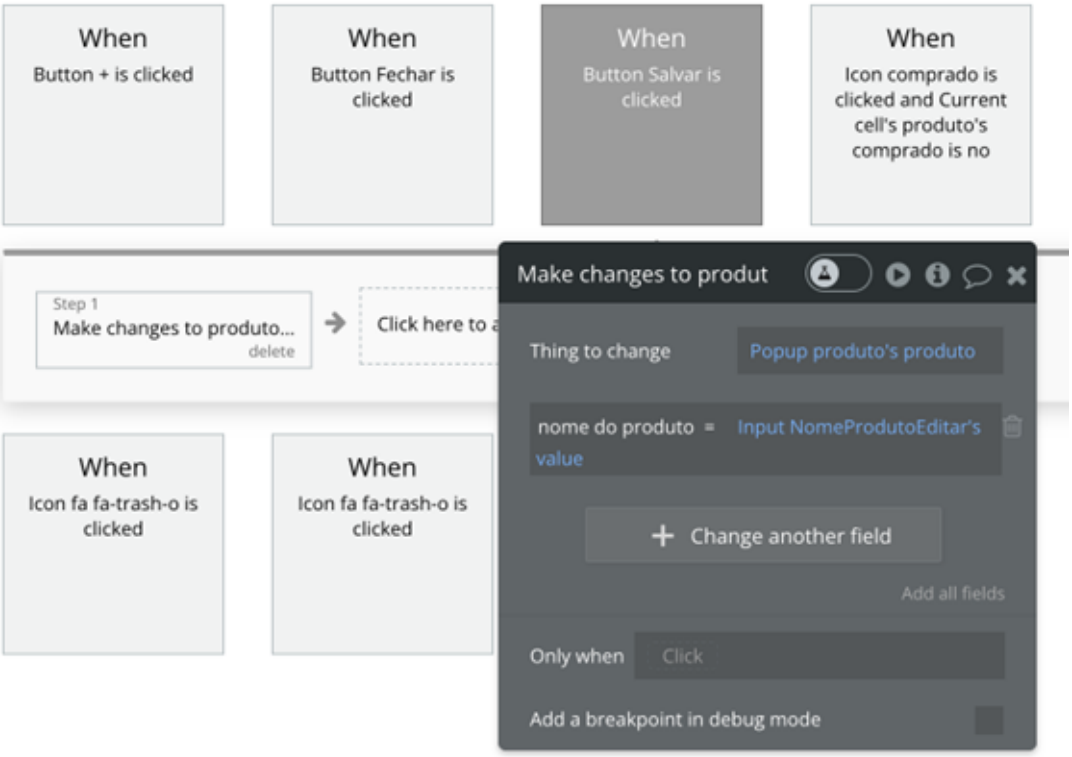


Teste agora no **Preview**. Veja como a visualização é mais simples.

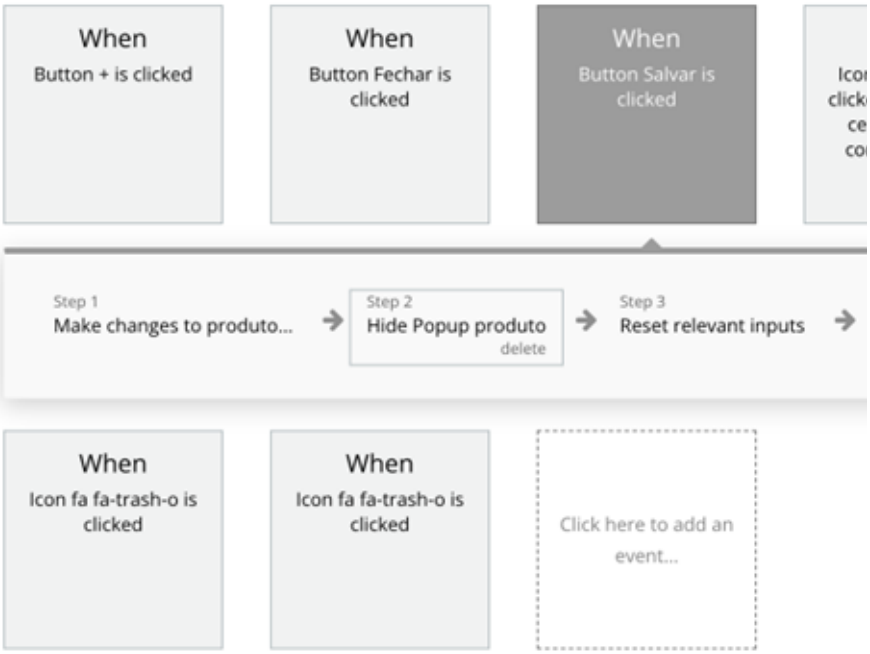
Agora podemos ajustar o workflow dos botões. O Botão Fechar vai esconder a tela do Popup e o Botão Salvar vai fazer o mesmo Workflow que fizemos na página de edição que criamos na seção anterior.



Muita atenção ao nome dos elementos! Sempre coloque nomes únicos para não se perder.



Além dessa ação, vamos esconder o popup e resetar os campos relevantes.



Agora você já sabe duas formas de passar dados e de gerar novas páginas. Isso vai te dar a liberdade de fazer diversos tipos de aplicativos!

Exercício

Agora é com você! Envie o link do seu projeto completo.

Implantação(Deploy)

Implantar seu aplicativo significa transferir ou atualizar a versão atual de sua aplicação do ambiente de Desenvolvimento para o ambiente de Produção (às vezes chamado de Live), tornando-a acessível aos usuários finais. Em outras palavras, este é o ponto em que você publica seu aplicativo para os usuários que realmente o utilizarão. Implantar um aplicativo no Bubble é feito com um clique de botão.

Embora você tenha a liberdade de implantar seu aplicativo quando quiser, o Bubble exige que o rastreador de problemas **(issue tracker)** não esteja sinalizando nenhum problema. Se o rastreador de problemas estiver mostrando algum problema, esses problemas precisam ser resolvidos antes de implantar.

Implantar seu aplicativo é um marco importante, pois significa que suas últimas alterações estão indo ao ar para sua audiência. Esse evento merece um planejamento cuidadoso, pois traz suas atualizações e melhorias diretamente aos usuários finais. A quantidade de testes e revisões necessários antes da implantação depende de você. Se houver alterações significativas e/ou mais de uma pessoa estiver envolvida no desenvolvimento do aplicativo, é recomendável tomar precauções antes da implantação.

Uma lista de verificação sugerida que pode ajudar na organização desse processo é mostrada abaixo. Você pode adaptá-la conforme necessário para atender às necessidades de sua equipe e projeto.

- **Revisar alterações:** Recapitular todas as atualizações desde a última implantação.

- **Teste funcional:** Testar todas as funcionalidades recém-adicionadas.

• **Teste de regressão:** Verificar funcionalidades previamente operacionais quanto a possíveis problemas inesperados.

• **Verificação do banco de dados:** Confirmar que a estrutura do banco de dados no ambiente de Desenvolvimento é compatível com o ambiente de Produção (por exemplo, talvez seja necessário copiar dados entre os dois ambientes).

• **Responsividade:** Verificar a aparência e sensação do aplicativo em diferentes dispositivos e tamanhos de tela.

• **Acesso do usuário:** Validar funções e permissões dos usuários, testando o aplicativo com diferentes perfis de usuários.

• **Integrações:** Testar conexões de API e plugins.

• **Revisão por pares:** Solicitar a revisão das alterações por um membro da equipe ou colega.

Embora nem todos os pontos sejam necessários em todas as ocasiões, recomenda-se adotar uma abordagem sistemática para a implantação, a fim de minimizar a chance de os usuários ativos do seu aplicativo enfrentarem problemas.

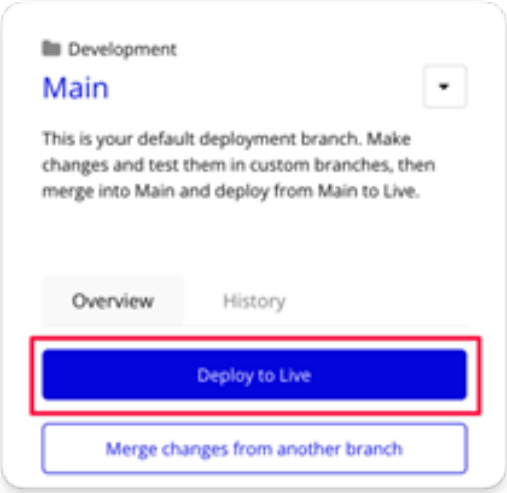
Quando você implanta seu aplicativo, os usuários verão instantaneamente uma notificação na parte superior da tela informando que precisam atualizar a aba. Clicar na aba atualizará a página atual. Usuários que não tiverem o aplicativo aberto no momento da implantação não perceberão nada.

Seus usuários não poderão usar o aplicativo até que tenham atualizado a página. Portanto, o momento da implantação pode ser relevante. Se possível, planeje suas implantações em horários de baixo uso do aplicativo.

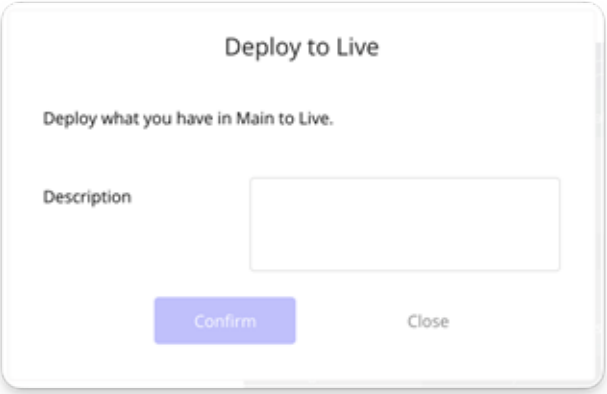
Então como dar o nosso deploy?

Quando o rastreador de problemas estiver em zero e você já puder implantar, você pode iniciar o processo. A implantação do seu aplicativo é praticamente instantânea para todos os fins práticos. Primeiro, clique no botão de controle de versão no canto superior direito da barra de ferramentas do Bubble.

O painel de controle de versões será aberto



Clicar no botão **Deploy to live** iniciará o processo de implantação. Se houver algum problema impedindo a implantação, você será notificado neste ponto. Por fim, você pode fornecer uma descrição da implantação para documentar as alterações que você fez:



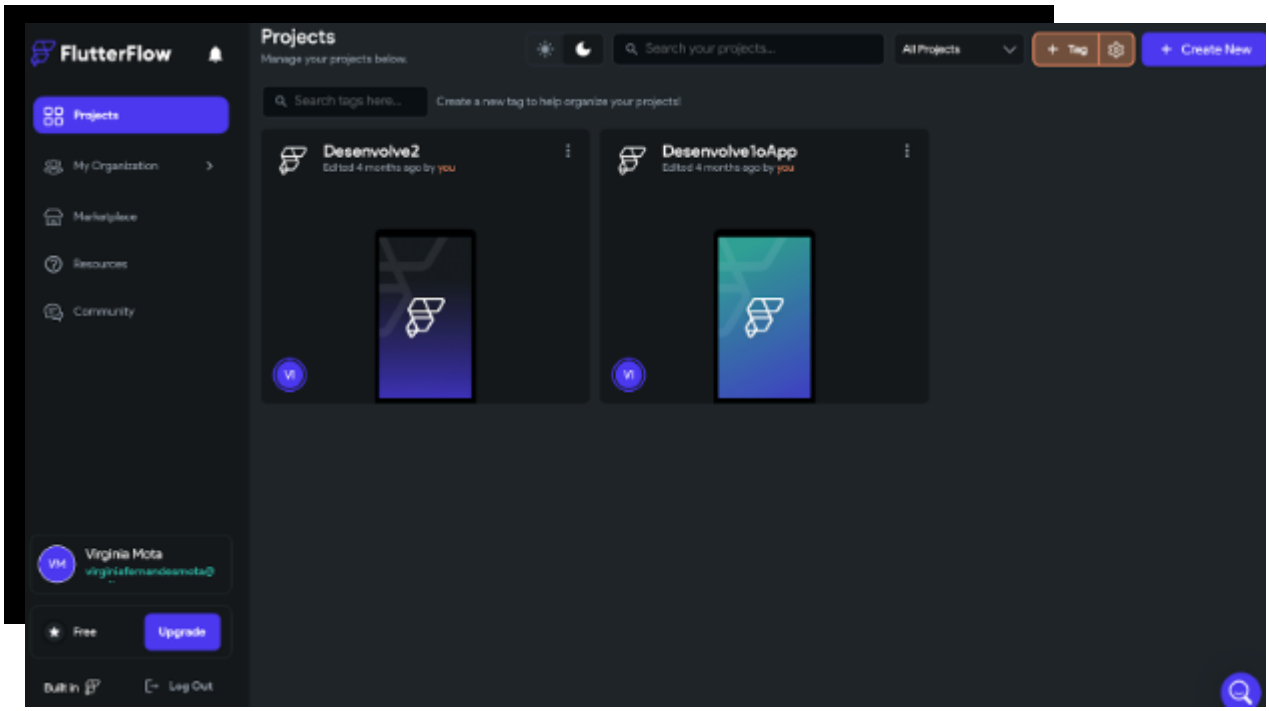
Essa descrição será salva junto com o ponto de salvamento automaticamente. Dessa forma, se uma implantação introduzir algum problema, você poderá reverter para uma versão mais antiga, se necessário.

Se você trabalha com uma equipe ou em um aplicativo particularmente complexo, pode estar utilizando o sistema de controle de versão do Bubble para isolar recursos ou projetos específicos enquanto estão em desenvolvimento.

E aqui a gente encerra com um problema de ferramentas no code: preço. Para implantar seu aplicativo, você deve passar para a versão paga. Mas não se preocupe! Você já foi capaz de criar um app do zero e pode passar o link para testes!

Parte III FlutterFlow

Nesta parte do curso vamos trabalhar com a ferramenta FlutterFlow. Você já fez a sua conta, então a primeira tela que irá aparecer para você será em: https://app.flutterflow.io/



Todos os aplicativos que você criar, irão aparecer nesse Dashboard da plataforma.

O FlutterFlow é uma ferramenta low-code com algumas peculiaridades em relação ao Bubble. Então vamos entender alguns aspectos primeiro. O FlutterFlow é um aplicativo web que pode ser utilizado pelo seu navegador. Por favor, revise esses requisitos de sistema para obter a melhor experiência:

• Suportado em qualquer laptop ou desktop.

• Recomenda-se o uso de uma tela com resolução de pelo menos 1280 x 1084.

• O FlutterFlow funciona melhor no Google Chrome (se estiver enfrentando problemas, recomendamos mudar para o Chrome).

• O FlutterFlow não serve especificamente para a criação de sites, pois a sua espe-

cialidade é o desenvolvimento de aplicativos mobile nativos para Android e iOS - A plataforma é uma excelente solução para criar aplicações móveis multiplataformas, mas não é a melhor ferramenta para criar sites tradicionais.

• Por ser uma ferramenta low-code é possível inserir códigos CSS, HTML, Javascript e principalmente Dart no FlutterFlow - Você também consegue importar todos os packages Flutter para dentro de seu aplicativo FlutterFlow.

• Assim como no Bubble, para publicar seu aplicativo você precisa migrar para um plano pago.

Antes de partirmos para entender a nova ferramenta, vamos comentar as diferenças entre o Bubble e o FlutterFlow.

Bubble é uma plataforma No-Code que já possui servidor próprio e banco de dados nativo. Por isso, a experiência de criação e aprendizado costuma ser mais rápida.

FlutterFlow não possui servidor ou banco de dados, para tal, é necessário integrar com um backend externo, o que aumenta a curva de aprendizagem e o tempo de criação. Mas, uma das grandes vantagens do FlutterFlow é ter mais independência, te permitindo exportar o código fonte.

Podemos dizer que Bubble é uma ferramenta **full stack**, enquanto FlutterFlow é uma ferramenta **front end**.

No entanto, essa integração externa também oferece muitas possibilidades de escala e customização. A recomendação é: use FlutterFlow se você pretende criar um aplicativo que faça uso de recursos nativos, sendo eles principalmente:

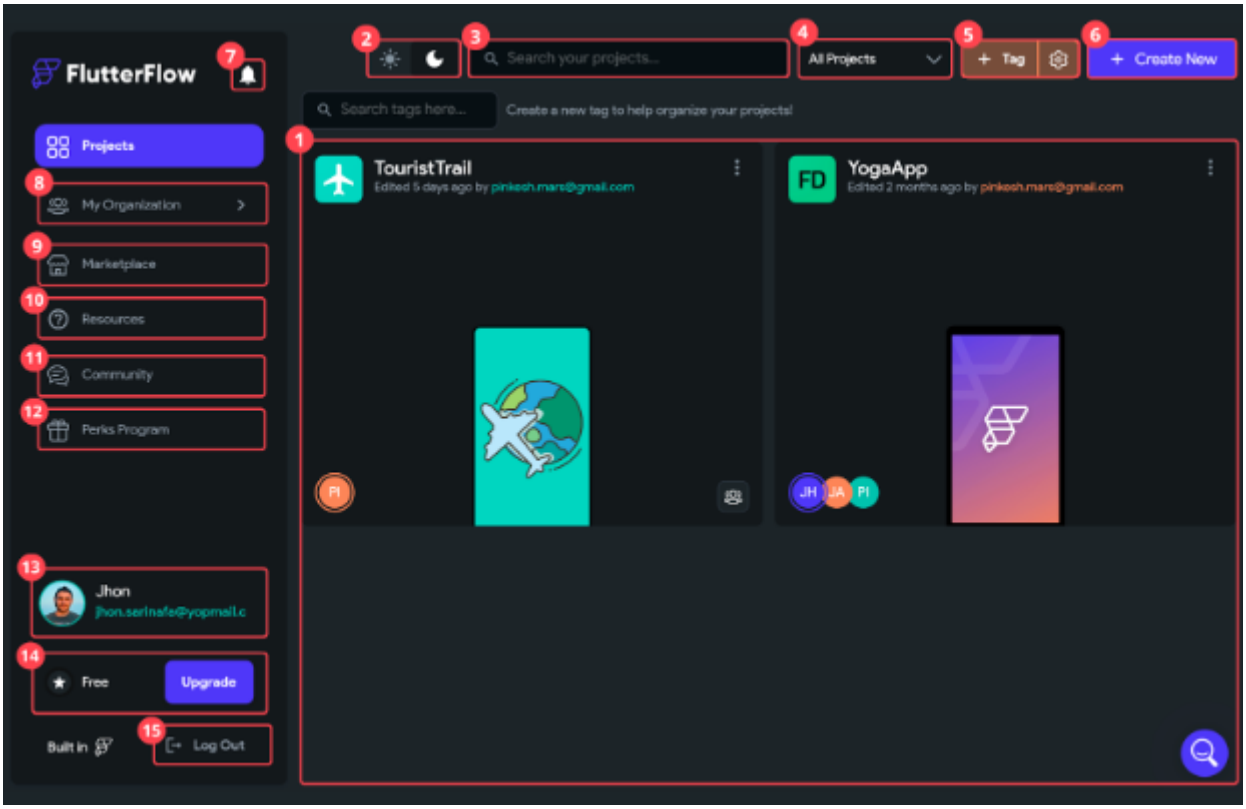
• Armazenamento Offline;
• Geolocalização em Segundo Plano; ou
• Notificações Push Nativas.

Dashboard do FlutterFlow

A primeira coisa que você vê ao entrar no FlutterFlow é o Painel **(Dashboard)**. Ele fornece um local centralizado para gerenciar projetos, como criar novos, pesquisar, excluir e duplicar projetos. Além disso, o painel permite definir o tema desejado, seja escuro ou claro, para facilitar a visualização.

O painel oferece fácil acesso aos recursos organizacionais, permitindo que os membros da equipe colaborem e trabalhem juntos de forma integrada. Além disso, o painel está integrado a um mercado, permitindo que os usuários naveguem e baixem widgets, modelos e plugins.

Você também pode obter links para vários recursos para ajudá-lo a construir aplicativos usando o FlutterFlow. As informações da sua conta e detalhes do plano também são acessíveis a partir desta página.



1 - Projetos: Esta seção exibe os projetos que você criou no FlutterFlow. Você pode navegar facilmente por eles a partir daqui. Use o menu de overflow () para renomear, duplicar, excluir, sair do projeto, adicionar tags e abrir o projeto em uma nova aba do navegador.

Ao duplicar um projeto que contém a configuração do Firebase, é necessário excluir os arquivos de configuração no seu projeto duplicado e concluir uma nova configuração do Firebase para o seu projeto.

2 - Interruptor (Switch) de Modo Escuro/Claro:

O interruptor de Modo Escuro/Claro permite que você escolha entre um esquema de cores claro e escuro para o construtor de aplicativos.

3 - Pesquisa:

Aqui, você pode procurar por seu projeto.

4 - Filtrar projetos:

Você pode usar este menu suspenso para filtrar seus projetos com base nas configurações de privacidade. Você pode filtrar projetos marcados como privados, compartilhados por você ou compartilhados com você.

5 - Etiquetar (tag) projeto:

Aqui, você pode criar e adicionar uma etiqueta aos projetos, proporcionando uma maneira rápida e organizada de classificar e identificar projetos com base em suas características, finalidade ou status.

Para etiquetar um projeto, basta clicar no menu de opções de três pontos no azulejo do projeto, selecionar Adicionar Etiqueta e, em seguida, clicar na etiqueta que deseja definir. Mais tarde, você pode pesquisar e filtrar os projetos com base nas etiquetas.

6 - Criar um novo projeto:

Para criar um novo projeto, você pode usar o botão + Criar Novo. Saiba mais sobre como criar um novo projeto.

7 - Centro de Notificações:

O centro de notificações simplifica a forma como você gerencia comentários e convites em vários projetos. Ele centraliza convenientemente todas as comunicações

do seu projeto. E quando estiver pronto para responder a um comentário, basta tocar nele, e você será levado diretamente para a seção relevante do projeto.

8 - Minha Organização:

Aqui, você pode compartilhar código personalizado, ativos, sistemas de design e APIs entre os usuários da sua equipe e em projetos diferentes.

9 - Mercado:

Melhore seu aplicativo FlutterFlow com facilidade usando o FlutterFlow Marketplace; acesse componentes e modelos pré-criados criados por outros usuários para integração perfeita de novas funcionalidades.

10 - Recursos:

Você encontrará vários links úteis que podem ajudá-lo a criar aplicativos no FlutterFlow clicando no botão Recursos. Tutoriais em vídeo são extremamente úteis para aprender sobre qualquer conceito.

11 - Comunidade:

Isso o redirecionará para nosso Fórum da Comunidade, um lugar onde você pode compartilhar ideias, fazer perguntas e resolver problemas com outros construtores do FlutterFlow. A comunidade compartilha muitas ideias incríveis!

12 - Programa de Benefícios (Perks):

Programa de descontos de empresas como Backendless, Pilot, Intercom e muitas outras, incluindo Google for Startups e Dropbox.

13 - Conta:

Clique aqui para abrir sua página de conta. Isso é útil se você quiser verificar as informações da sua conta, fazer upload de uma foto de perfil, redefinir sua senha, ver suas indicações ou excluir sua conta.

14 - Informações do Plano:

Aqui, você vê o nome do seu plano atual e também pode fazer upgrade para acessar benefícios adicionais.

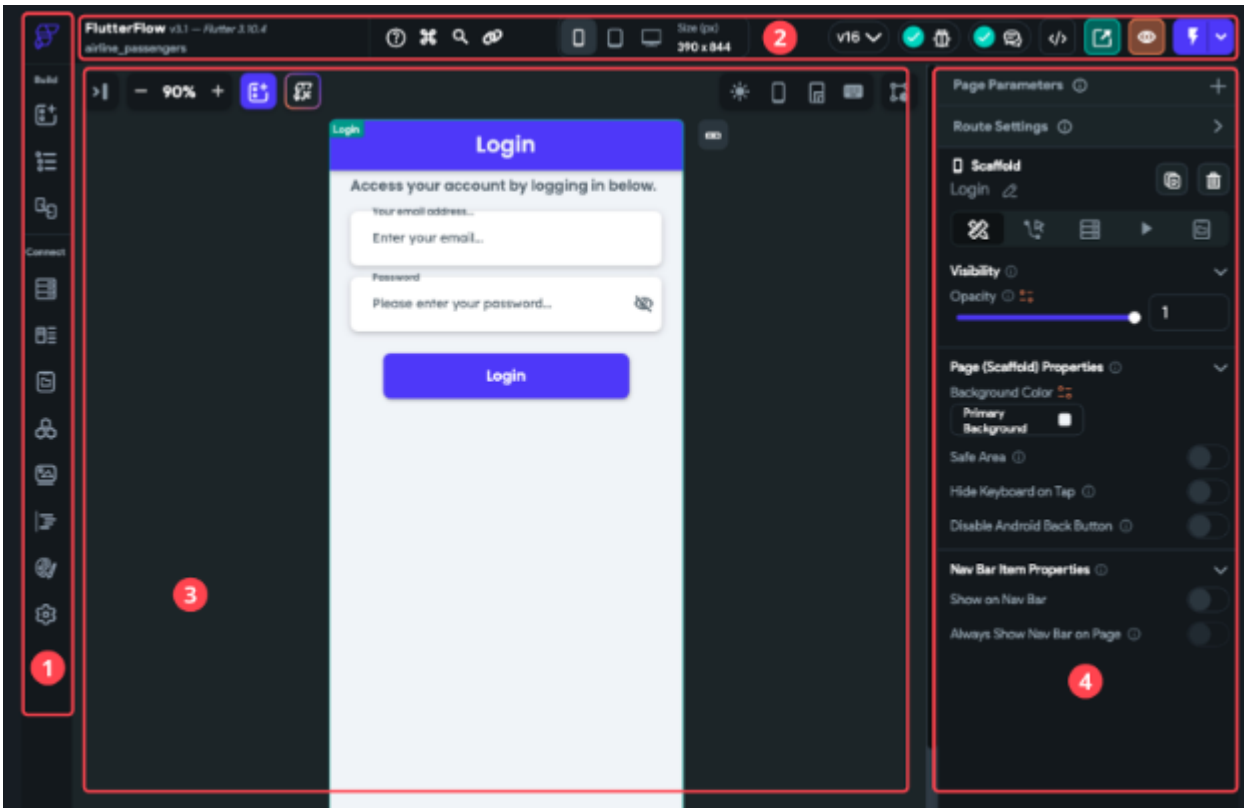
15 - Sair (Logout):

Você pode usar este botão para fazer logout com segurança da sua conta FlutterFlow.

Com essa visão geral do Painel, podemos ir para o **App Builder**, onde iremos trabalhar nosso aplicativo.

App Builder

Ao abrir o projeto, você verá o App Builder composto por quatro seções principais: 1 - Menu de Navegação, 2 - Barra de Ferramentas, 3 - Tela **(Canvas)**, 4 - Painel de Propriedades.



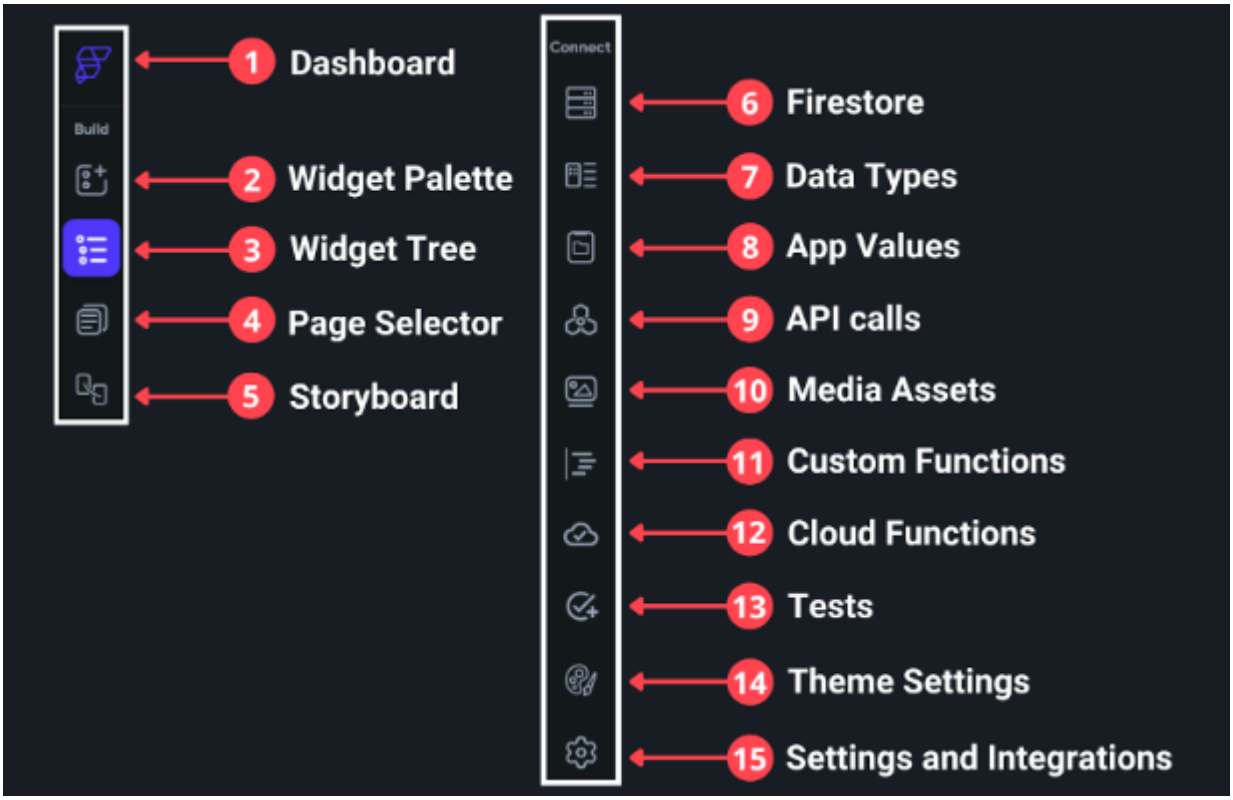
Menu de Navegação

O menu mais à esquerda na página do seu projeto FlutterFlow é chamado de Menu de Navegação **(Navigation Menu)**. Ele fornece acesso ao construtor de interface do usuário do projeto, à árvore de widgets e ao storyboard. Ele ajuda na navegação entre páginas para adicionar recursos ao projeto, incluindo o banco de dados Firestore, configuração de chamadas de API, upload de ativos de imagem, adição de integrações e muito mais.

Alguns conceitos importantes no FlutterFlow: **Widget** e **Storyboard**.

Um **widget**, numa interface gráfica, é um elemento de interação - tal como janelas, botões, menus, ícones, barras de rolagem etc. O termo pode também se referir aos pequenos aplicativos que flutuam pela área de trabalho e fornecem funcionalidades específicas ao utilizador

O **Storyboard** permite que você visualize o design geral e a navegação do seu aplicativo. No Storyboard, você pode ver diferentes telas e interações do usuário que compõem seu aplicativo. Isso permite que você veja como os usuários navegarão pelo seu aplicativo e garanta que a experiência do usuário seja o mais intuitiva e amigável possível. Os Storyboards também podem ajudar a identificar e resolver possíveis problemas de navegação ou becos sem saída. Isso economiza tempo e recursos que, de outra forma, seriam gastos na correção de problemas posteriormente.



1. Dashboard: Aqui, você pode acessar o painel onde gerencia seus projetos (criar novos projetos, remover projetos existentes, duplicar e sair de um projeto). Você também pode acessar as informações da sua conta e links para vários recursos do FlutterFlow.

2. Widget Palette: A Widget Palette fornece acesso a todos os widgets que podem ser adicionados à aplicação.

3. Widget Tree: O painel Widget Tree oferece uma visão geral de alto nível de todos os widgets que você adicionou a uma página selecionada.

4. Seletor de Página: Daqui, você pode gerenciar todas as páginas e componentes do seu aplicativo. Você também pode criar uma estrutura de pastas personalizada para atender às necessidades específicas do seu projeto; por exemplo, organizando páginas por recurso. Basta clicar com o botão direito na pasta e você verá a opção de adicionar uma nova pasta, página ou componente.

5. Storyboard: O storyboard ajuda a visualizar o design e a navegação de todo o seu aplicativo em uma única interface.

6. Firestore: Na página do Firestore, você pode criar coleções de banco de dados usando o Firebase. Você pode aprender mais sobre o Firebase no FlutterFlow aqui.

7. Tipos de Dados: Daqui, você pode criar tipos de dados personalizados que podem ser usados em todo o seu aplicativo, como no App State, Firestore, código personalizado e muito mais!

8. Valores do App: Daqui, você pode gerenciar suas variáveis de App State e Constantes.

9. Chamadas de API: Daqui, você pode definir chamadas de API.

10. Ativos de Mídia: Daqui, você pode fazer upload de quaisquer ativos que deseja usar em seu aplicativo. Isso também mostra os ativos de mídia da equipe.

11. Funções Personalizadas: Daqui, você pode adicionar funcionalidades personalizadas ao seu aplicativo, definindo Funções Personalizadas, Widget Personalizado e Ações Personalizadas.

12. Funções em Nuvem: Você pode escrever e implantar Funções em Nuvem para o seu projeto Firebase diretamente no FlutterFlow.

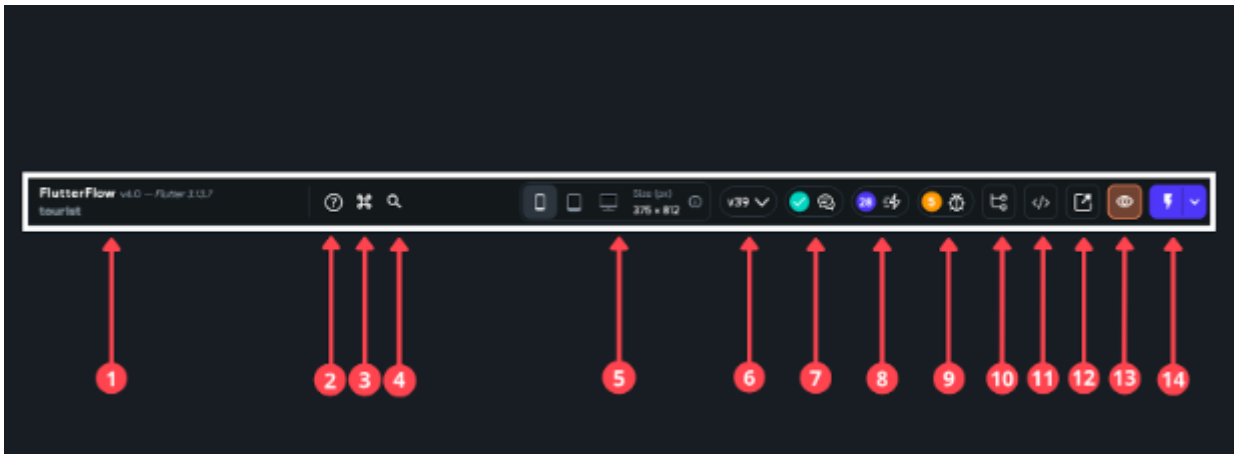
13. Testes: Daqui, você pode adicionar testes automatizados para o seu projeto.

14. Configurações de Tema: As configurações de tema permitem personalizar a aparência visual do seu aplicativo para corresponder a um design ou marca específicos.

15. Configurações e Integrações: Nesta página, você pode acessar configurações relacionadas ao aplicativo, configurações gerais, configurações do projeto, compras e assinaturas no aplicativo, e integrações.

Barra de Ferramentas (Tool bar)

A barra de ferramentas está localizada no topo do construtor de aplicativos. Ela consiste em vários botões para configurar seu projeto, salvar a versão do seu aplicativo, obter ajuda, relatar/depurar problemas, ver comentários do projeto, baixar o código do seu aplicativo e executar seu aplicativo diretamente no FlutterFlow.



1. Informações do Projeto: Ao passar o mouse sobre esta seção, você verá as informações essenciais sobre o seu projeto. Aqui, você pode verificar facilmente o nome do projeto, a versão atual do FlutterFlow que você está usando, juntamente com sua data de lançamento. Além disso, mostra a versão do Flutter na qual seu projeto está atualmente sendo executado.

2. Ajuda: Ao clicar no botão Ajuda, você terá acesso a links de recursos essenciais que serão úteis durante a construção de aplicativos.

3. Atalhos do teclado: Com atalhos do teclado, você pode realizar ações comuns relacionadas a widgets e executar seu projeto nos modos de teste ou execução com apenas alguns toques, economizando tempo e esforço.

4. Paleta de Comando: A Paleta de Comandos facilita a localização e o uso de elementos no Construtor de Aplicativos. Clique uma vez para ver todas as páginas, componentes, widgets, ações, chamadas de API, variáveis e muito mais. Abra a Paleta de Comandos clicando no botão de pesquisa ou pressionando Cmd/Ctrl + K. Pesquise por qualquer coisa e, para descobrir onde ela é usada, clique na seta

roxa - mostraremos todos os lugares. Para acesso direto à página com o item, clique no próprio item.

5. Tamanho do Canvas: É permitido visualizar e projetar seu aplicativo para vários dispositivos móveis, tablets e desktops. O tamanho do canvas é definido como largura (em pixels) x altura (em pixels). Este tamanho é determinado de acordo com o dispositivo selecionado. Para definir um tamanho personalizado do canvas, clique em Tamanho do Canvas e insira a largura e altura (em pixels) desejadas.

6. Histórico do Projeto: As versões permitem que você salve estados específicos nomeados do seu projeto. Essa função é particularmente útil para fins de recuperação. Se você excluir acidentalmente uma página ou componente ou alterar configurações, pode facilmente reverter para uma versão salva, restaurando seu projeto ao seu estado anterior. As capturas (snapshots) são salvas automaticamente enquanto você constrói.

7. Comentários do Projeto: Isso ajuda você a deixar seus pensamentos, opiniões, perguntas ou feedback sobre um widget específico com sua equipe de projeto ou um cliente. Ao adicionar um comentário, você pode marcar usuários, e eles poderão responder, criando uma sequência de conversa.

8. Otimizações e Aprimoramentos: FlutterFlow analisa e sugere aprimoramentos para elevar o design e a velocidade do seu aplicativo. Imagine ter um consultor pessoal para design e desempenho dedicado a melhorar a experiência do usuário do seu aplicativo. Otimizações: FlutterFlow destaca elementos que podem diminuir a velocidade do seu aplicativo. Por exemplo, uma consulta em uma coluna, uma consulta de backend não utilizada e duplicada, um widget com tamanho ilimitado, etc. Aprimoramentos de UI: FlutterFlow oferece dicas para um design mais atraente visualmente e amigável ao usuário, como redimensionar um alvo de toque pequeno de qualquer widget.

9. Problemas do Projeto: Se houver problemas ou avisos presentes no seu projeto

atual que possam resultar em falha na construção ou travamento do aplicativo, eles serão exibidos nesta seção. Você pode clicar neste botão para visualizar uma breve descrição de cada problema e navegar até o local onde pode corrigir o problema clicando no respectivo aviso.

10. Branching: A ramificação permite que você crie uma cópia separada do seu projeto existente para construir novos elementos sem comprometer recursos existentes.

11. Menu do Desenvolvedor: Você pode visualizar o código do seu projeto, conectar-se ao GitHub e também baixar o código-fonte do seu projeto.

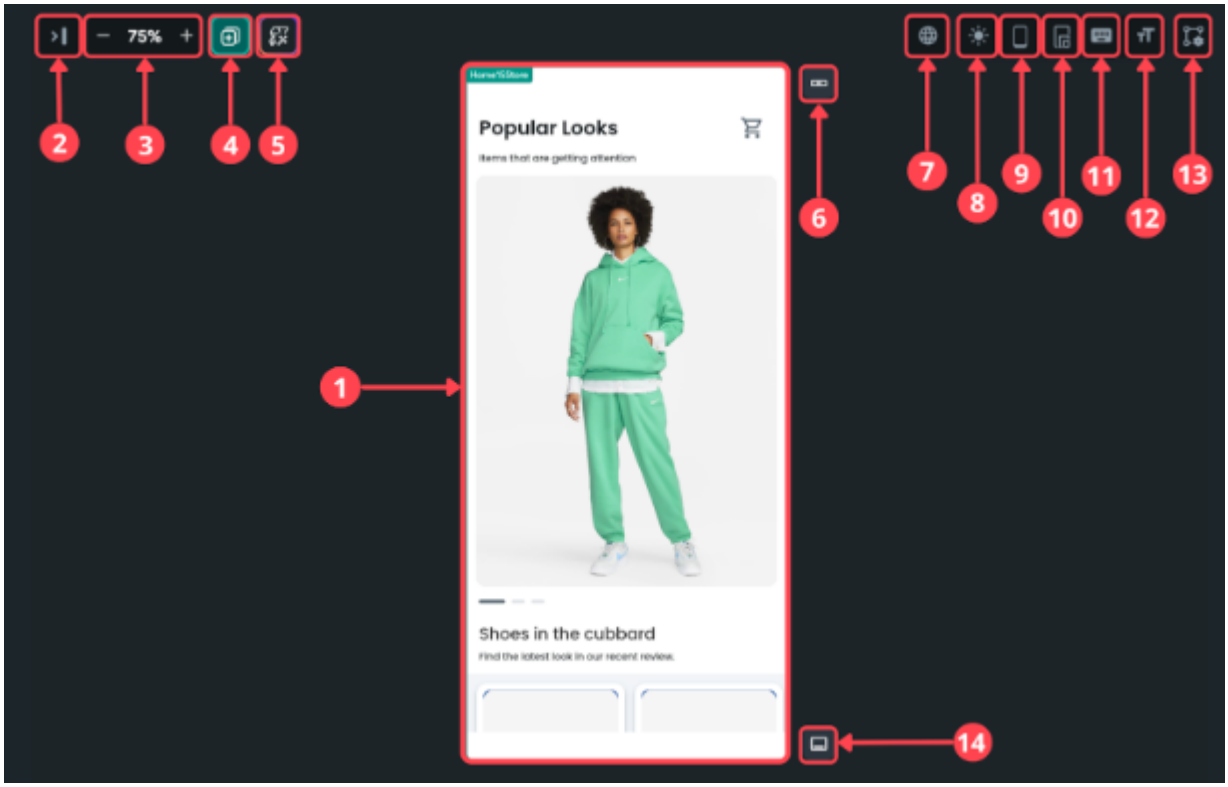
12. Compartilhar projeto: Você pode tornar um projeto público para que outras pessoas possam visualizar e clonar. Antes de compartilhar seu projeto, certifique-se de remover qualquer informação sensível.

13. Pré-visualização (Preview): Você pode usar o modo de pré-visualização para experimentar rapidamente seu aplicativo em um dispositivo virtual sem precisar esperar pelo processo de construção. Isso ajuda a ter uma boa visão da navegação e animações que você adicionou ao seu aplicativo. Você também pode visualizar seu aplicativo nos modos Claro/Escuro e visualizá-lo em vários dispositivos móveis, tablets e desktops. Este modo tem algumas limitações; você não pode usar o Firebase ou realizar chamadas de API no modo de pré-visualização. No entanto, as ações de toque duplo, pressão longa e envio funcionam neste modo.

14. Testes: Este menu permite que você execute seu aplicativo no modo de teste ou execução

Tela (Canvas)

A Área de Canvas exibe a tela do dispositivo móvel, onde você pode adicionar widgets arrastando e soltando-os no canvas. Ela possui botões para ajustar o nível de zoom e o tamanho do canvas, além de visualizar seu aplicativo nos modos claro/escuro. Também é possível visualizar como seu aplicativo ficará quando o teclado estiver aberto. Quando a funcionalidade de multi-idioma está habilitada, você pode ver um menu suspenso para alterar o idioma e visualizar como seu aplicativo aparece em diferentes idiomas.



1. Tela de Pré-visualização: Este é o canvas da tela do dispositivo onde você pode construir a interface do usuário. Você pode personalizar a tela adicionando widgets usando o recurso de arrastar e soltar do Widget Palette e aplicando propriedades presentes no Painel de Propriedades.

2. Mostrar ou ocultar o Menu de Navegação: Usando este botão, você pode abrir ou fechar o Menu de Navegação.

3. Controles de Zoom: Há um botão de zoom in (+) e um botão de zoom out (-) para controlar o nível de zoom do canvas. Ao trabalhar em designs de interface do

usuário complexos, isso é útil para dar uma melhor prévia dos widgets, ampliando para uma área específica e depois dando uma visão geral ao ampliar.

4. Adicionar página, componente ou fluxos: Se preferir manter a árvore de widgets sempre fechada, você pode usar este menu para adicionar rapidamente uma nova página, componente ou fluxo.

5. Página Gerada por IA: A partir daqui, você pode criar uma página usando IA.

6. Adicionar AppBar: Este botão permite adicionar o AppBar à sua página. Clicar neste botão abre o pop-up que mostra diferentes estilos de AppBar que você pode adicionar à sua página. Basta clicar em qualquer estilo de AppBar da lista, e a AppBar será exibida dentro da Tela de Pré-visualização.

7. Pré-visualização em Vários Idiomas: Você pode pré-visualizar todo o texto do seu aplicativo em diferentes idiomas no canvas usando o menu suspenso de vários idiomas. O menu suspenso mostra a lista de idiomas suportados com a bandeira do país, e ao alterar o idioma, o texto traduzido aparece no canvas. Isso é muito útil para testar o design do seu aplicativo com texto traduzido sem precisar executar seu aplicativo. Esta opção está disponível apenas se você tiver habilitado o suporte a vários idiomas para o seu projeto.

8. Modo Claro/Escuro: Você pode usar o ícone de alternância de Modo Claro/Escuro para pré-visualizar o tema Claro ou Escuro do seu aplicativo. Mudar para o modo Escuro ou Claro mostrará os widgets e componentes do seu aplicativo conforme as cores definidas no tema de modo escuro ou claro. Esta opção está disponível apenas se você tiver habilitado o suporte ao tema Dark Mode para o seu projeto.

9. Exibir Dispositivo: Você pode usar isso para mostrar o frame do dispositivo na pré-visualização. Isso pode ser útil se você quiser visualizar melhor ou capturar a interface do usuário em um dispositivo específico.

10. Exibir Barras de Redimensionamento: Habilitar as Barras de Redimensionamento mostrará as barras de manipulação nos lados direito e inferior da tela de pré-visualização. Isso permite alterar o tamanho da tela de pré-visualização para testar melhor seu aplicativo em uma tela com uma resolução personalizada e a responsividade geral do seu aplicativo.

11. Exibir Teclado: Habilitar esta opção abre o teclado na tela de pré-visualização do dispositivo; isso ajuda a visualizar a interface do usuário com o teclado exibido. Abrir o teclado geralmente empurra o conteúdo da tela para cima.

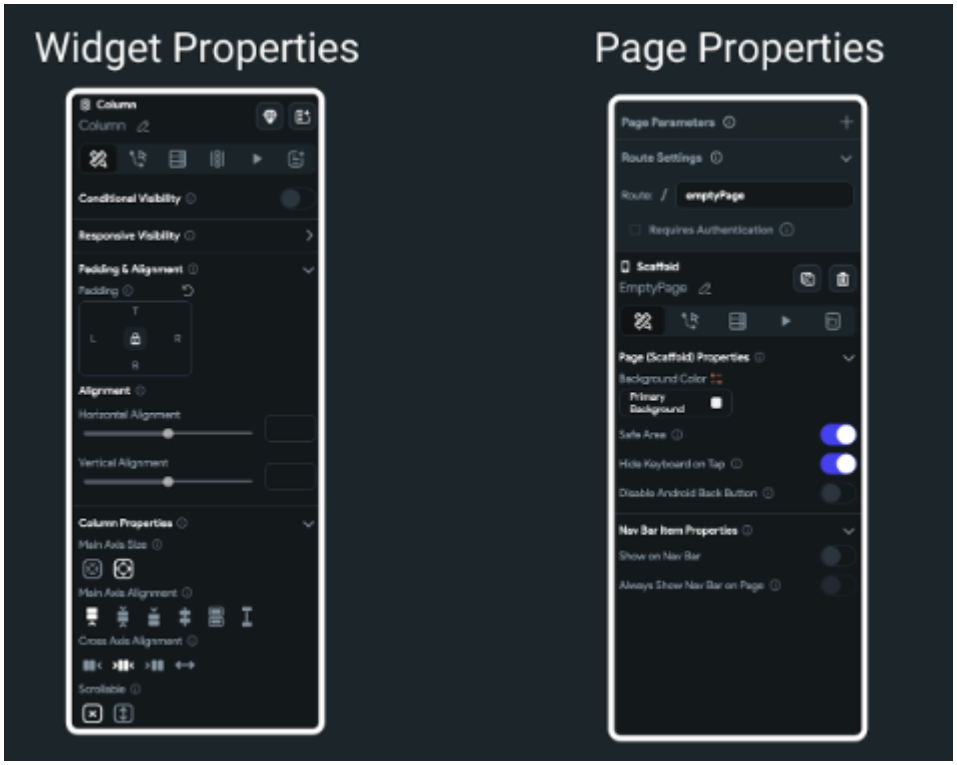
12. Visualização de Fonte Maior: Isso permite simular como seu aplicativo aparece quando os usuários ajustam a escala de texto ou configurações de acessibilidade em seus dispositivos. É vital para desenvolver um aplicativo visualmente atraente e também acessível, especialmente para aqueles que necessitam de tamanhos de texto maiores.

13. Configurações do Canvas: Área Segura - Quando a área segura está habilitada, ela garante que o conteúdo não seja ocultado por recursos físicos como recortes de câmera, sensores ou bordas arredondadas da tela. Você pode habilitar ou desabilitar usando o interruptor de Área Segura. Redimensionar com Snap - Ao redimensionar um widget, você pode ajustar seu tamanho para valores específicos de pixels ou uma porcentagem do tamanho da tela. Por padrão, é definido como 1 px. Para alterar isso, clique aqui, certifique-se de que o Redimensionamento com Snap está habilitado e insira o valor em pixels ou porcentagem.

14. Adicionar NavBar: Este botão permite adicionar a NavBar à sua página. Clicar neste botão abre o pop-up que permite habilitar a NavBar para o seu projeto. Depois que a NavBar é habilitada, você pode personalizá-la para combinar com o seu design.

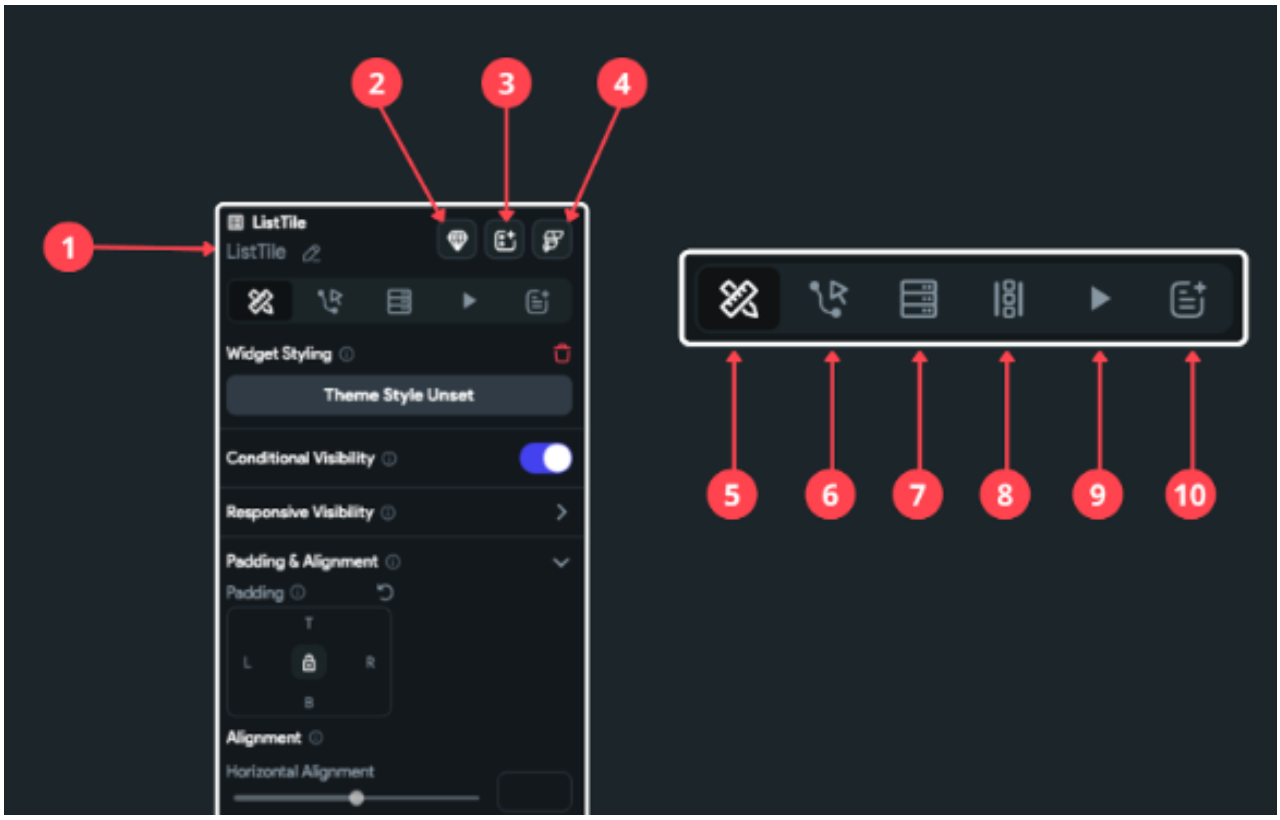
Painel de Propriedades

O Painel de Propriedades permite que você personalize a aparência visual e o comportamento do widget e da página. Você também pode acessar outras guias para especificar Ações (por exemplo, onTap e onSubmit), configurar Consultas de Backend (para qualquer chamada de API ou coleção Firebase) e adicionar Documentação e Animações ao widget selecionado. Além disso, quando a página está selecionada, você pode ver uma opção para adicionar uma variável de estado da página.



Propriedades de Widgets

A partir daqui, você pode personalizar as propriedades para o widget selecionado, o que pode variar com base no tipo de widget.



1. Nome do Widget: Quando você seleciona qualquer widget, seu nome aparece no painel de propriedades. O nome padrão para qualquer widget é o seu tipo. Por exemplo, se você selecionar o widget Container, o nome aparecerá como 'Container'. No entanto, você pode usar o ícone de edição à direita para alterar seu nome.

2. Criar um componente: Este botão permite que você crie um componente a partir do widget selecionado.

3. Salvar como modelo: Este botão permite que você salve o widget selecionado como um widget de modelo.

4. Salvar como widget de tema: Este botão permite que você salve o widget selecionado como um widget de tema.

5. Propriedades: Na guia Propriedades, você encontrará as propriedades do widget

que pode modificar. Quando nenhum widget está selecionado, este painel exibe as propriedades da página (ou seja, as propriedades do Scaffold). As propriedades disponíveis variarão de acordo com o tipo de widget. Para Elementos de Página, a Visibilidade é a única propriedade comum; outras propriedades variam de acordo com o widget selecionado. Para Elementos Básicos, Elementos de Layout e Elementos de Formulário, Visibilidade, Preenchimento e Propriedades de Alinhamento são comuns.

6. Ações: As Ações permitem que você defina o que deve acontecer quando um usuário interage com os elementos/widgets do aplicativo, como tocar em um botão. As ações diferem de acordo com o widget selecionado (em alguns widgets, você não pode aplicar nenhuma ação).

7. Consulta de Backend: Você pode usar isso para definir uma consulta que será acionada assim que o widget aparecer na tela. Quando os dados são carregados, eles podem ser acessados por todos os widgets presentes neste widget.

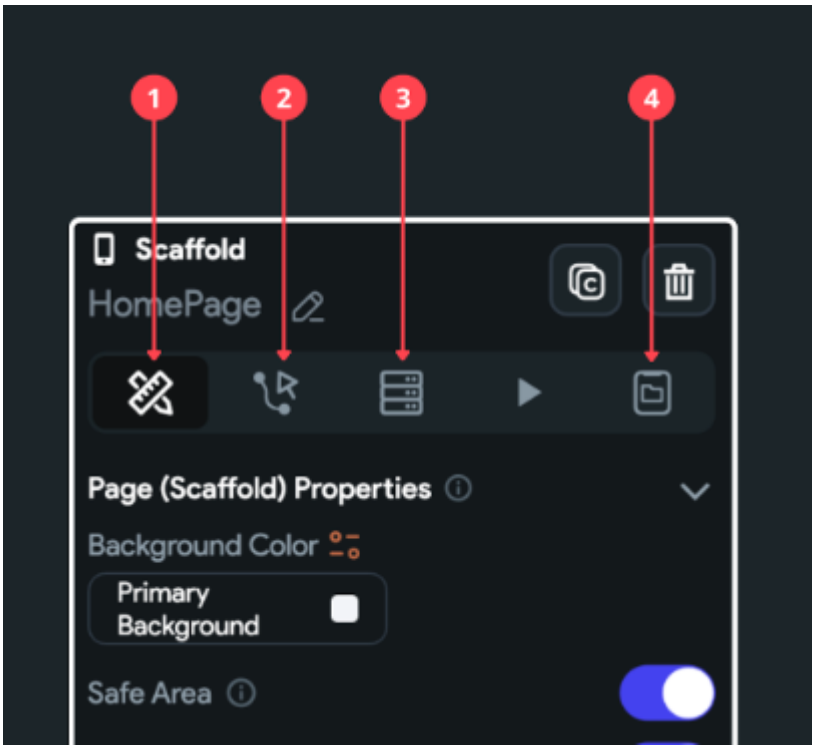
8. Gerar Filhos Dinâmicos: Os widgets que podem lidar com vários widgets filhos têm uma guia adicional chamada Gerar Filhos Dinâmicos. Isso ajuda a gerar vários widgets filhos a partir de uma variável de lista. Isso é geralmente útil quando você está recuperando dados de uma chamada de API. Alguns dos widgets que podem lidar com vários filhos são Column, Row, Stack, ListView, GridView e PageView. Usando esta guia, você pode selecionar o Nome da Variável no qual armazenar os valores (para que possa ser facilmente referenciado em qualquer outro widget), Máximo de Itens e a Fonte a ser usada para obter os valores.

9. Animações: Você pode aplicar animações a um widget. Saiba mais sobre como adicionar animações aqui.

10. Documentação e Rótulos Semânticos: A partir daqui, você pode adicionar documentação e rótulos semânticos para seus widgets.

Propriedades da Página

A partir daqui, você pode personalizar as propriedades para a página selecionada.



1 - Propriedades da Página: Usando as propriedades da página, você pode alterar o comportamento e a aparência da página atual. Aqui estão elas:

• Visibilidade: Use isso para controlar a opacidade de uma página.

• **Cor de Fundo:** Use isso para alterar a cor de fundo da página.

• **Área Segura:** Isso é usado para ajustar automaticamente o layout para que não sobreponha áreas específicas do sistema, como entalhes, barras de status ou cantos arredondados específicos do dispositivo. Essa propriedade é ativada por padrão para garantir que o conteúdo do seu aplicativo seja visível e acessível sem ser obscurecido por esses elementos do sistema.

• **Ocultar Teclado ao Toque:** Essa opção é ativada por padrão para implementar um comportamento comum em aplicativos móveis, onde o teclado virtual é automaticamente fechado quando o usuário toca fora de qualquer widget de campo de texto. Essa funcionalidade melhora a experiência do usuário, permitindo que eles fechem facilmente o teclado e interajam com outros elementos do aplicativo.

2 - Ações: Você pode adicionar os seguintes tipos de ações na página selecionada.

• **No Carregamento da Página:** As ações adicionadas aqui são acionadas assim que a página se torna visível para os usuários. Isso é útil quando você deseja realizar tarefas específicas quando a página é carregada, como fazer uma chamada de API para buscar dados que precisam ser exibidos na página.

• **No Agitar do Telefone:** As ações adicionadas aqui são acionadas quando o usuário agita o telefone. Isso é útil quando você deseja realizar tarefas específicas ou acionar ações específicas em resposta a um gesto de agitar o telefone. Por exemplo, você pode usar isso para construir funcionalidades como:

• Randomização de conteúdo: Agite o telefone para gerar um número aleatório, exibir uma citação aleatória ou alterar a imagem de fundo.

• Atualização de dados: Agite o telefone para acionar uma atualização de dados, como buscar as últimas notícias ou atualizar um feed ao vivo.

• Redefinir o estado do aplicativo: Agite o telefone para redefinir o estado do aplicativo, limpar os campos do formulário ou retornar à tela inicial do aplicativo.

3 - Consulta ao Backend: Você pode usar isso para definir uma consulta que será acionada quando a tela estiver visível. Todos os widgets nesta página podem acessar o resultado da consulta ao backend.

4 - Variável de Estado da Página: Você pode criar e utilizar variáveis específicas para a página selecionada, que não são compartilhadas com outras páginas do seu aplicativo. Saiba mais sobre adicionar variáveis de estado da página.

Estudo de caso: Lista de Supermercado

Para que possamos fazer uma boa comparação entre Bubble e FlutterFlow, vamos montar nosso mesmo aplicativo na nova ferramenta. Lembrando do escopo do nosso projeto:

- Um campo de formulário para preencher com o nome do produto e um botão adicionar;
- Uma lista com todos os produtos já adicionados;
- Uma função para quando clicar no item, vai marcar como comprado;
- Uma função para quando clicar no item já comprado, desmarcá-lo.

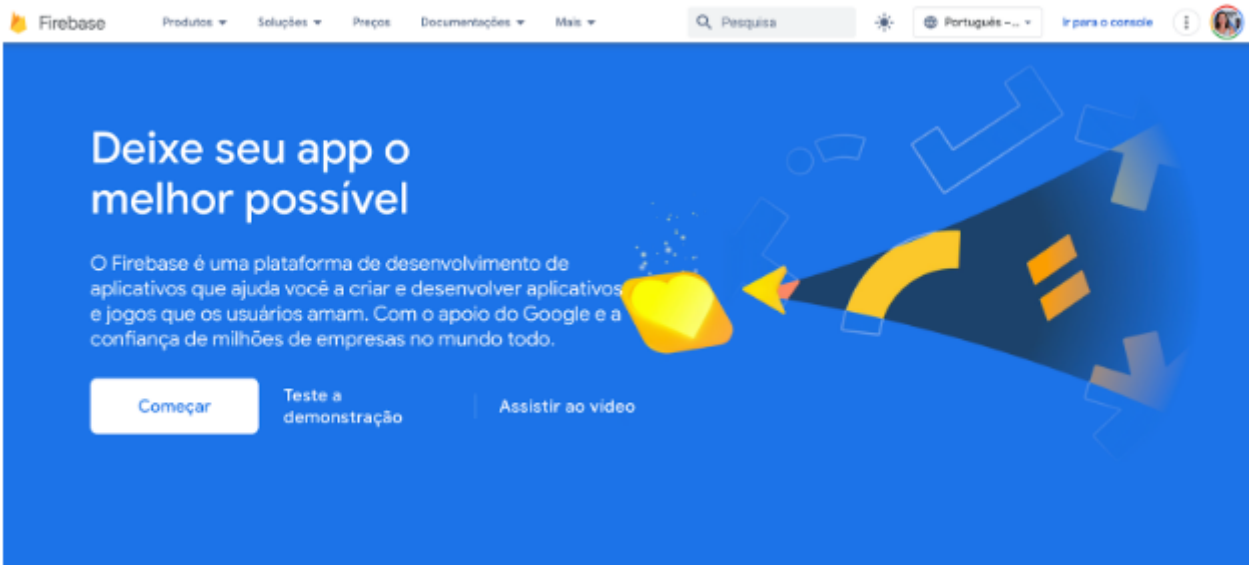
Como já ganhamos uma boa expertise montando nosso MVP com o Bubble, já vamos adicionar a tela de edição ao escopo do projeto.

Como comentado, o FlutterFlow é apenas ferramenta **Front End**, logo vamos precisar nos conectar com **Back End** externo. Aqui usaremos uma ferramenta da própria Google: **Firebase**.

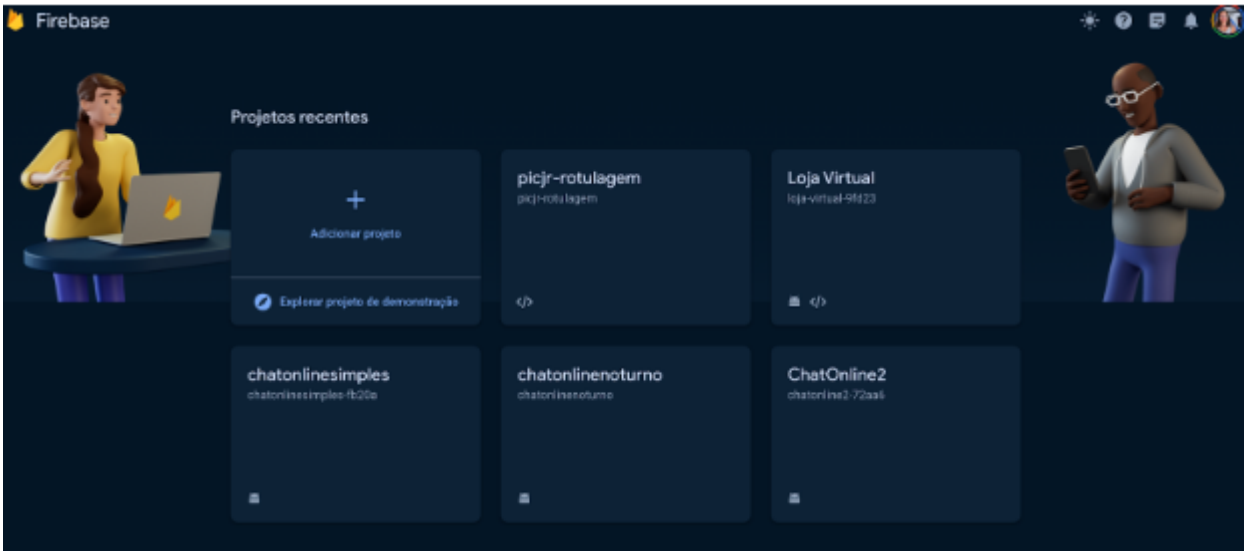
Firebase

Firebase é uma plataforma que ajuda no desenvolvimento, medição, aprimoramento e crescimento de aplicativos móveis. É respaldada pelo Google e abrange uma ampla gama de serviços, incluindo banco de dados em tempo real, autenticação, monitoramento de falhas, análises, notificações push e muito mais. O Firebase oferece todas as ferramentas relacionadas ao backend e à plataforma como serviço, permitindo que você se concentre mais na construção das principais funcionalidades do aplicativo.

Acesse https://firebase.google.com/ e crie sua conta gratuita com seu acesso Gmail. Use o mesmo email que usou para a conta do FlutterFlow.

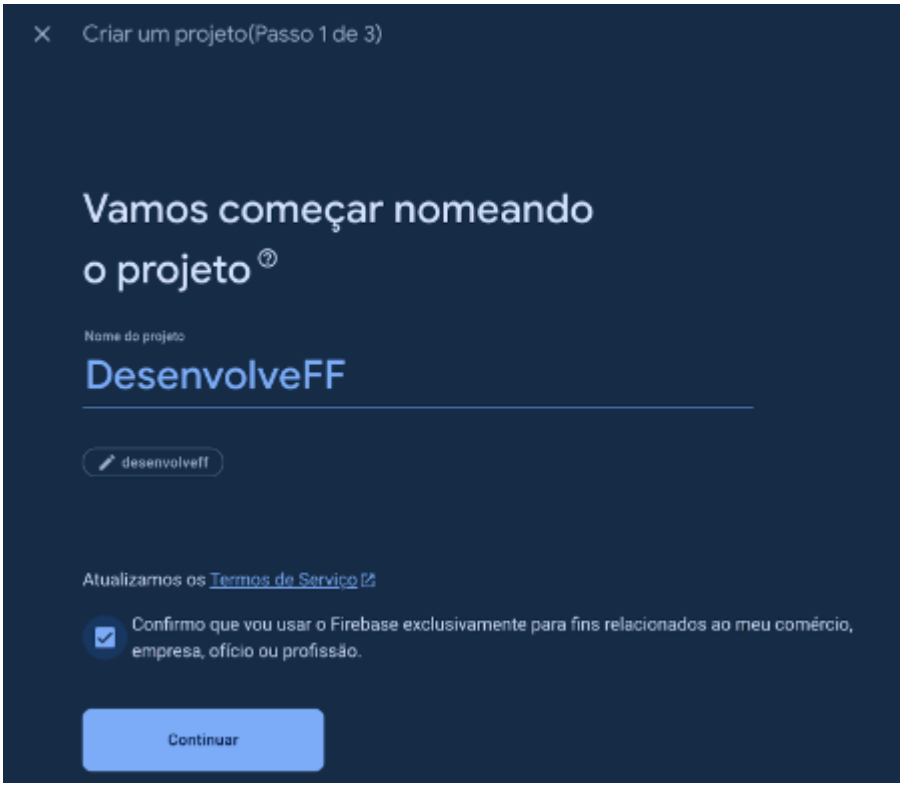


Agora vamos criar um novo projeto. Clique em Começar ou em Criar um novo projeto. Se o botão for Começar, abriremos o Dashboard do Firebase.

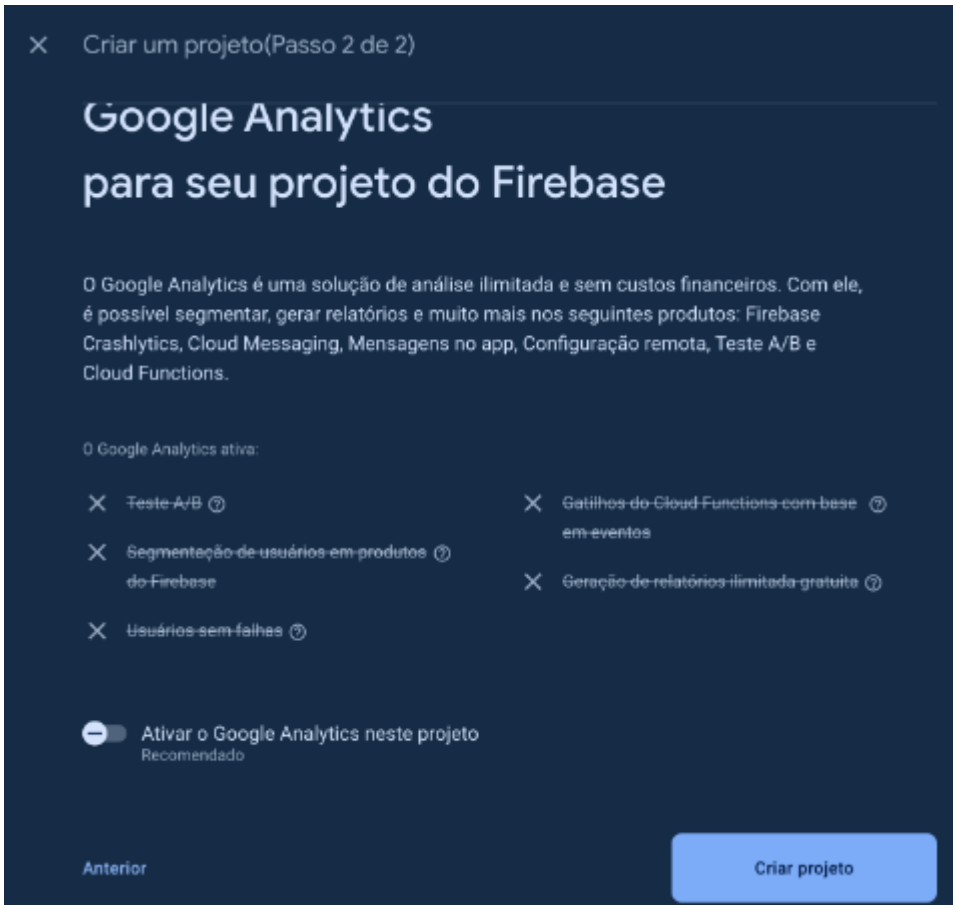


O meu Firebase já tem bastante coisa, mas o de vocês vai aparecer com apenas **adicionar projeto.**

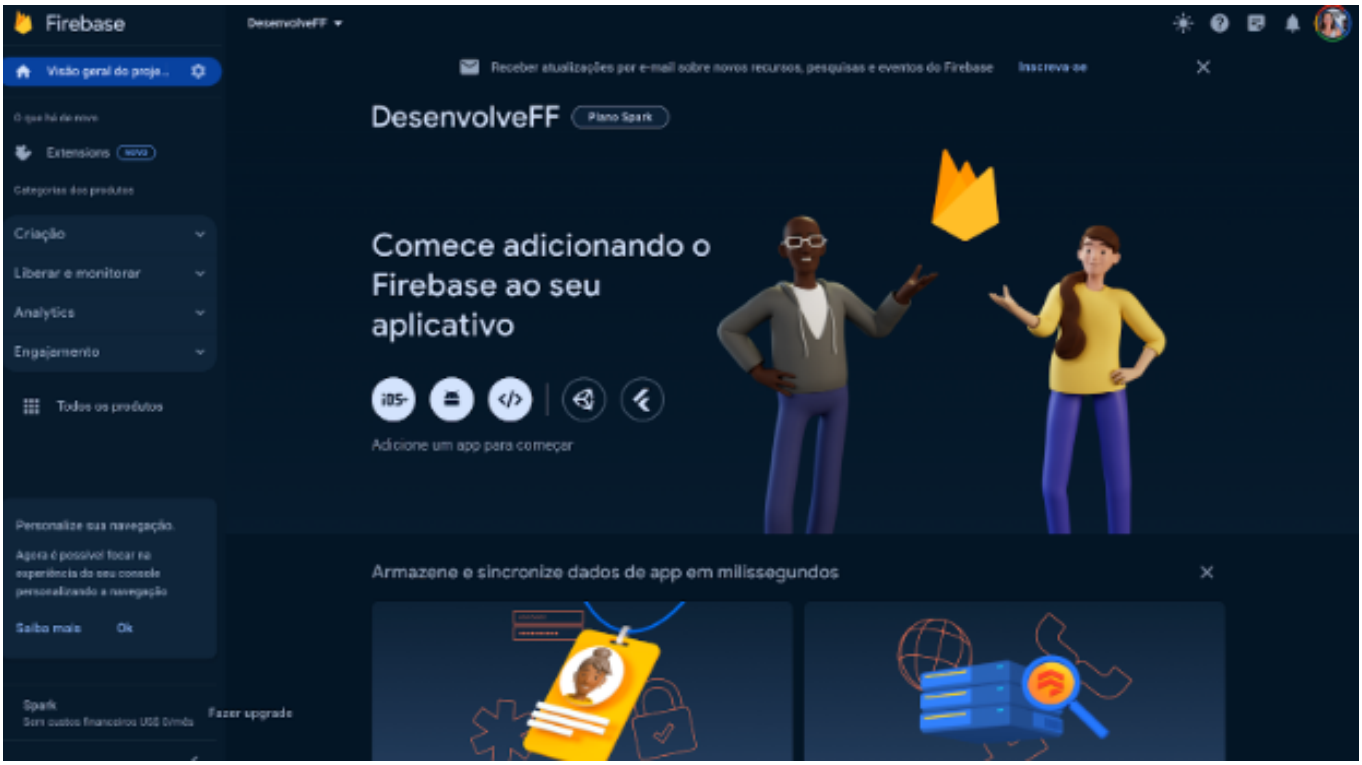
Se o botão for Criar Projeto, já vamos para a tela de criação de projetos. Escolha um nome para seu projeto e clique em Continuar.



Desative o Analytics e clique em Criar Projeto.
Essa é uma ferramenta mais avançada que não vamos utilizar agora.



Agora chegamos no Dashboard do seu projeto. Neste curso, não nos aprofundaremos nos detalhes do Firebase, somente utilizaremos uma parte dele. Trabalharemos mais com ele ao longo do Projeto Desenvolve.

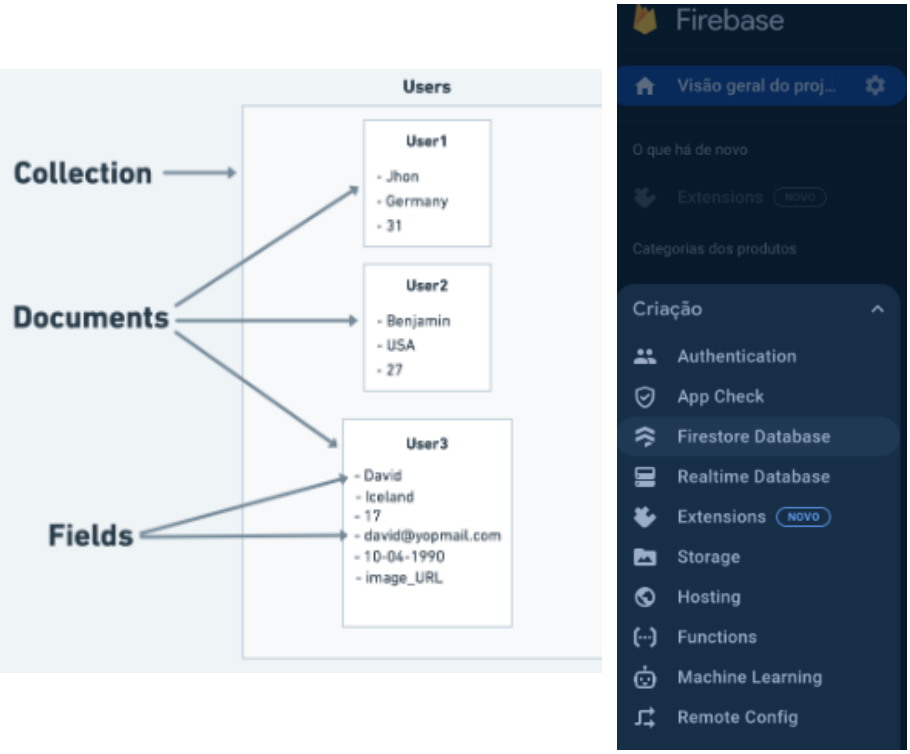


Vamos então criar nosso banco de dados. E aqui vem uma coisa bem interessante e diferente. Lembra que chamamos de Tabela o banco de dados do Bubble? Isso é porque ele é um banco de dados relacional, ou seja, é um banco de dados que modela os dados de uma forma que eles sejam percebidos pelo usuário como tabelas, ou mais formalmente relações. Estes bancos só armazenam dados que chamamos de estruturados, dados que conseguimos colocar em tabelas.

Já o Firestore Database é um banco que chamamos de Não Relacional ou NoSQL. Bancos de dados não relacionais são mais flexíveis e úteis para dados com requisitos variáveis. Você pode usá-los para armazenar imagens, vídeos, documentos e outros conteúdos semiestruturados e não estruturados.

Enquanto em bancos relacionais chamamos de Tabelas e Campos, aqui teremos outras nomenclaturas.

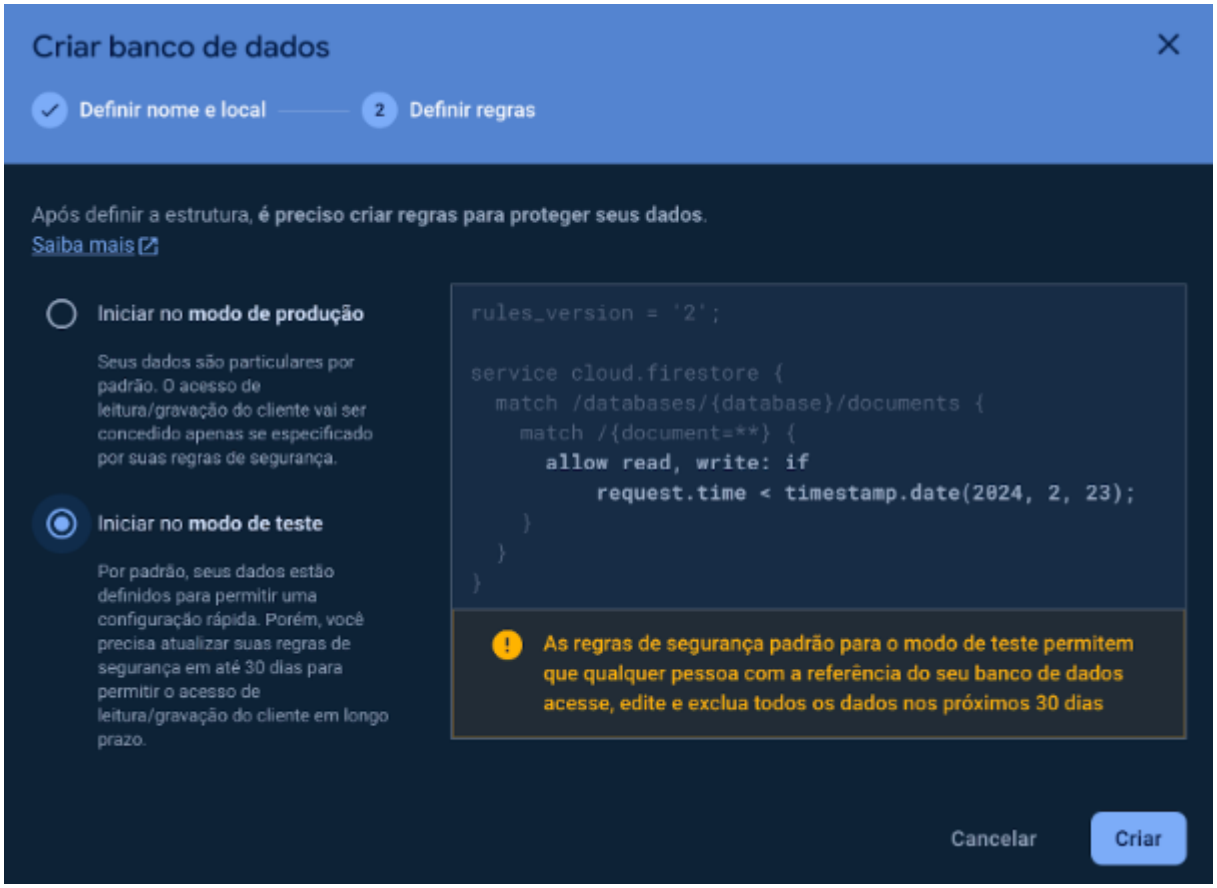
• Coleção **(Collection)**: Uma coleção é simplesmente um conjunto de 'documentos'.
• Documento **(Documents)**: Um documento é um registro que contém os 'campos'.
• Campos **(Fields)**: Os pares chave-valor dentro do documento são chamados de 'campos', por exemplo, nome, local, idade, etc.



No menu da esquerda clique em Criação > Firestore Database.

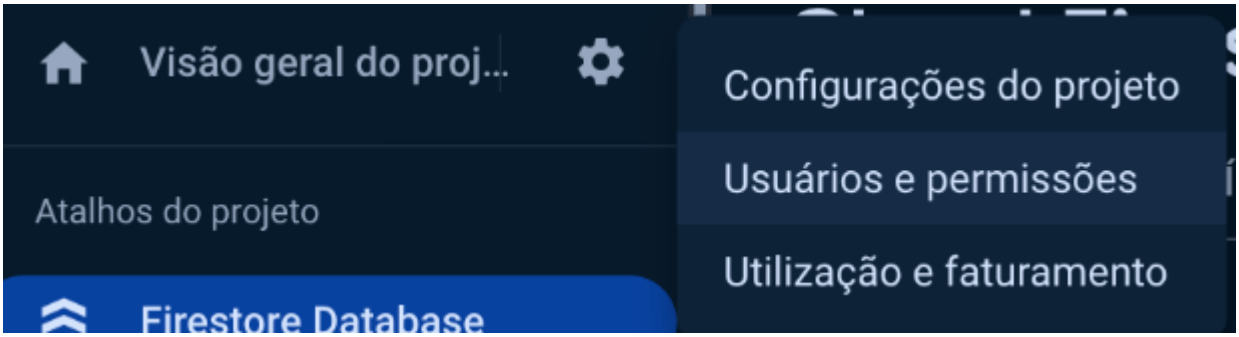
Você será levado para **Cloud Firestore**. Clique em Criar Banco de Dados.

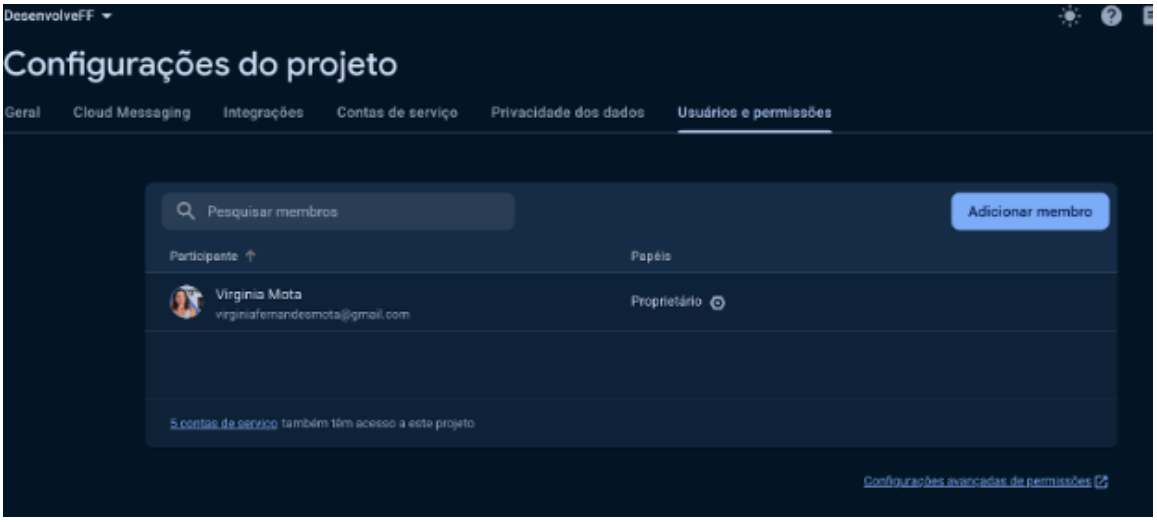
Nessa primeira tela selecionamos o serviço de nuvem mais próximo da nossa localidade.



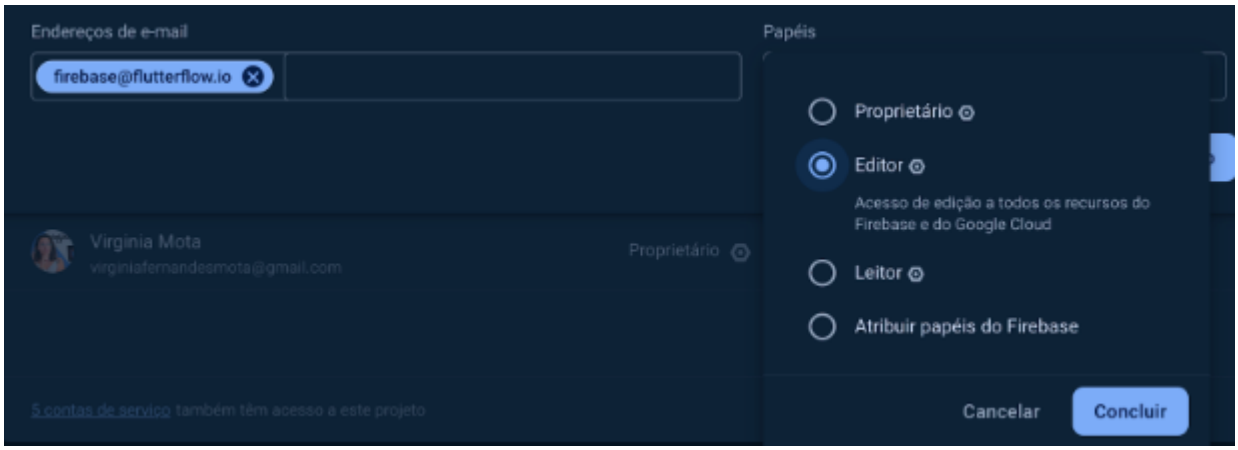
Nessa tela selecionamos as regras. No modo de produção as regras de privacidade são mais robustas. Mas como estamos aprendendo ainda, pode deixar no modo de teste.

Agora precisamos conceder as permissões ao FlutterFlow. Para isso, clique na engrenagem no canto superior esquerdo da tela e em seguida Usuários e Permissões.

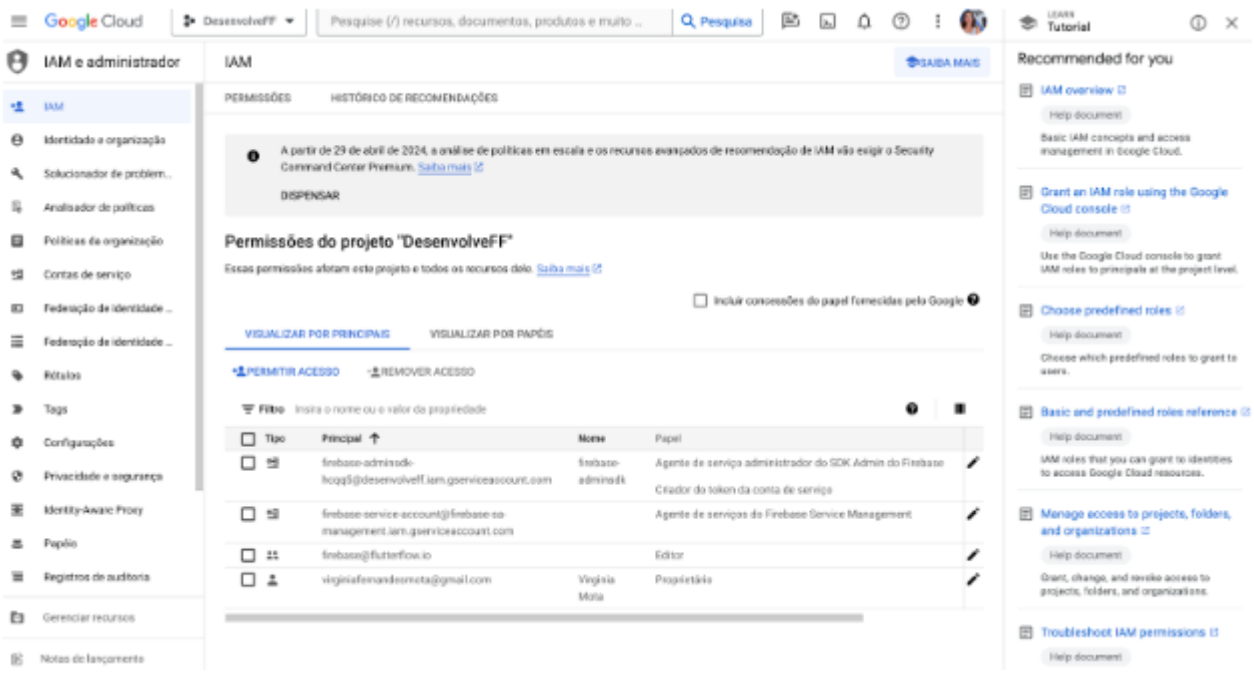




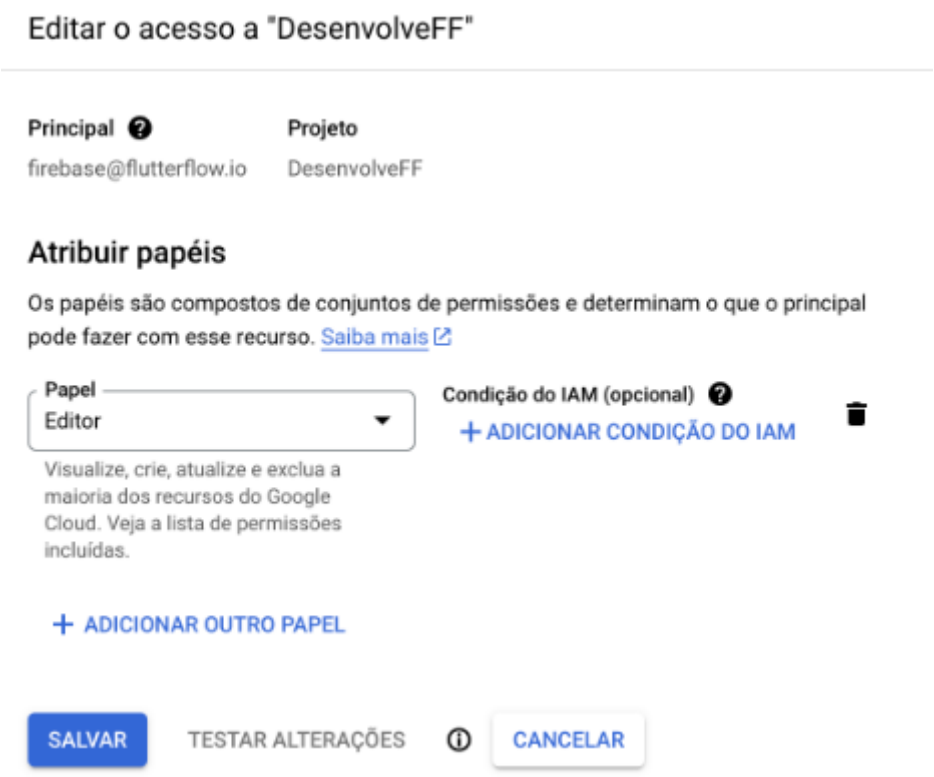
Clique em Adicionar membro e adicione o e-mail firebase@flutterflow.io como editor.



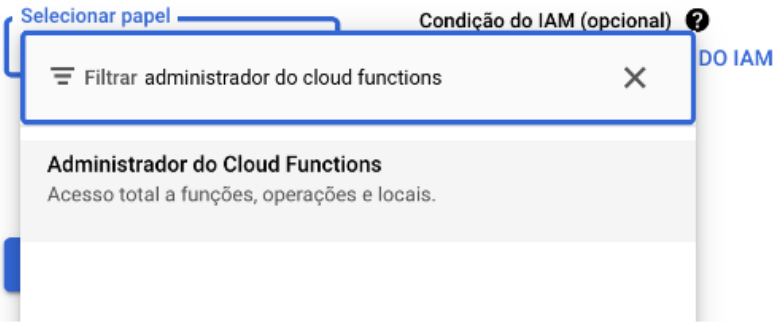
Agora clique em Configurações avançadas de permissões. Isso abrirá uma nova aba com o Google Cloud.



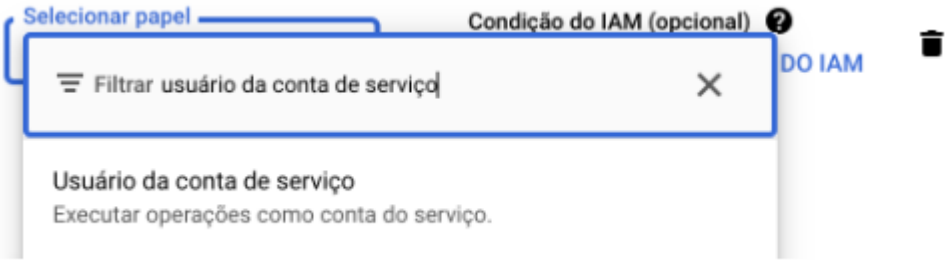
Vá até o email do firebase@flutterflow.io e clique no ícone para editar (lápis).



Clique em ADICIONAR OUTRO PAPEL e selecione o papel **ADMINISTRADOR DO CLOUD FUNCTIONS.**

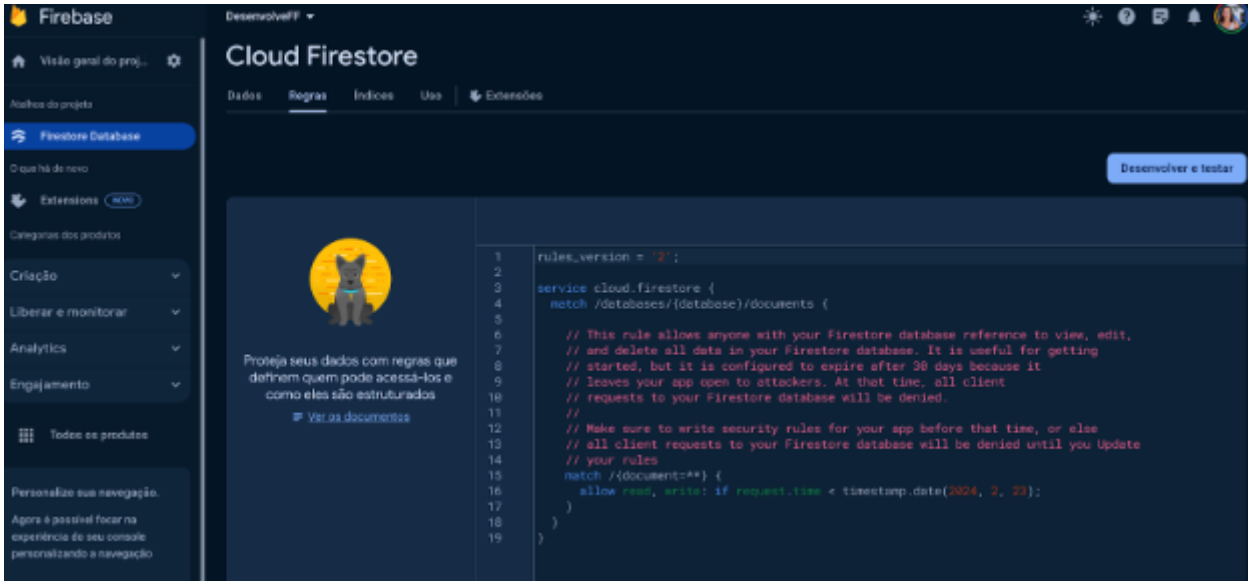


Agora adicione mais um papel: **USUÁRIO DA CONTA DE SERVIÇO.**



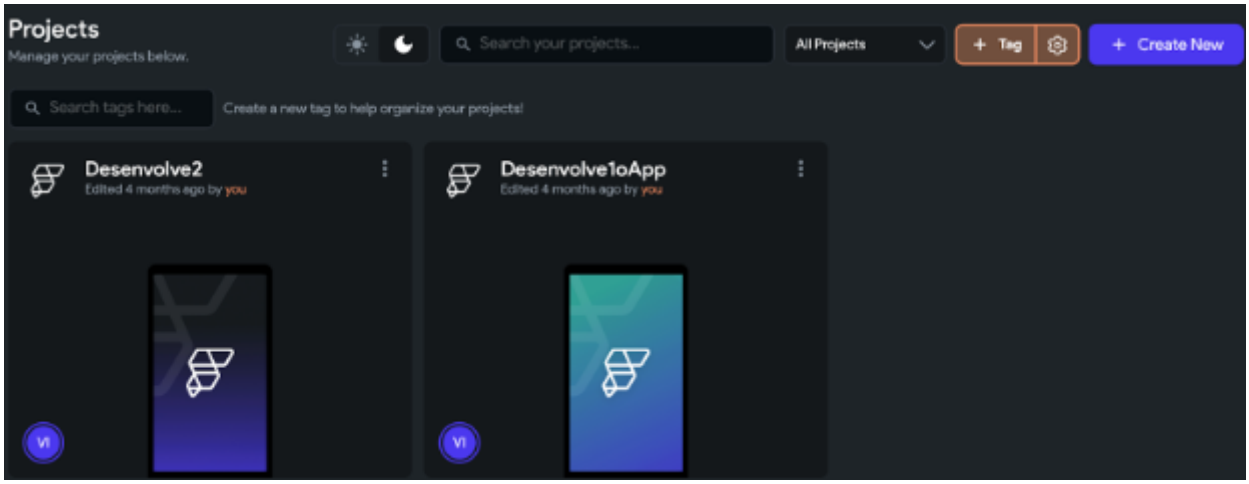
Feito isso, clique em Salvar.

Uma coisinha antes de voltarmos ao FlutterFlow. Por padrão, o Firestore Database cria uma regra que você pode trabalhar com seu banco de dados por 30 dias. Então clique no Firestore Database, vá na aba regras e altere a linha 16 para uma data mais longe. Você pode, por exemplo, colocar daqui um ano.

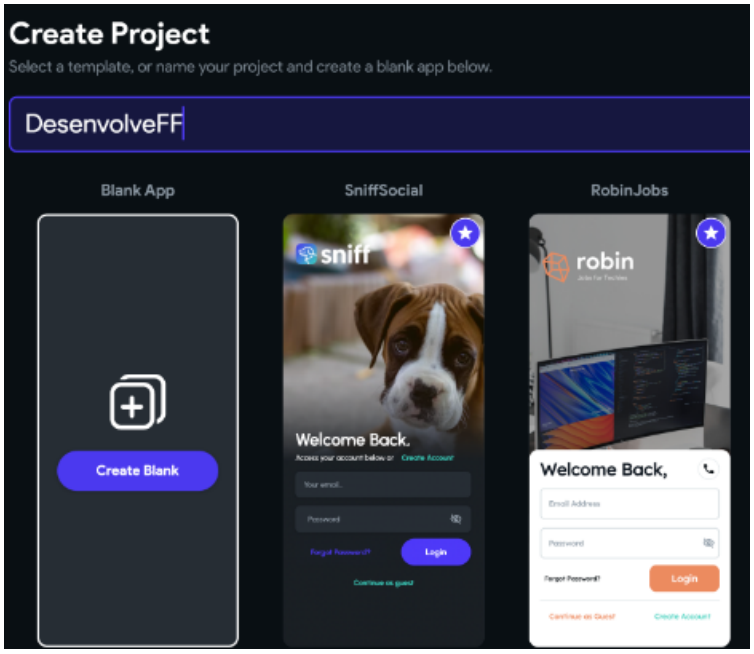


Criando o projeto

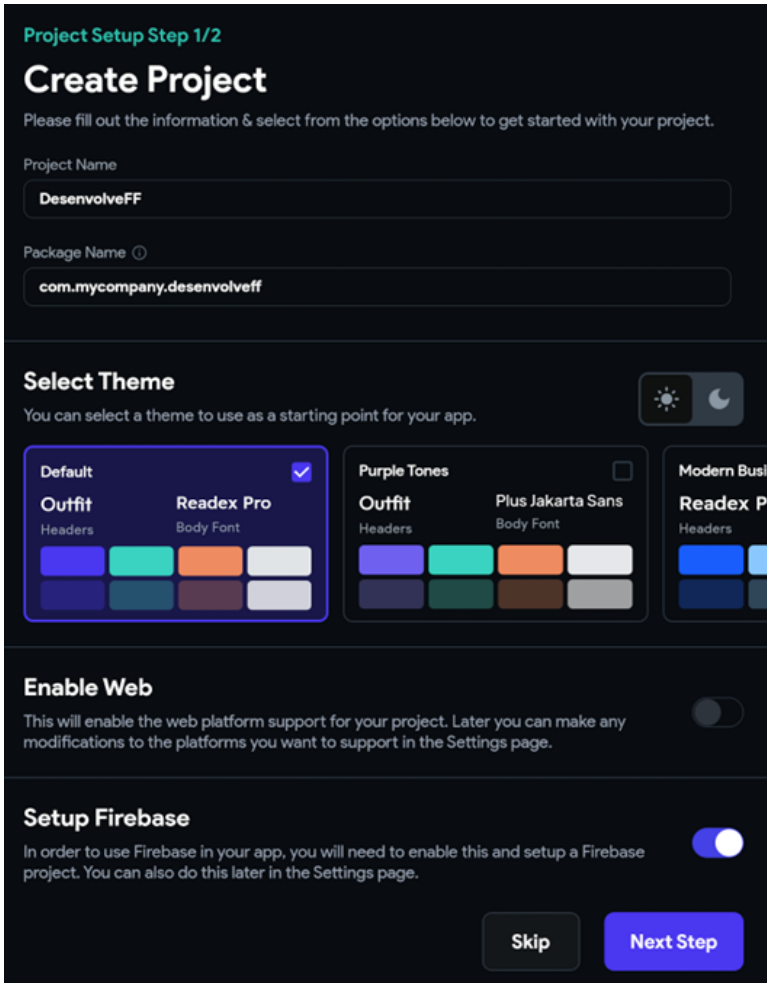
Vamos agora voltar para o FlutterFlow e criar nosso projeto. Volte para o Dashboard do FlutterFlow e clique em **Create New**.



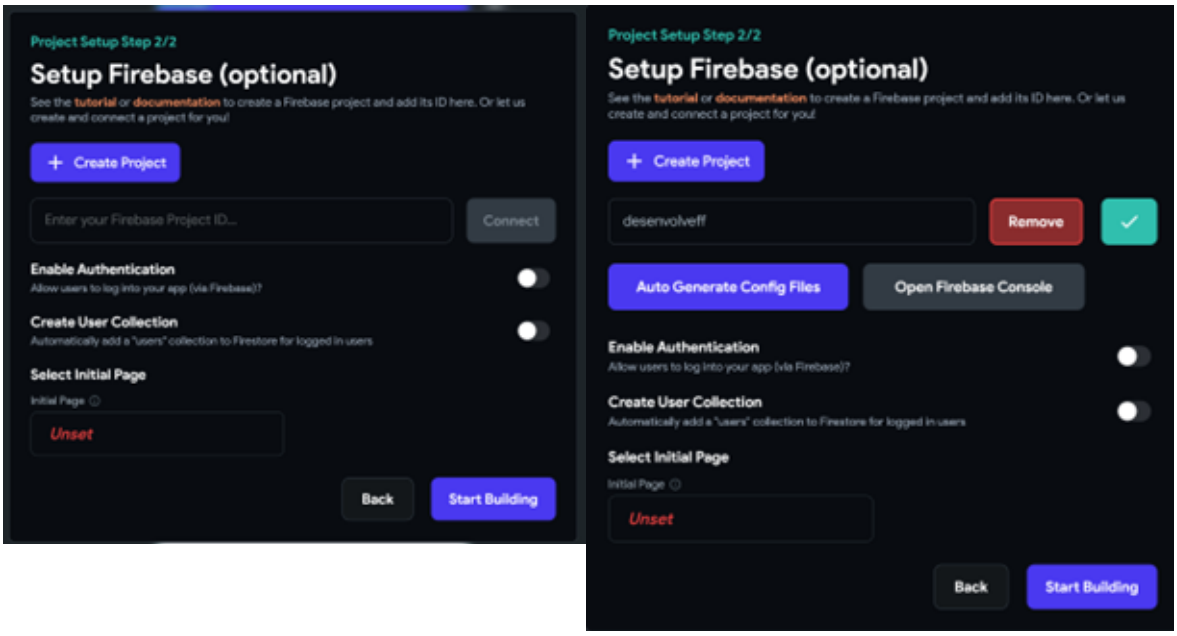
Escolha um nome e selecione **Create Blank**.



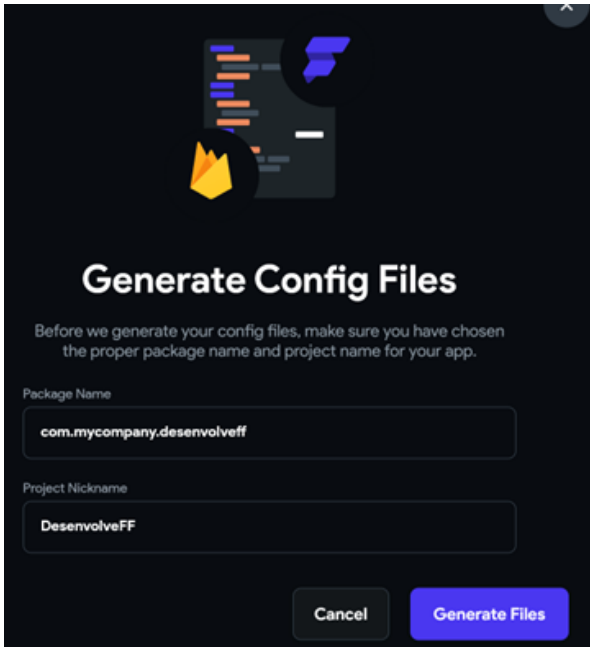
Agora precisamos definir as configurações iniciais do nosso projeto e clique em **Next Step**.



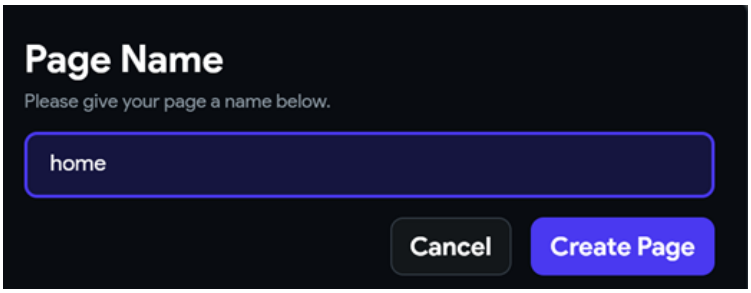
Nessa tela é pedido o ID do projeto Firebase. Para conseguir esse ID, volte ao Firebase clique na engrenagem, vá em Configurações de projeto e copie o código do projeto. Copie e cole no campo. Feito isso, clique em **Connect**.



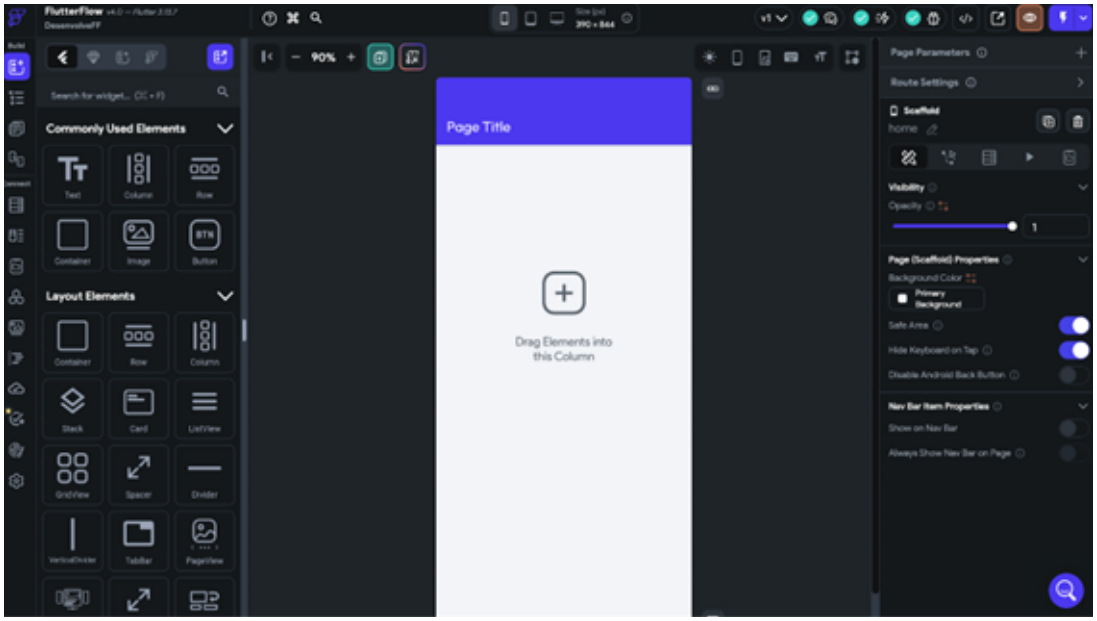
Depois clique em **Auto Generate Config Files > Generate files.**



Agora vá em **Initial Page**, que está marcada como **Unset** e selecione o **Create** blank. Essa página pode ser chamar **home**.

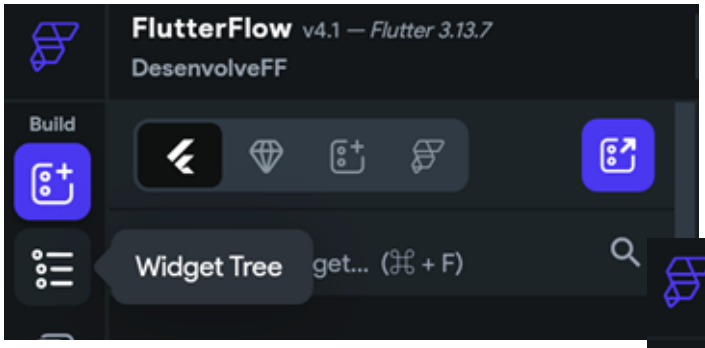


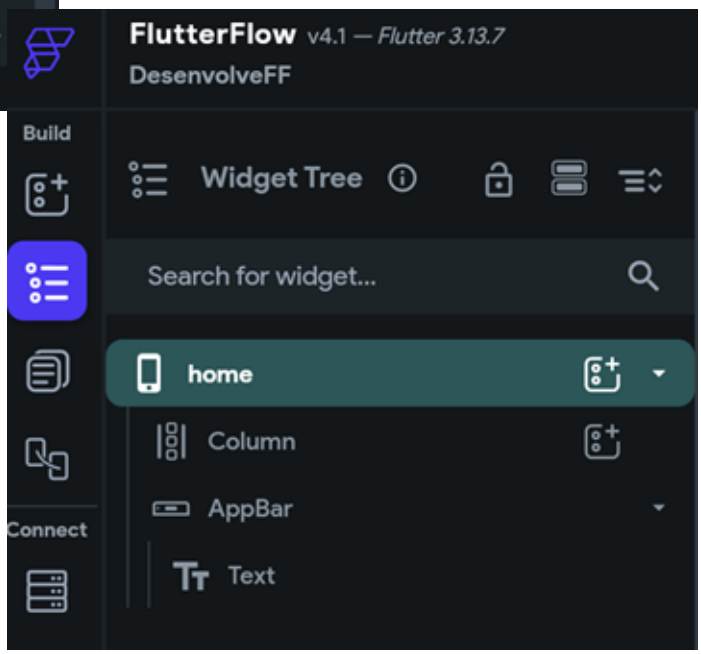
Prontinho! Podemos ir para **Start Building**. Assim chegamos no editor do Flutter Flow.



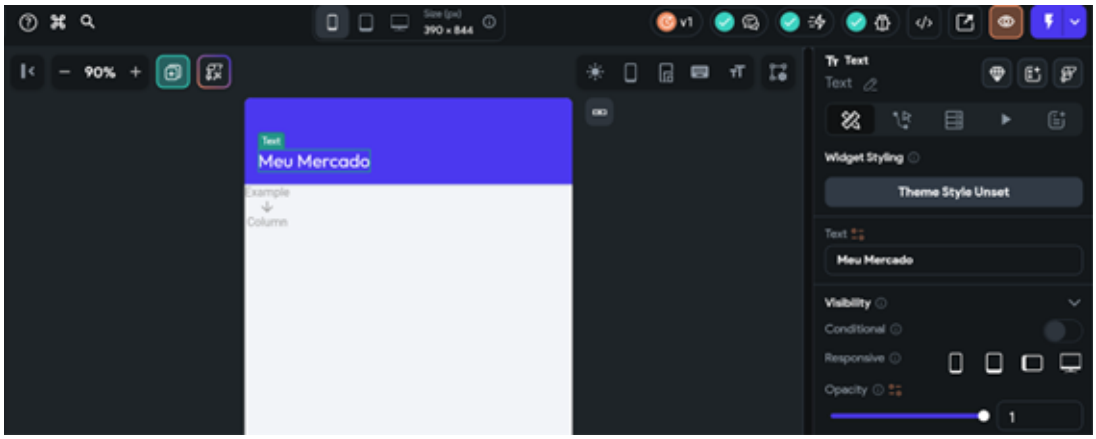
Criando as telas do aplicativo

Vamos começar a criar as telas do nosso aplicativo. Na árvore de Widget **(Widget)** vemos que já existem alguns elementos na nossa tela.



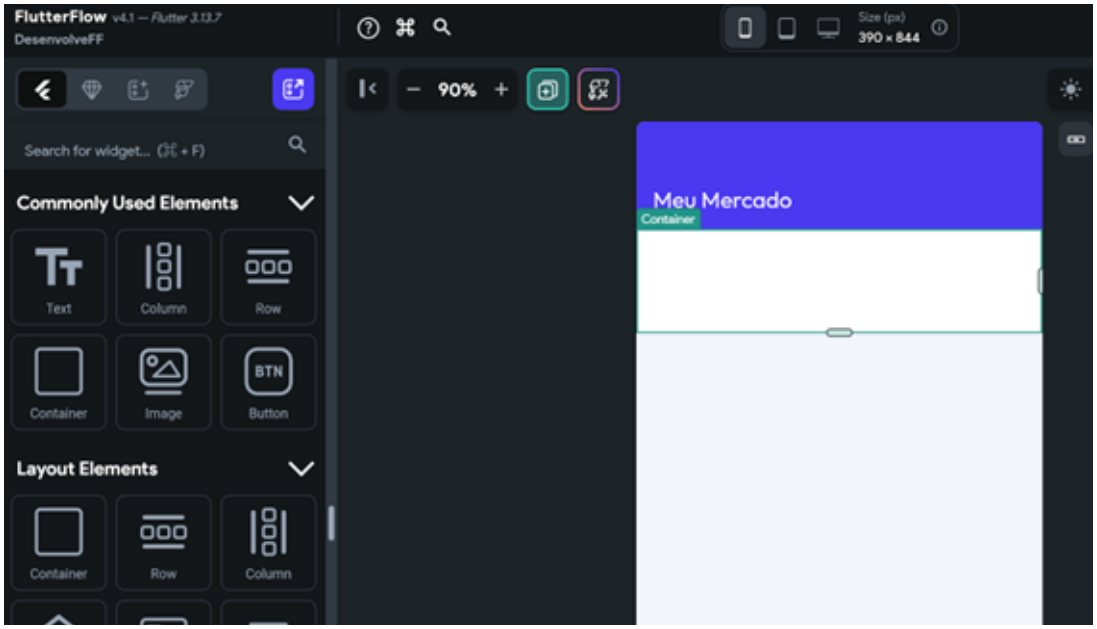


Vamos começar alterando a nossa barra do cabeçalho, a AppBar. Clique na **AppBar** (pode ser na árvore ou direto no elemento) e vá na barra lateral direita, a barra das propriedades.

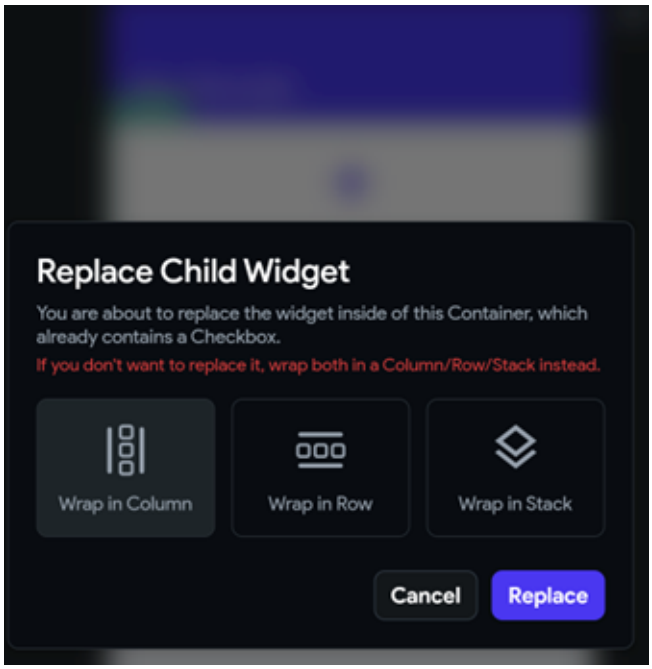


Agora vamos na **Widget Palette** para inserir novos elementos.

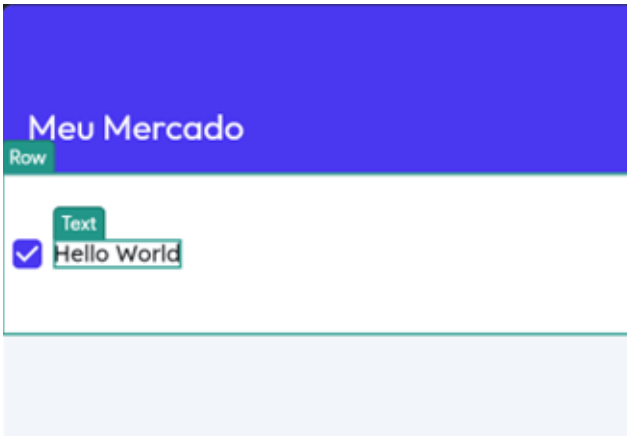
Vamos trabalhar assim como no Bubble e colocar elementos dentro de grupos. Aqui no FlutterFlow, a terminologia é **Container**. Então arraste o elemento **Container** para a tela e verifique se ele está abaixo de **Column** na árvore de widget.



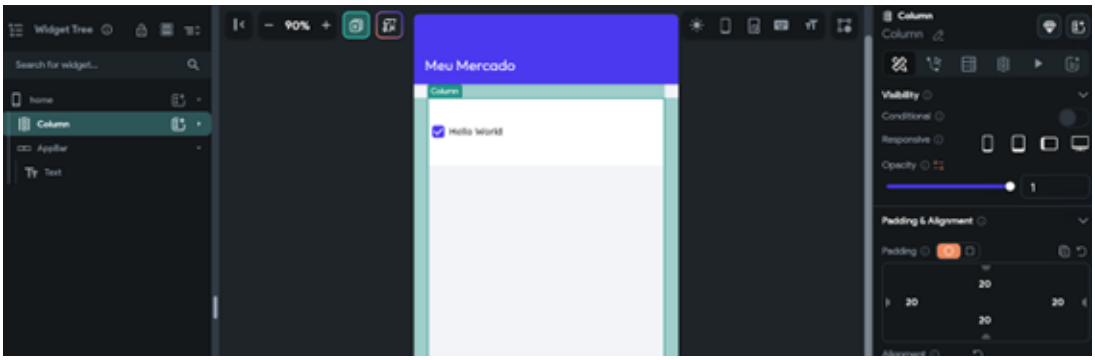
Arraste para o Container uma **Check Box** e um **Text**. Repare que ao arrastar o **Text** a seguinte mensagem aparecerá.



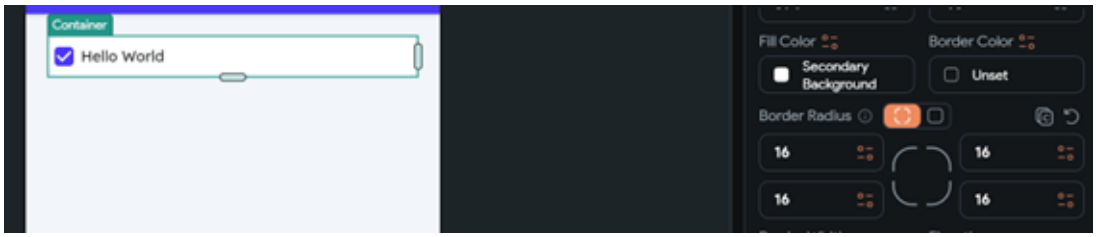
Isso acontece pois estamos trabalhando em colunas, ou seja, com elementos inseridos verticalmente. Mas dentro do nosso container nós queremos elementos em linha. Assim, clique em **Wrap in Row**.

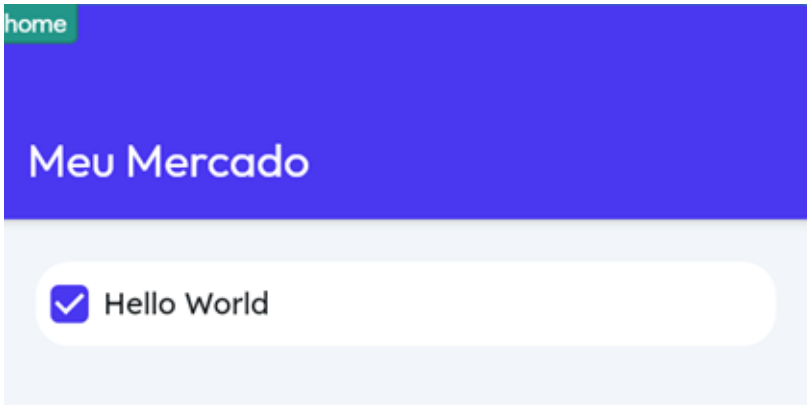


Vamos agora acrescentar um distanciamento das bordas, modificando o atributo **Padding** na propriedade da **Column**.



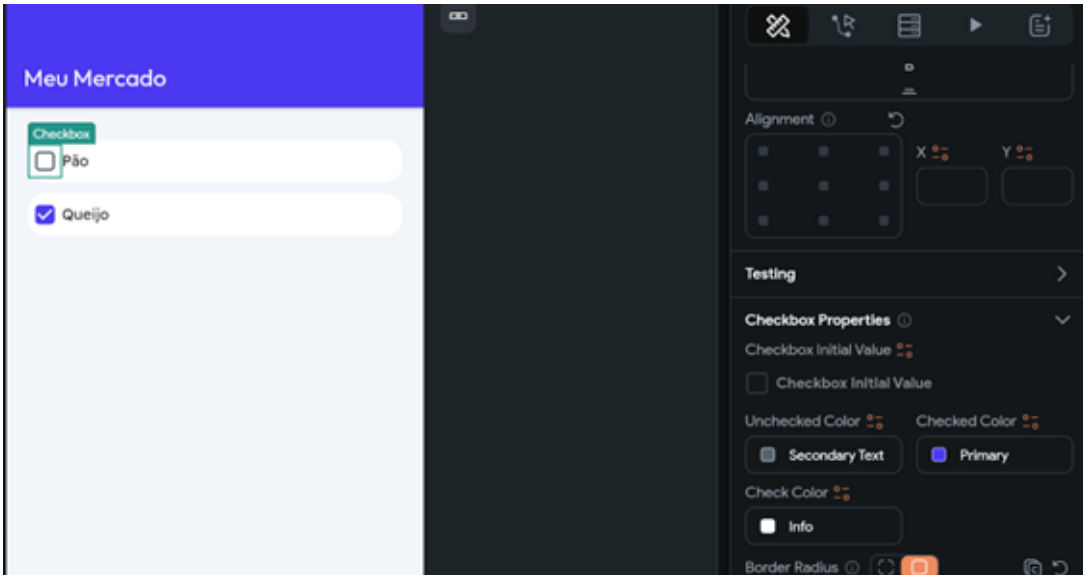
Agora clique no **Container** e vamos ajeitar mais um pouco o seu visual. Altere o **Border Radius.**



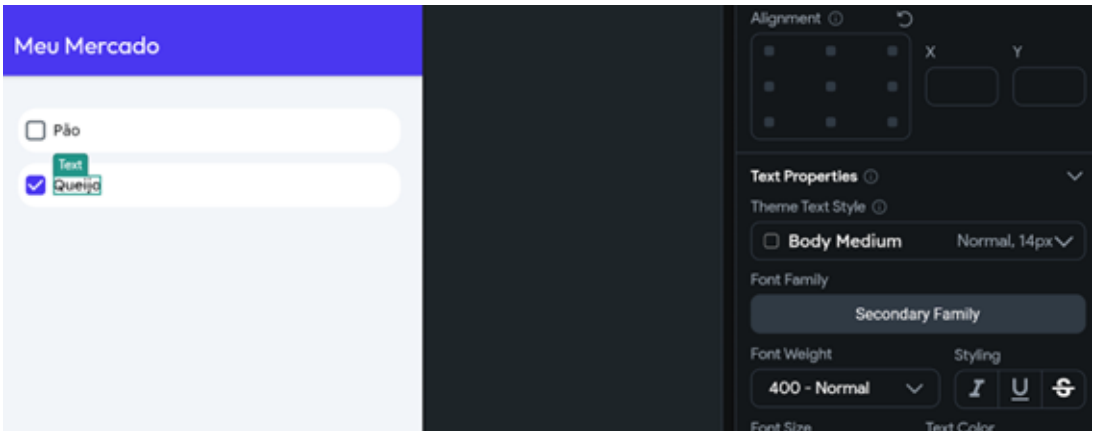


Agora vamos duplicar nosso Container para já ficar com a carinha de lista. Vamos desmarcar uma propriedade da check box para ela ficar desmarcada. Além disso, vamos transformar o texto da checkbox selecionada para **strikethrough**.

Veja que já conseguimos montar a interface mais rápida e bonita do que no Bubble.

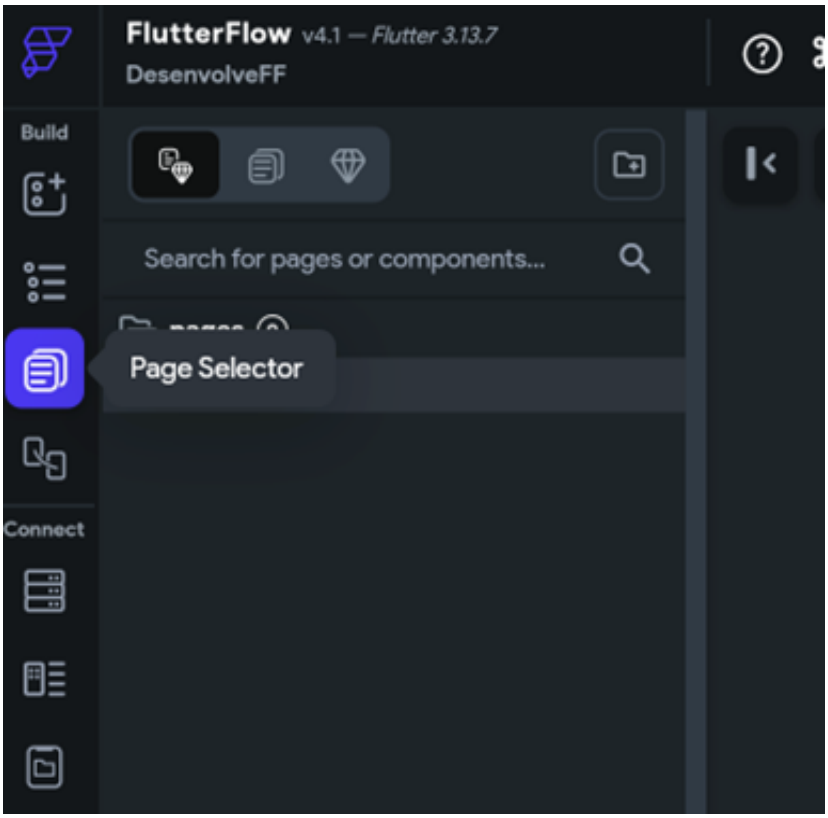


Desmarcando **Checkbox Initial Value** a caixa é desmarcada.

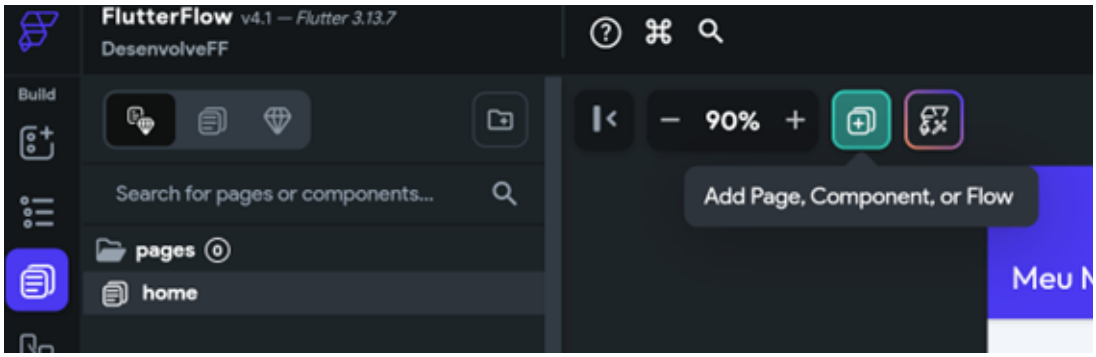


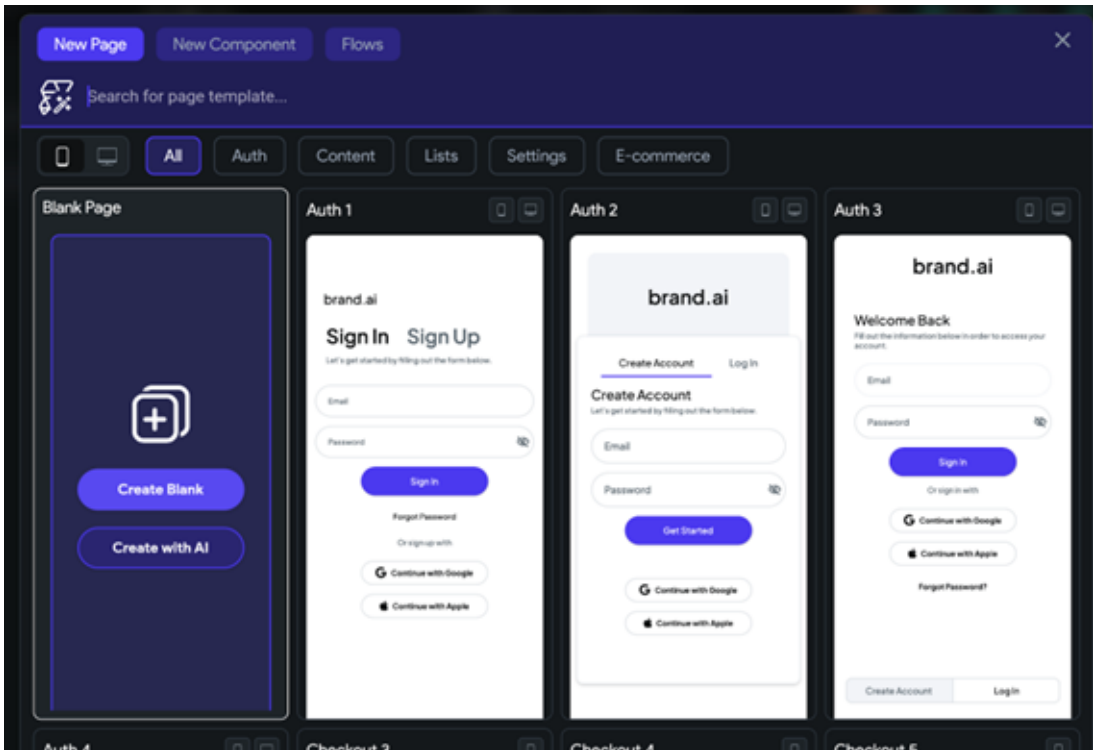
Na propriedade **Styling** do texto, conseguimos deixar o texto cortado.

Vamos agora para a tela de criar o produto. Vamos fazer de uma forma um pouco diferente do Bubble: vamos criar uma tela para adicionar um produto. Vá em **Page Selector**.

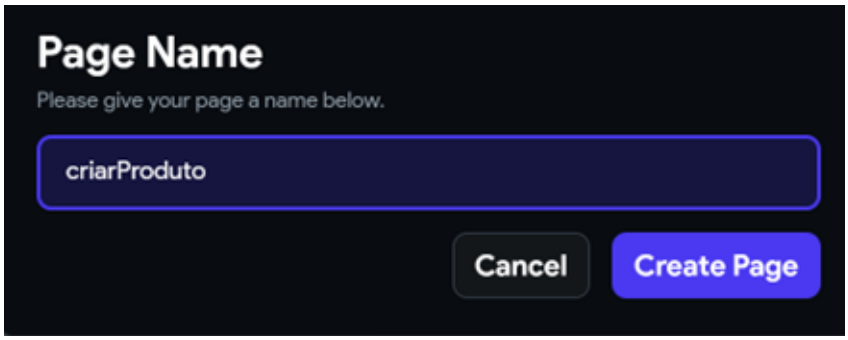


Em seguida, vá no botão verde **Add Page, Component or Flow** e vamos criar uma nova página em branco **Create Blank**.

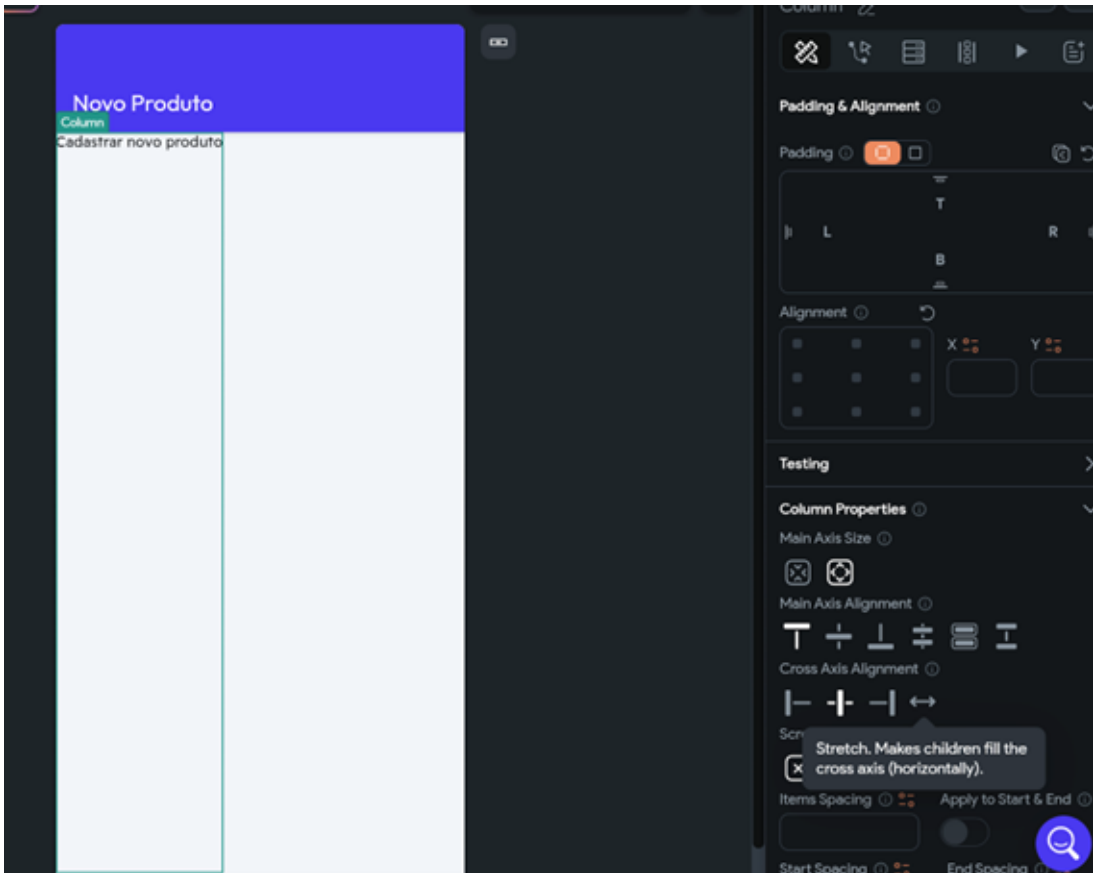


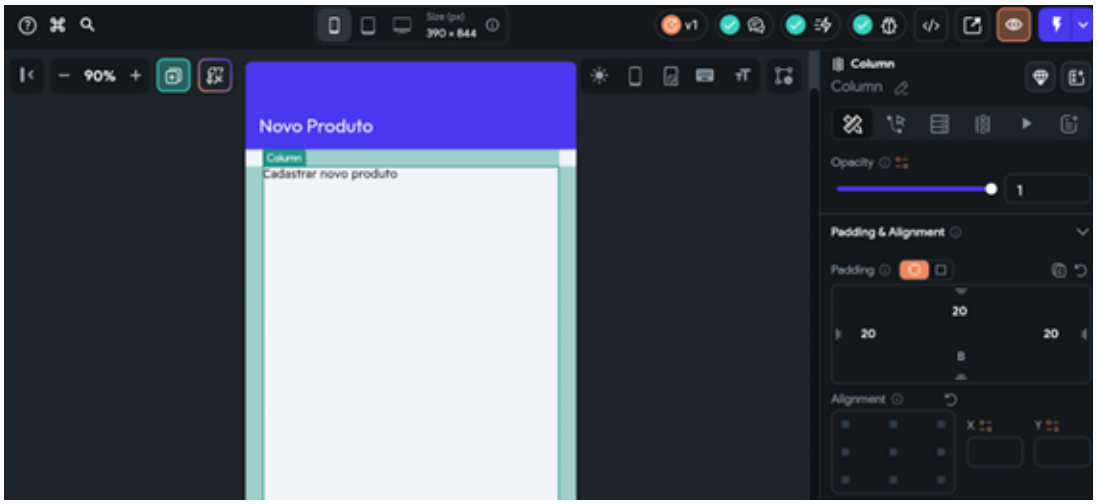


Vamos chamar a página de **criarProduto**.

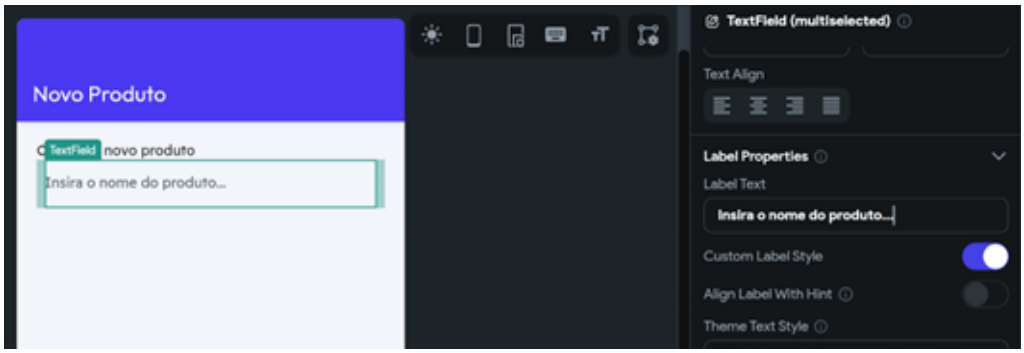


Esta página inicialmente é igual a que nós começamos a trabalhar na **home**. Inicialmente vamos alterar o texto da **AppBar**, acrescentar um **Text** e ajustar a coluna para ocupar toda a largura da página e colocar o espaçamento **(padding)**.

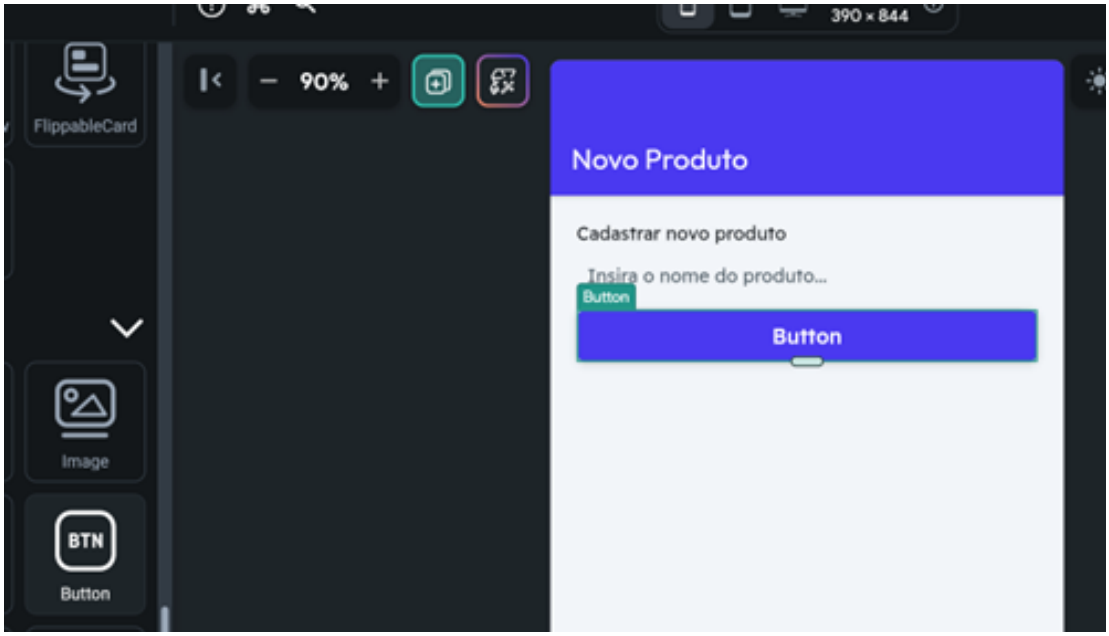




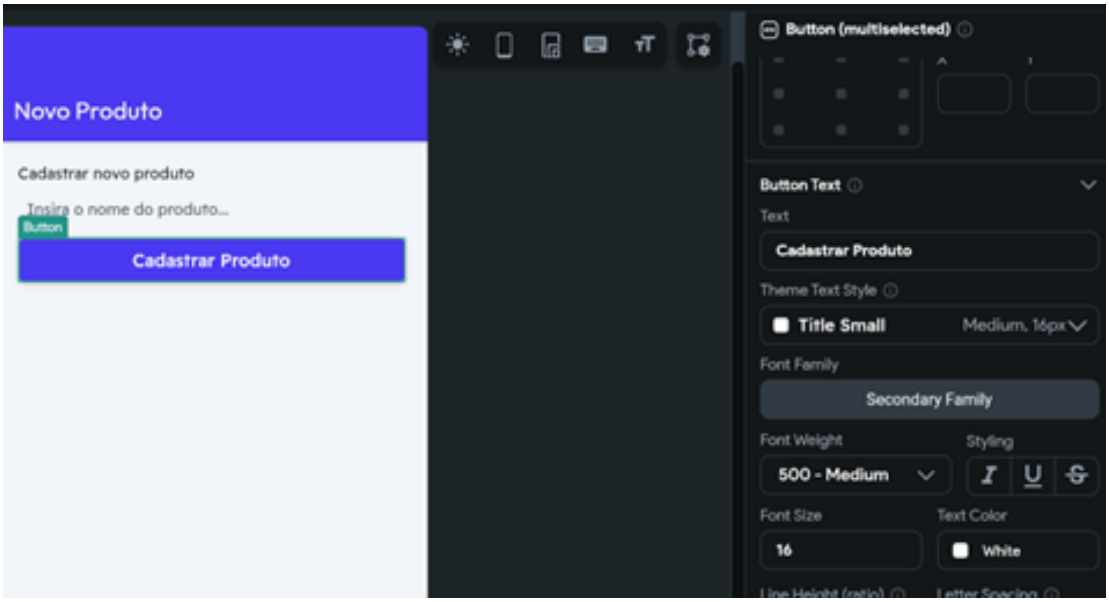
Vamos acrescentar o campo de formulário **(Form Elements) TextEdit**, que é onde iremos inserir o texto do produto que vamos adicionar na nossa lista de mercado. O **placeholder** do TextEdit no FlutterFlow fica em **Label Properties > Label text** e nós trocaremos para **Inserir o nome do produto**.



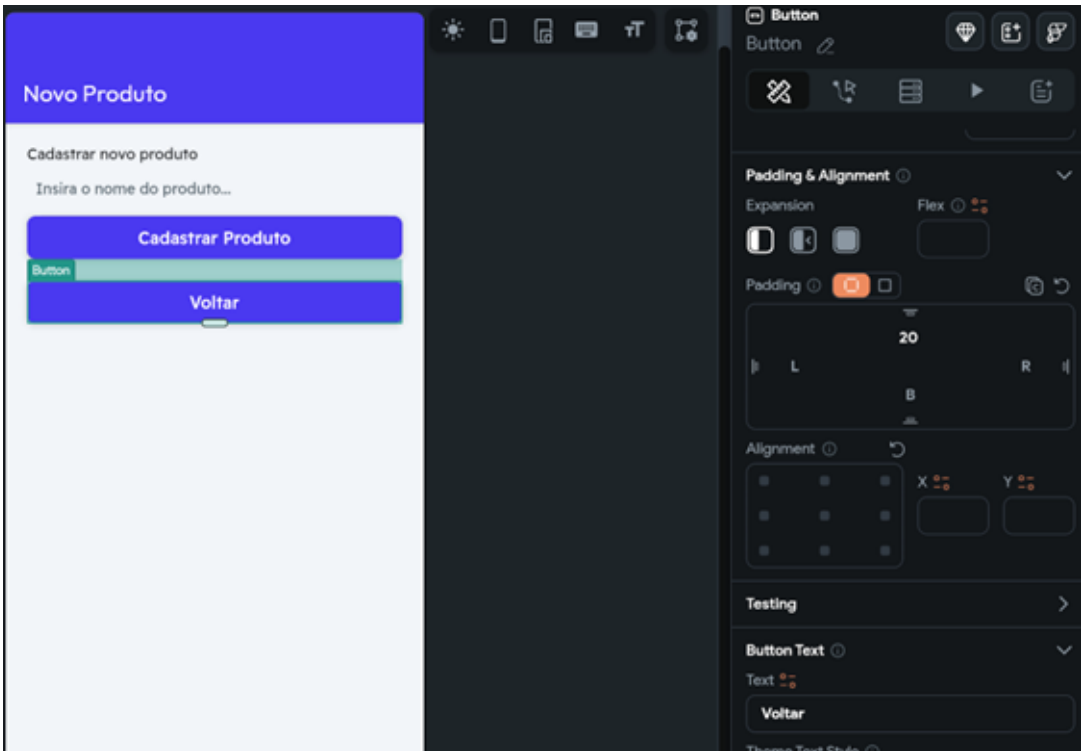
Agora vamos colocar um botão em nossa tela. O **Button** fica em **Basic Elements**.

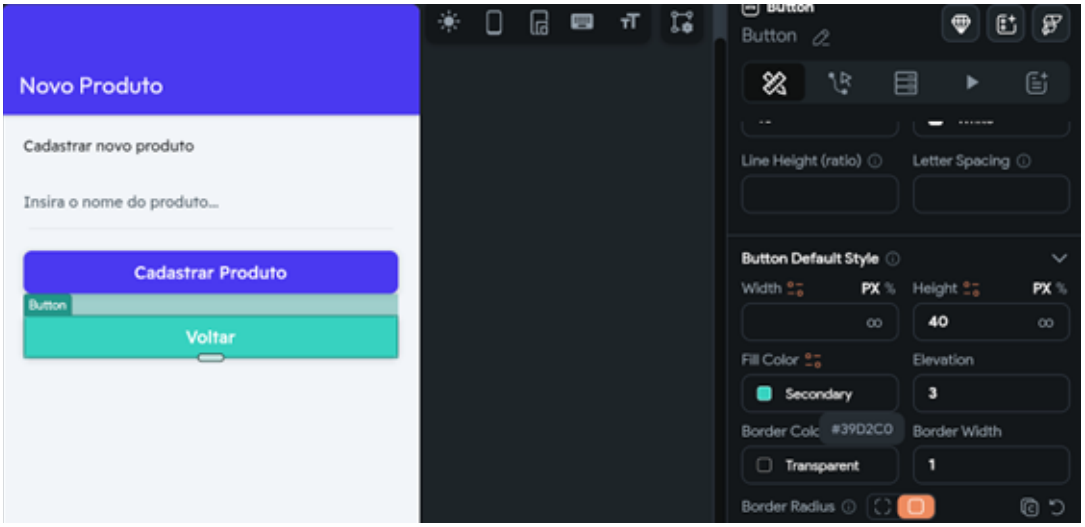


E vamos chamá-lo de **Cadastrar Produto**.



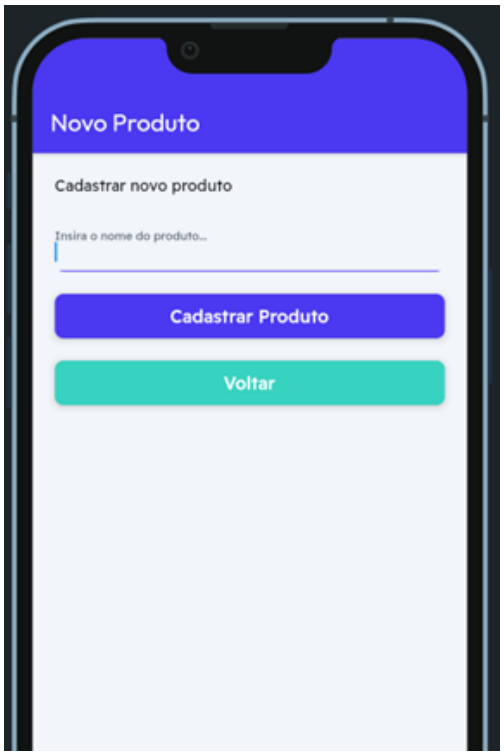
Por boa prática de usabilidade, vamos também criar um botão voltar. Esse botão não deve ter tanto destaque quanto o botão cadastrar, por isso vamos alterar a **Fill Color** dele.



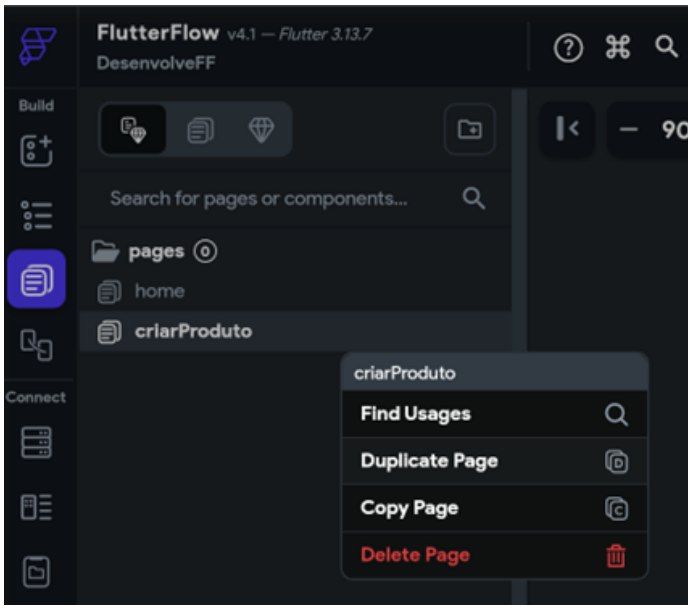


Explore as propriedades de todos os elementos, o FlutterFlow tem muitas possibilidades!

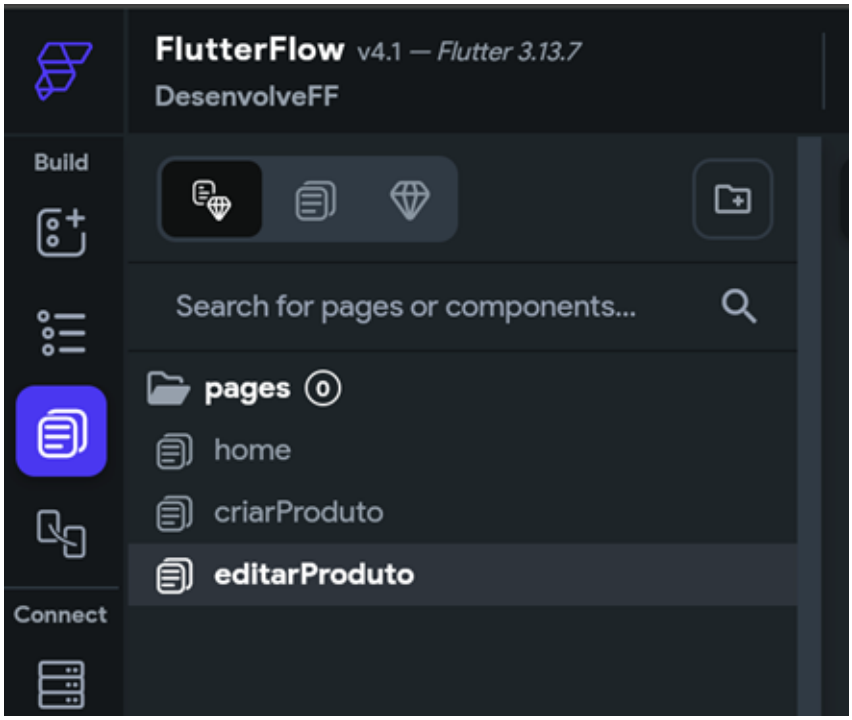
Clicando no botão **Preview**, já podemos ver como está ficando o nosso aplicativo.



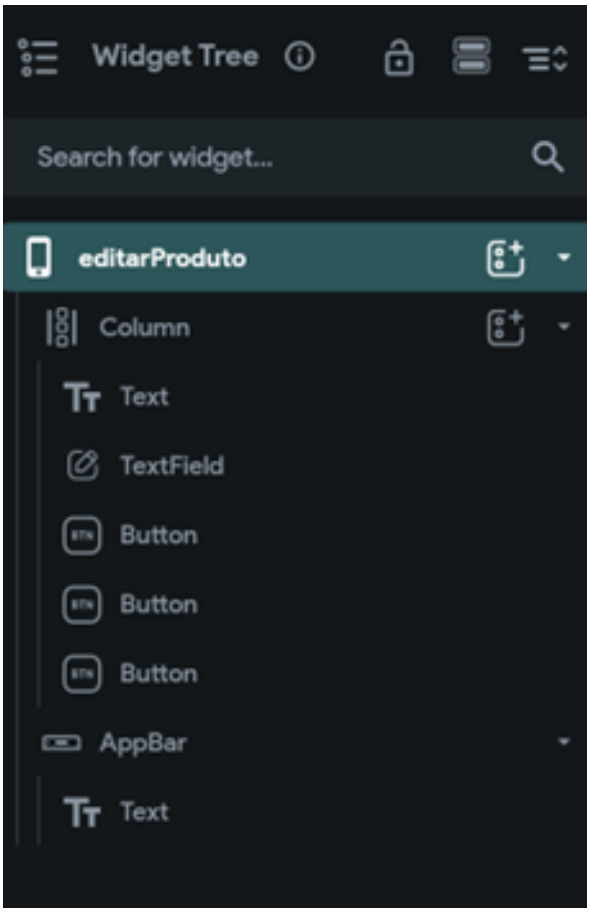
Pronto, já temos a tela **home** e a tela de **criarProdutos**. Vamos fechar agora com a tela de **editarProduto**. A nossa tela de edição será bem parecida com a tela de criação. Logo, vamos copiá-la e alterar o que for preciso. Clique com o botão direito na página **criarProdutos** no **Page Selector** e clique em **Duplicate Page**.



Para alterar o nome da página, clique duas vezes e digite o novo nome.



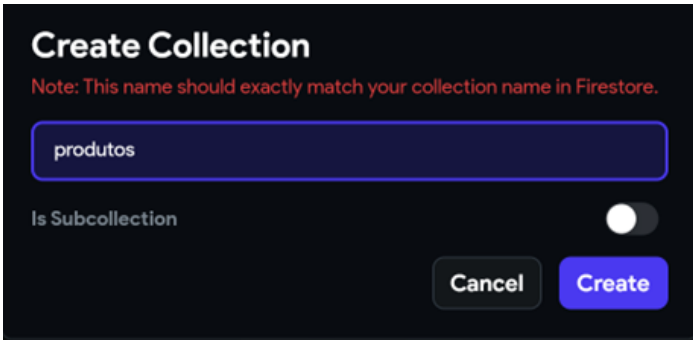
Além dos elementos que já temos na tela, vamos acrescentar um botão **Apagar** produto. Assim nossa tela de edição será bem completa. Nossa tela de edição ficará:



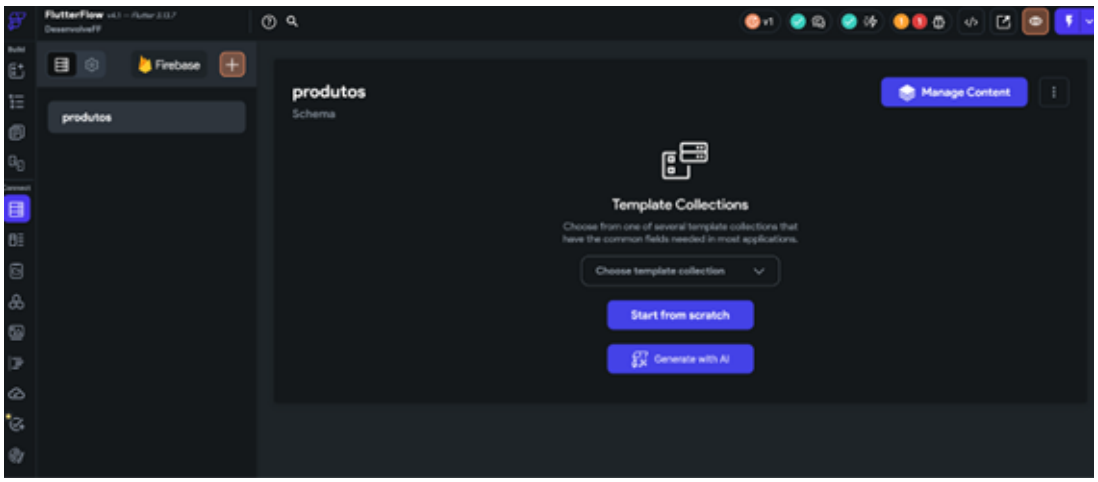
CRUD - Create

Até o momento exploramos as abas de **Build**, isto é, toda a parte de construção da interface do aplicativo. Vamos preparar o banco de dados do Firebase que configuramos anteriormente. Para isso começaremos a trabalhar na aba **Connect**.

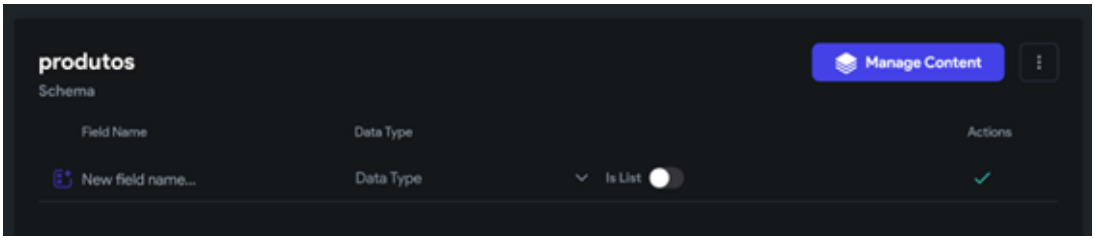
Na aba **Firestore**, vamos em **Create Collection**. A coleção que vamos criar é **produtos**.



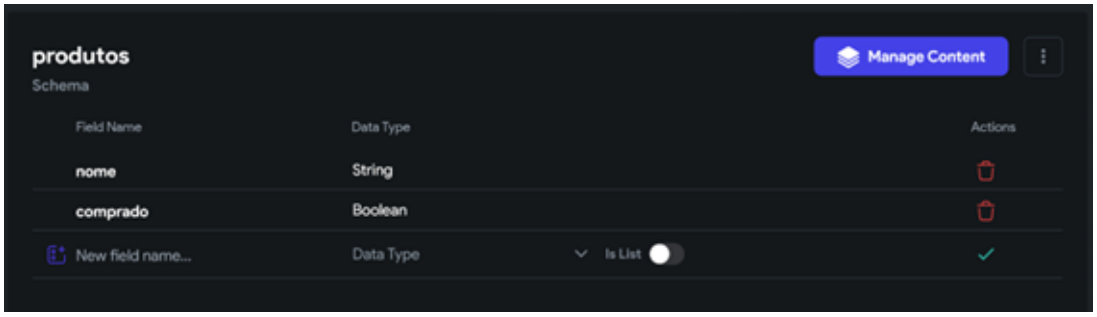
Para a criação do banco de dados, o Firebase oferece alguns templates prontos. Nós vamos trabalhar com um do zero, para entendermos bem os conceitos. Assim, clique em **Start from scratch**.



Agora é hora de montar o nome dos campos e o tipo desses dados.

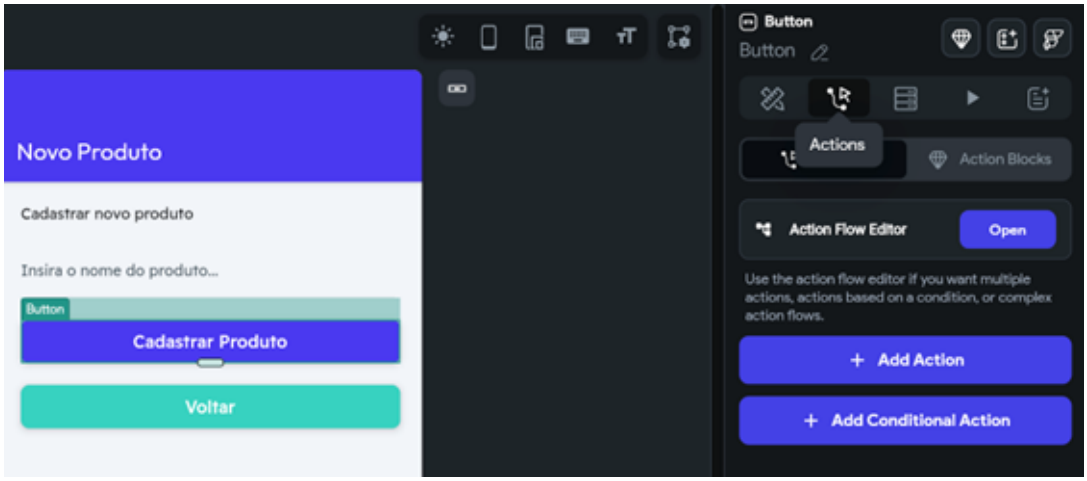


Assim como fizemos no banco de dados do Bubble, vamos ter dois campos para o produto: nome e o comprado. Nome é um campo de texto, logo uma String. Comprado é o campo sim ou não, logo booleano.

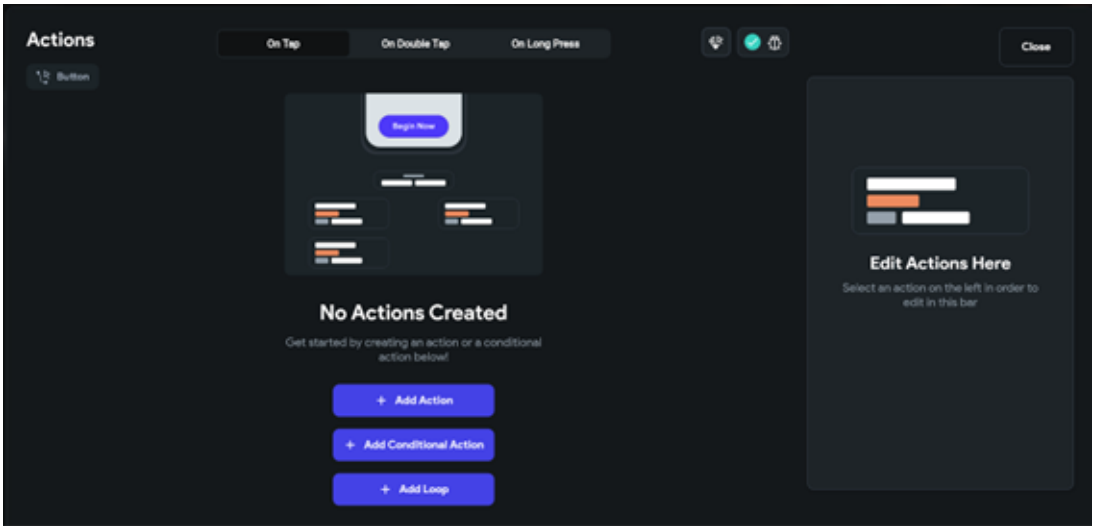


Nosso banco está preparado para criarmos as ações com nossos dados. Para isso vamos entender mais um conceito sobre banco de dados: CRUD. CRUD é o acrônimo para Create (criar), Read (ler), Update (atualizar) e Delete (apagar). Com essa explicação, já dá para intuir que o CRUD é uma sequência de funções de um sistema que trabalha com banco de dados, seja ele na sua máquina ou na nuvem.

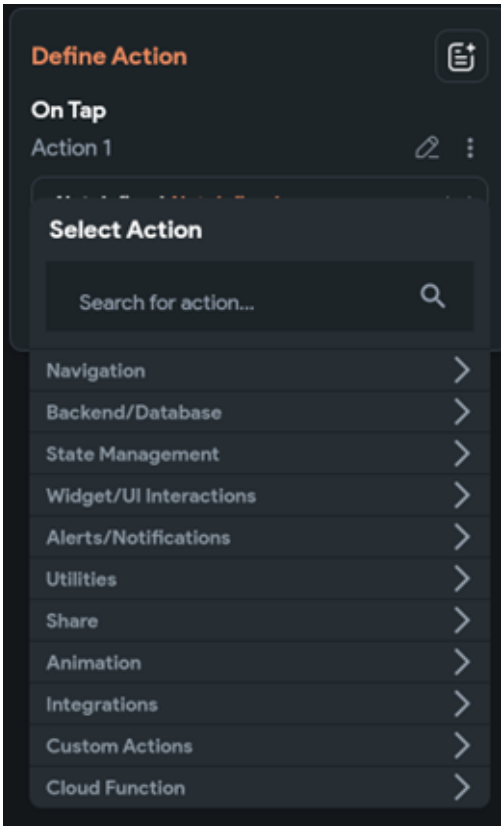
Vamos começar fazendo a função criar. Vá na tela **criarProduto** e clique no botão **cadastrar**. Ao clicar no widget do botão, vamos adicionar uma ação.



Aqui vamos abrir o editor de fluxo de ação **(Action Flow Editor)**.

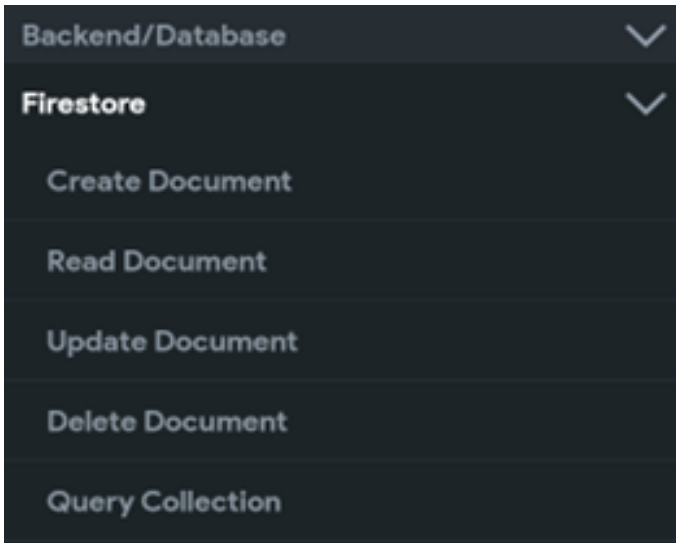


Clique no botão **Add Action**. Vamos adicionar uma ação em **On Tap**, isto é, ao clicar no botão. O FlutterFlow tem uma lista extensa de possibilidades de ação.



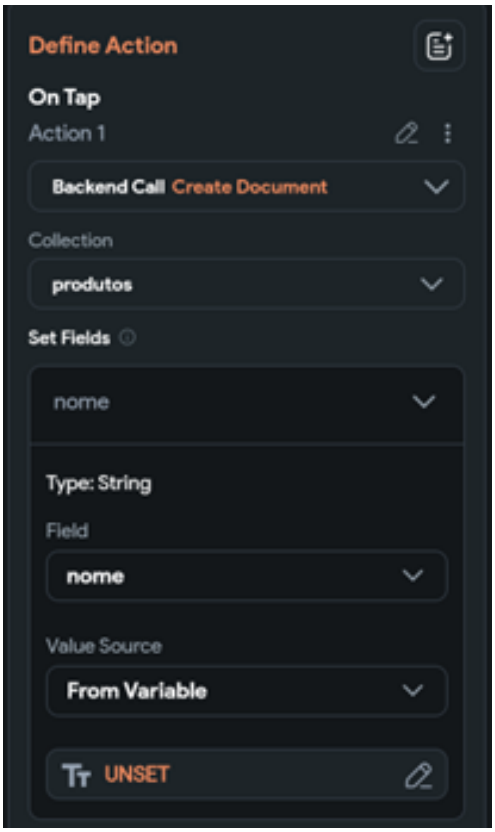
Aqui queremos trabalhar com o banco de dados, logo vamos nas ações de **Backend/Database**. Nós vamos trabalhar com o CRUD do Firestore.

Aqui vemos direitinho o CRUD.

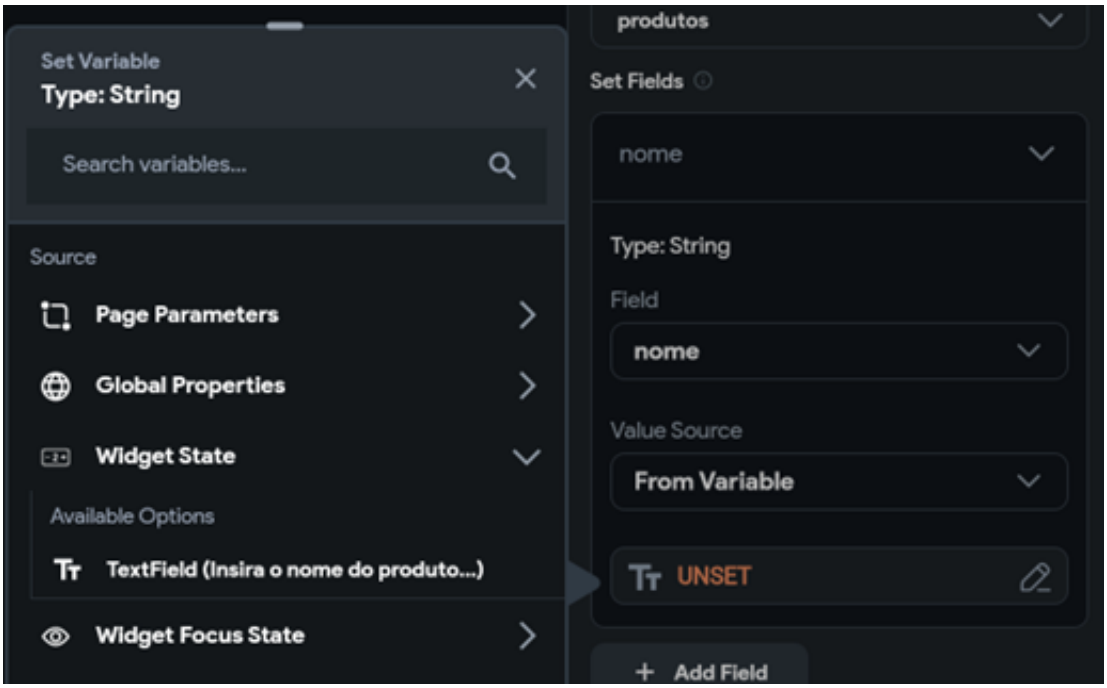


Nesse ponto, queremos criar um documento, logo **Create Document.**

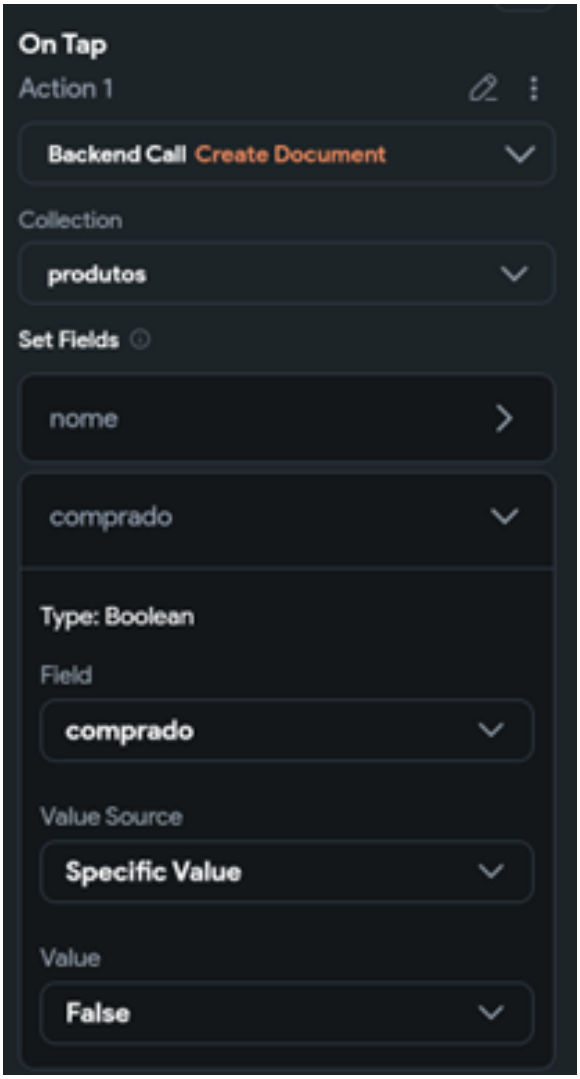
Nós precisamos selecionar a coleção produtos, o campo nome.



O campo nome vem de uma variável: o valor de TextField.

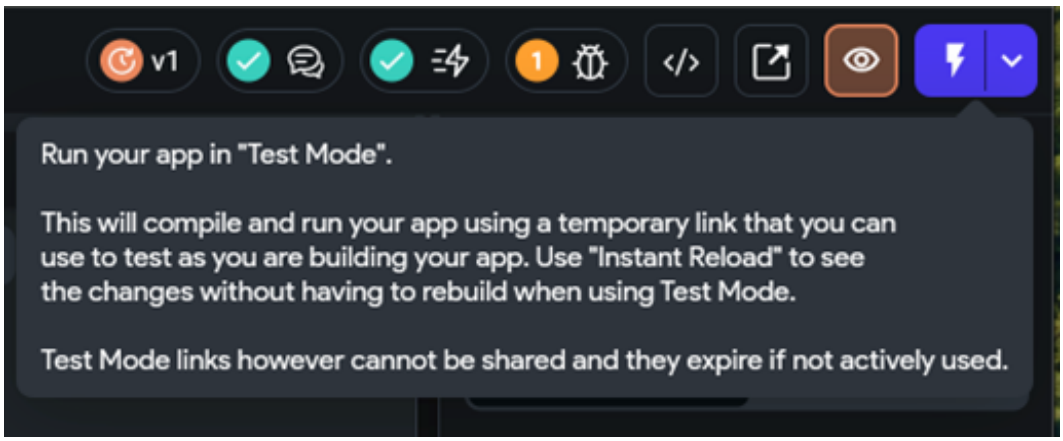


Vamos agora adicionar também o campo comprado, pois ao inserir um novo produto ele deve ter falso **(false)** como valor inicial.



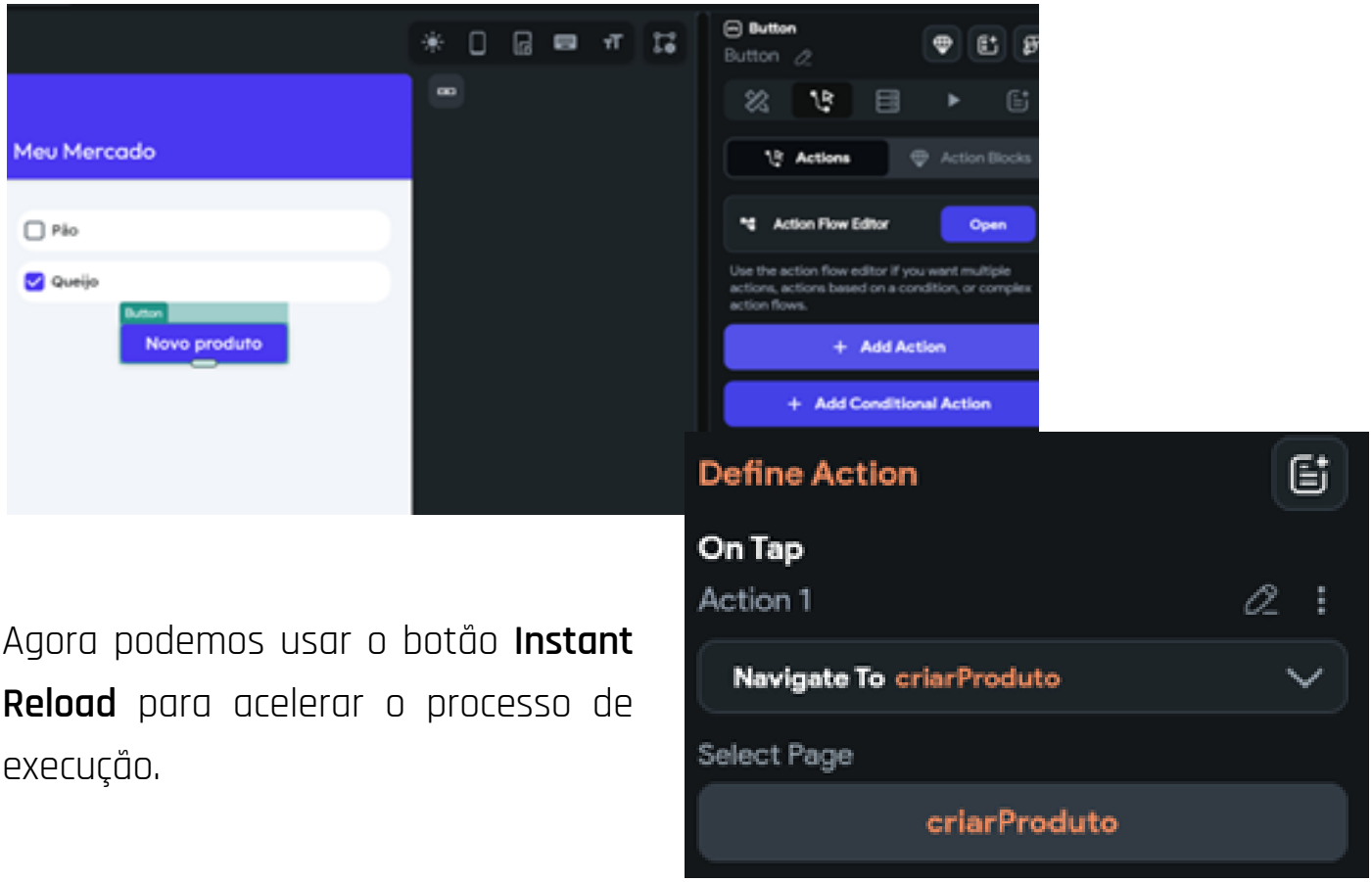
Pronto, só clicar em **Close** e nossa ação para o botão Cadastrar estará definida.

Para testarmos nossa ação, vamos rodar nosso aplicativo criando no raio do canto superior direito.

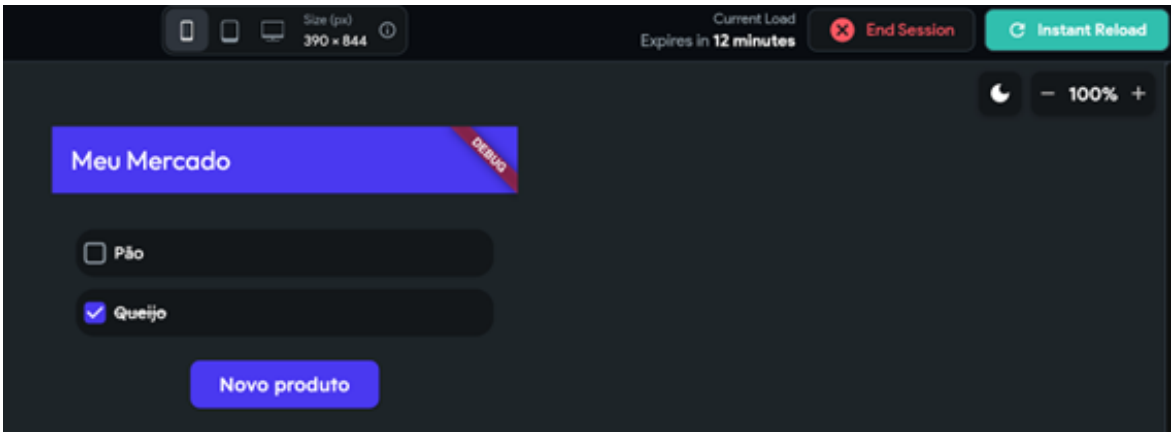


Na primeira vez que clicamos em **Run**, ele demora até 4 minutos para executar. Quando começar a rodar, percebemos que a tela que abre é nossa tela de **home**. Isso acontece pois por padrão essa é a tela inicial. Então o que falta? Falta colocar uma navegação para a tela de criação de produto.

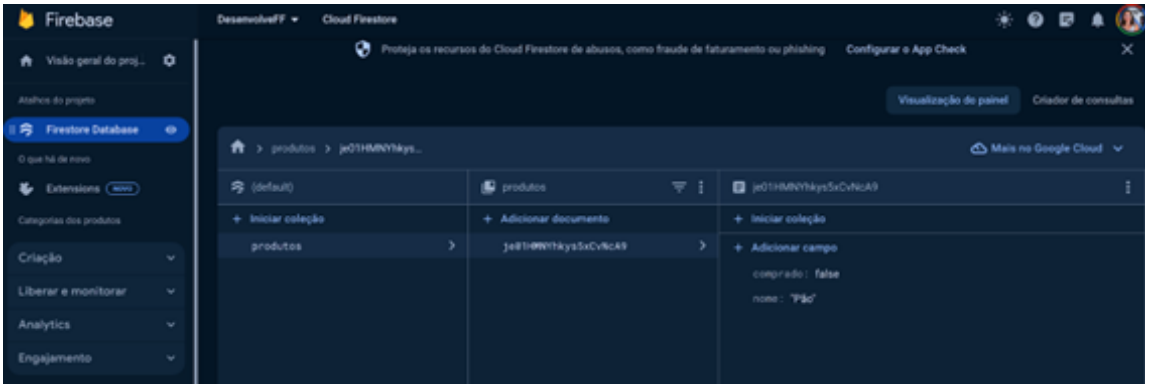
Basta então adicionarmos um botão na tela e criar a ação de **Navigate to**.



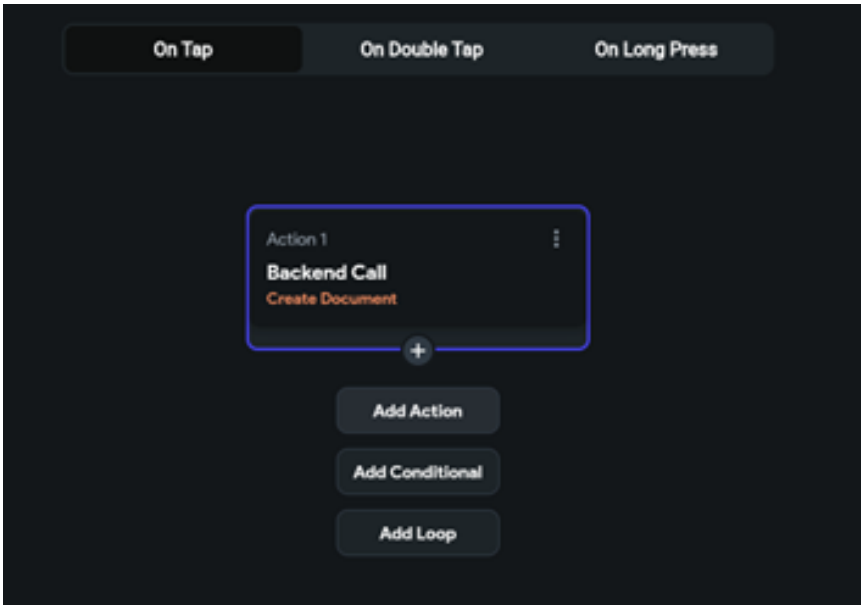
Agora podemos usar o botão **Instant Reload** para acelerar o processo de execução.

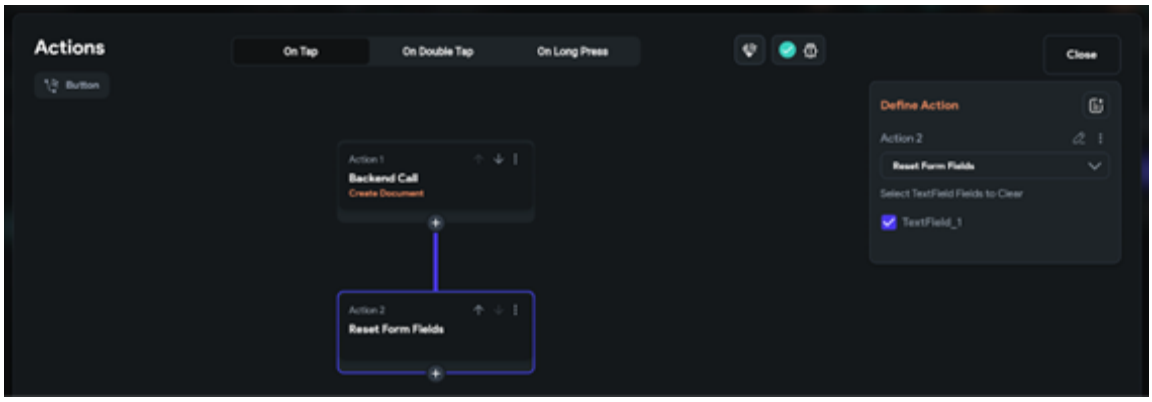


Pronto! Agora é só testar a conexão com o nosso banco. Cadastre um produto e vá até o **Firestore**.

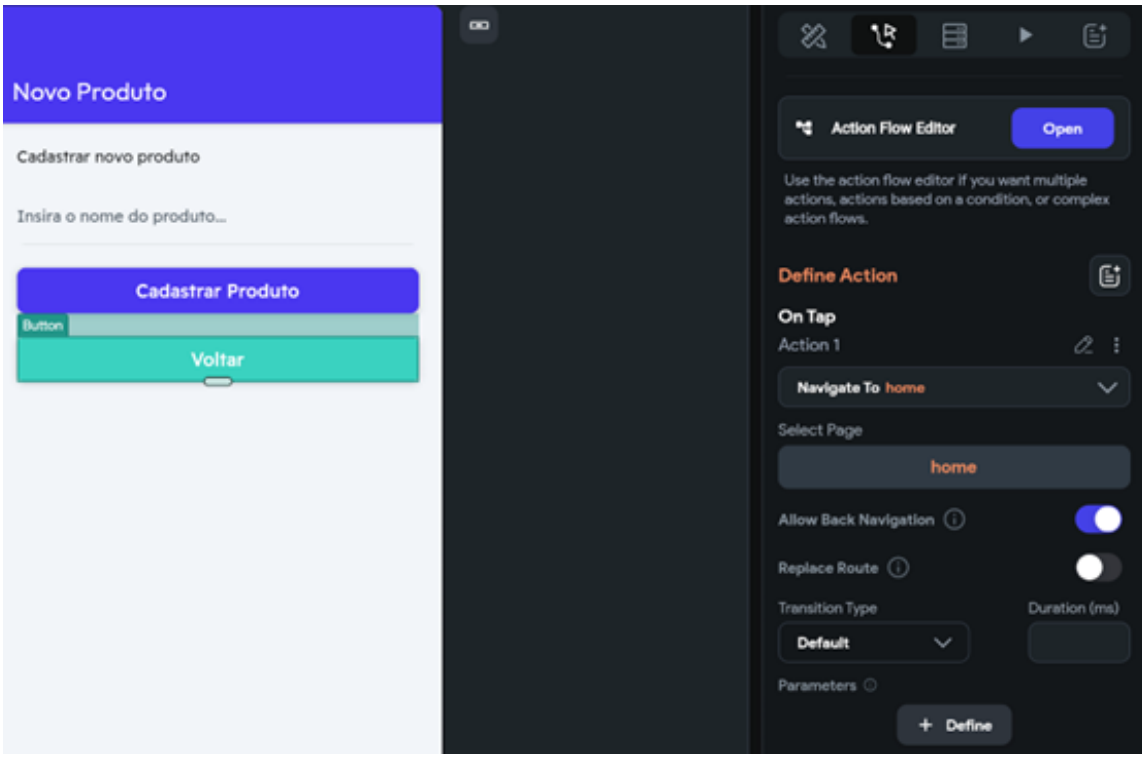


Vamos dar uma melhorada na usabilidade da nossa tela de cadastro e inserir mais uma ação: apagar o campo de texto depois de cadastrar um produto. Vamos voltar no fluxo de ações do botão de cadastro e adicionar uma ação de **Reset**.





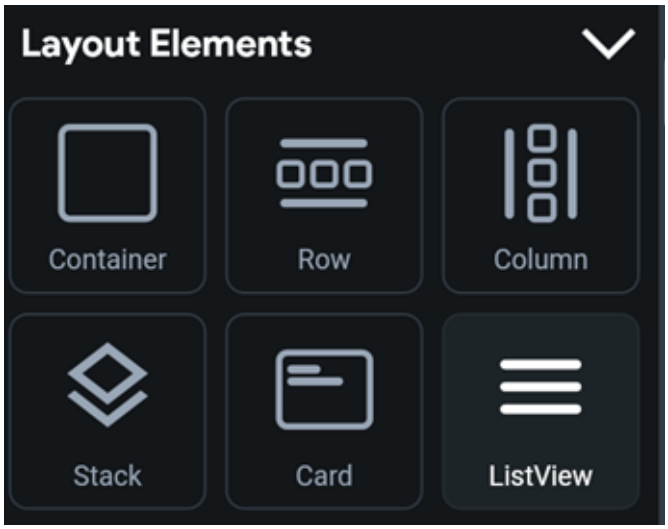
Como já sabemos criar uma ação para o botão, vamos aproveitar e configurar o botão **Voltar** da nossa tela de cadastro de produtos.



CRUD - Read

Vamos agora com o processo de leitura **(Read)** do nosso banco de dados para visualizar a lista na tela.

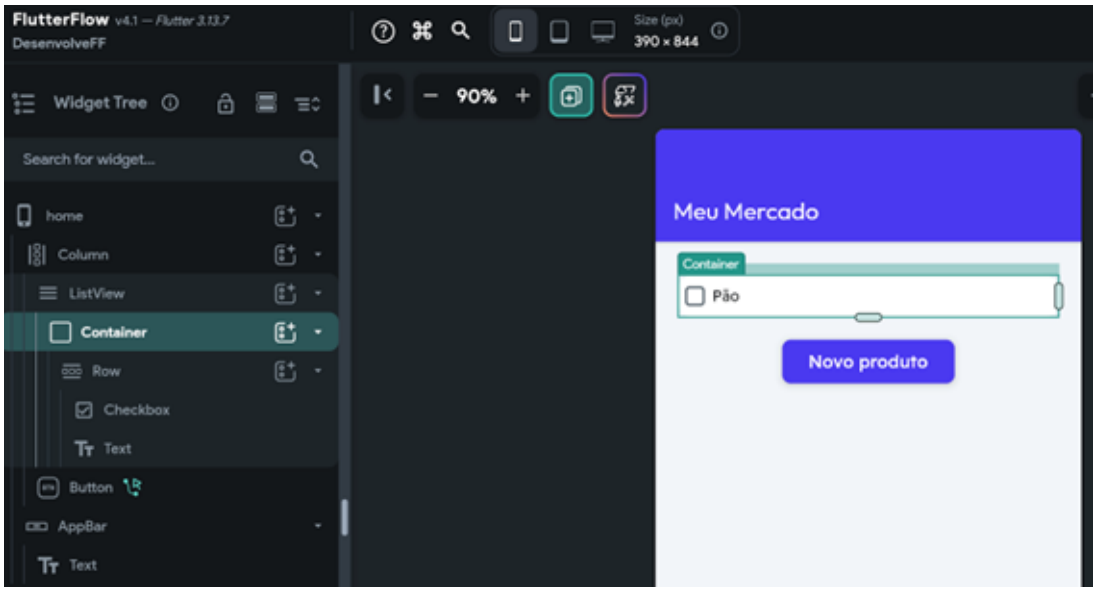
O primeiro passo é ver qual widget devemos utilizar para mostrar uma lista. No FlutterFlow esse elemento é o **ListView**, que fica em **Layout Elements**. Lembrando que você sempre pode pesquisar pelo nome do elemento para facilitar a criação da interface.



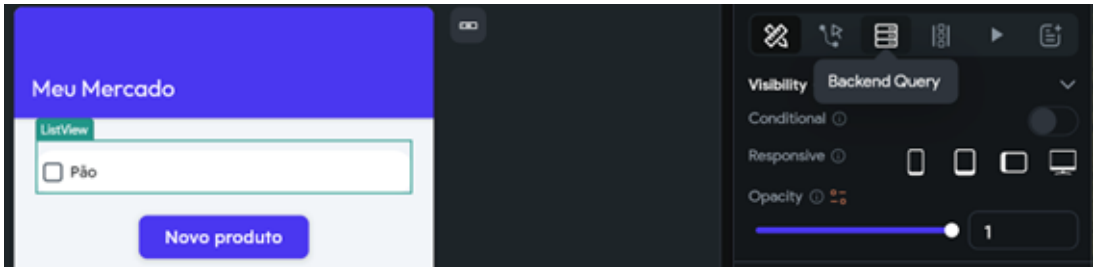
Arraste o widget para a tela.

Repare que ele está acima do nosso texto. Agora repare como ele está na nossa árvore de widgets. Ele está acima do nosso container. O que queremos é que o container esteja dentro do ListView. Para isso, basta arrastar o container para dentro dele nessa nossa árvore.

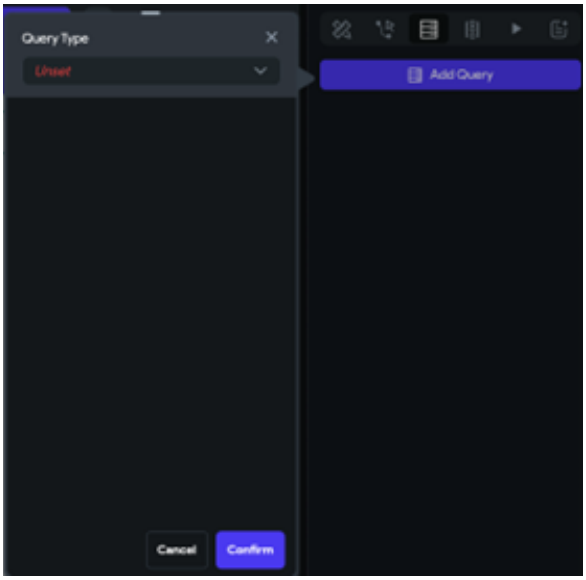
O segundo container que criamos foi criado de maneira estática, só para demonstrarmos. Assim, pode ser apagado. Nossa árvore e nossa interface ficam então:

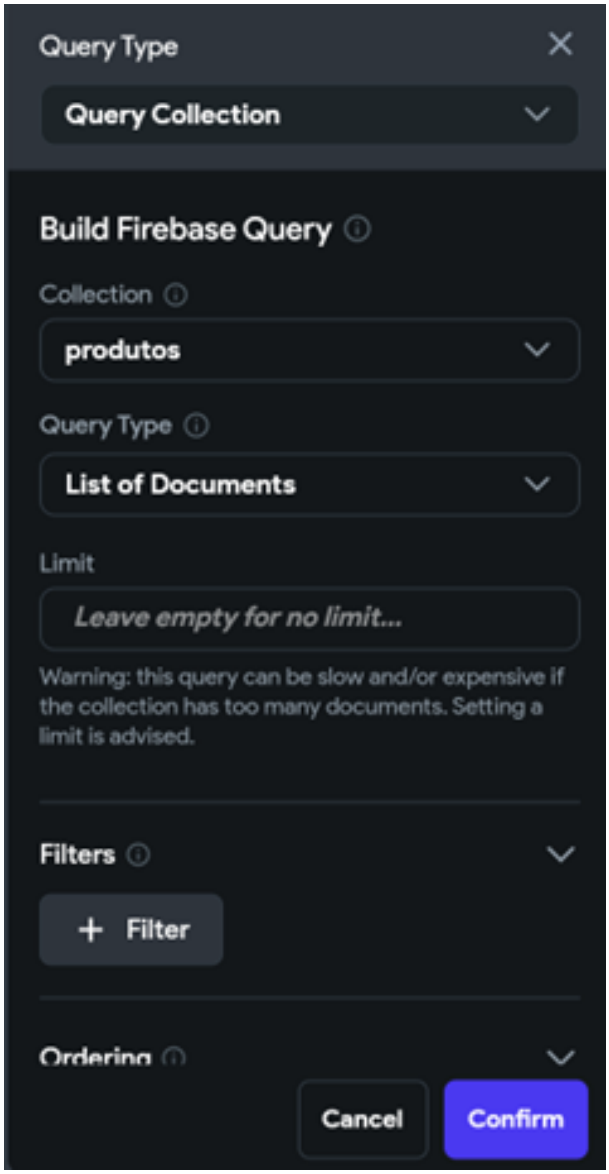


Agora vamos informar ao **ListView** que ele irá mostrar dados do nosso banco de dados. Clique em nosso ListView, em suas propriedades vá em **Backend Query**. Query é uma solicitação de informações a um banco de dados.



Assim, vamos criar uma solicitação **(query)** para o nosso banco de dados. Clique então em **Add Query**.

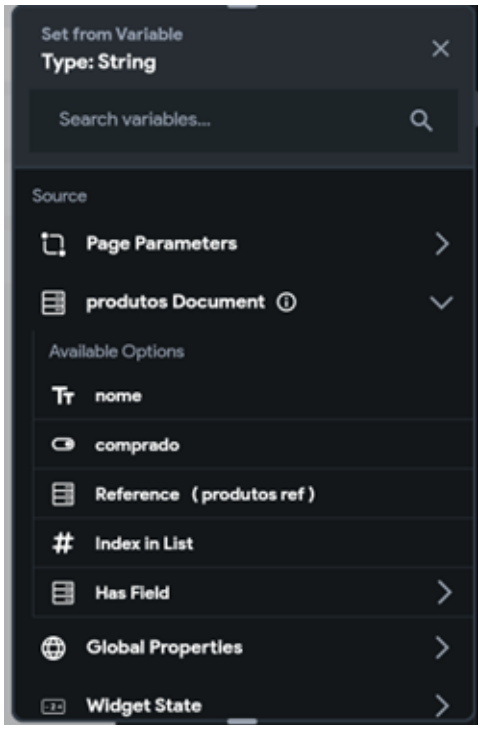




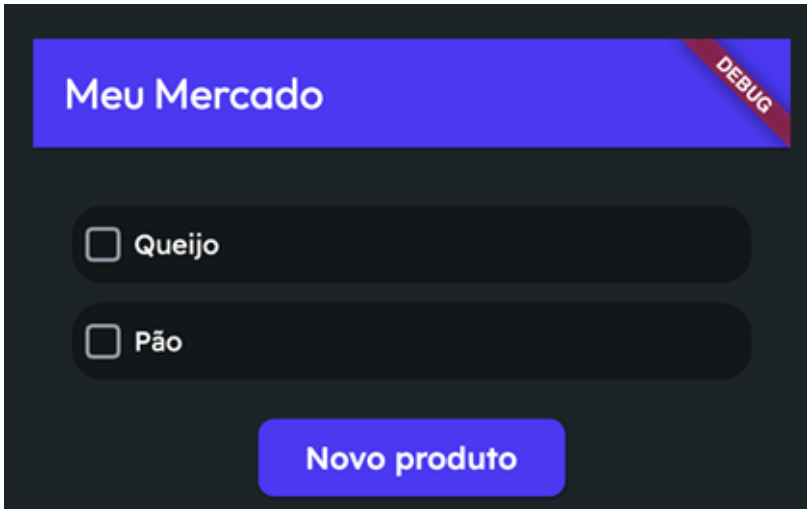
Nós queremos trazer uma coleção para nossa tela, então o tipo da nossa query será **Query Collection** e vamos selecionar nossa coleção **produtos**.

Agora é só confirmar!

Repare que o FlutterFlow já mostra que pode trazer várias informações para a nossa lista. Porém, nós inicialmente deixamos nosso texto estático. Agora, temos que transformá-lo em texto dinâmico. Clique no **Text** e em suas propriedades clique no botão laranja. Dessa forma, vamos trazer para esse texto uma variável, um valor dinâmico.

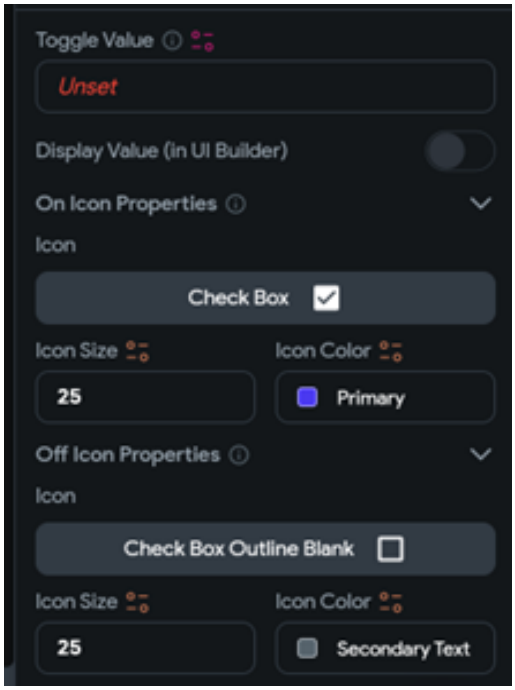


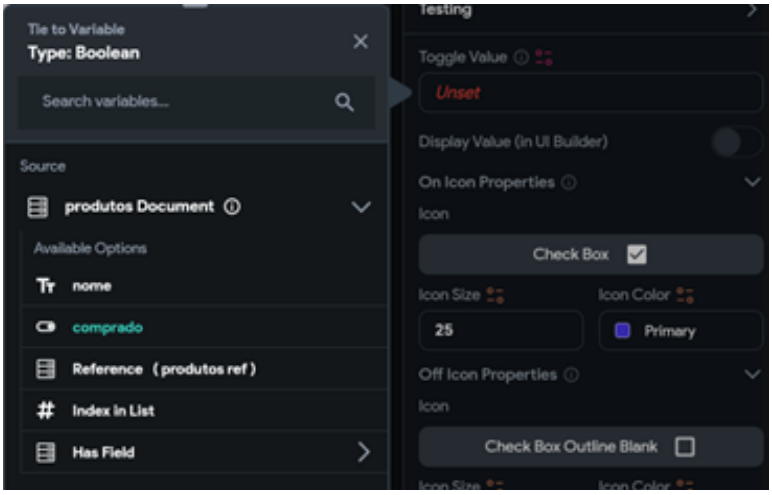
Executando agora nosso aplicativo, já conseguimos ver nossa lista de mercado. Teste à vontade criando mais produtos para ver a nossa lista funcionando.



O que falta agora é tornar dinâmico também a nossa **Checkbox**. Porém, por enquanto esse widget ainda não interage bem com o banco de dados. Vamos então apagá-la e trocá-la pelo elemento **ToggleIcon**. Você irá reparar que ele tem a mesma forma da checkbox.

Agora nas propriedades, vamos definir o **Toggle Value**. Assim, ele irá puxar essa informação do banco de dados. Mas por enquanto não estamos alterando ela. Para alterá-la precisamos **atualizar** o valor de comprado no banco.

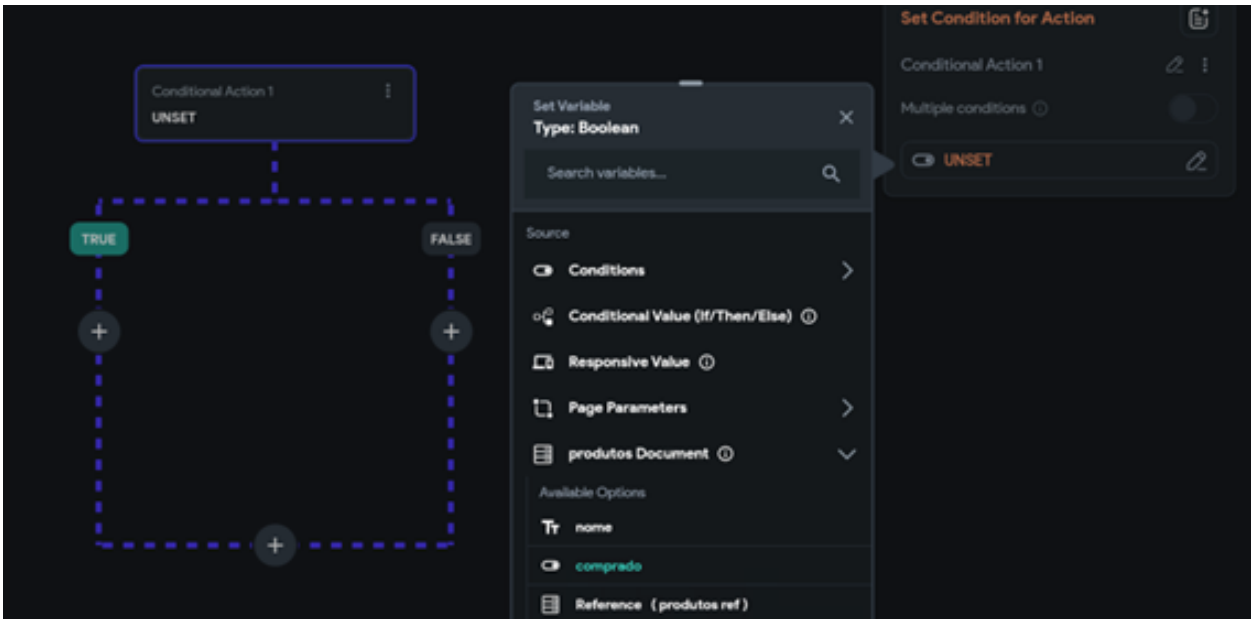




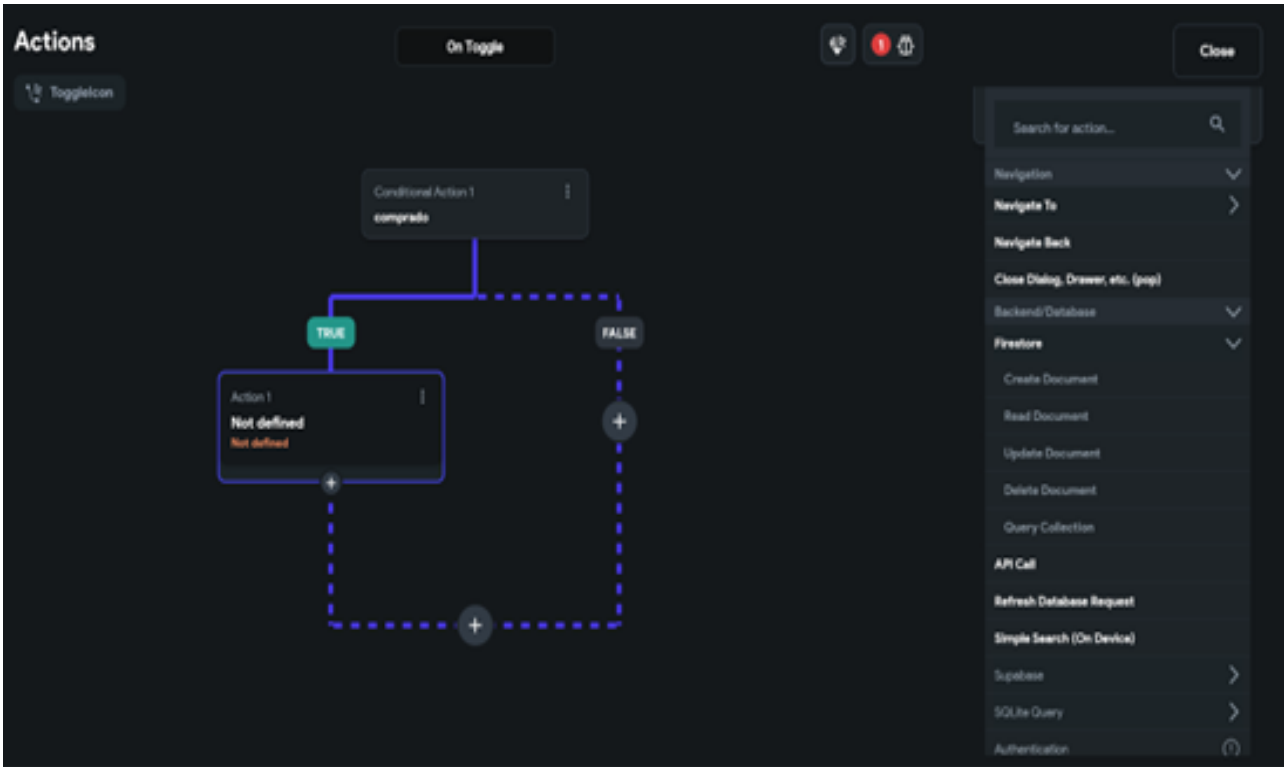
CRUD - Update

Nossa primeira função de atualização será alterar o valor de **comprado**. Para isso, vamos criar uma ação para o **ToggleIcon**. Nós queremos que SE comprado for falso, ele deve virar verdadeiro e SE comprado for verdadeiro, ele deve virar falso.

Uma das formas de fazer isso é adicionando uma ação condicional **(Add conditional action)** no fluxo de ações.

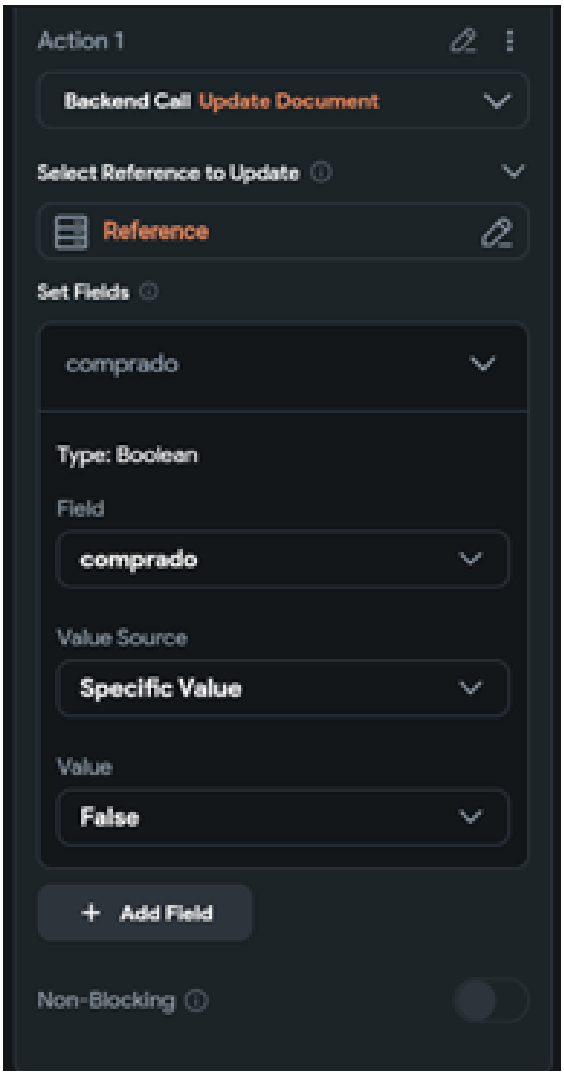


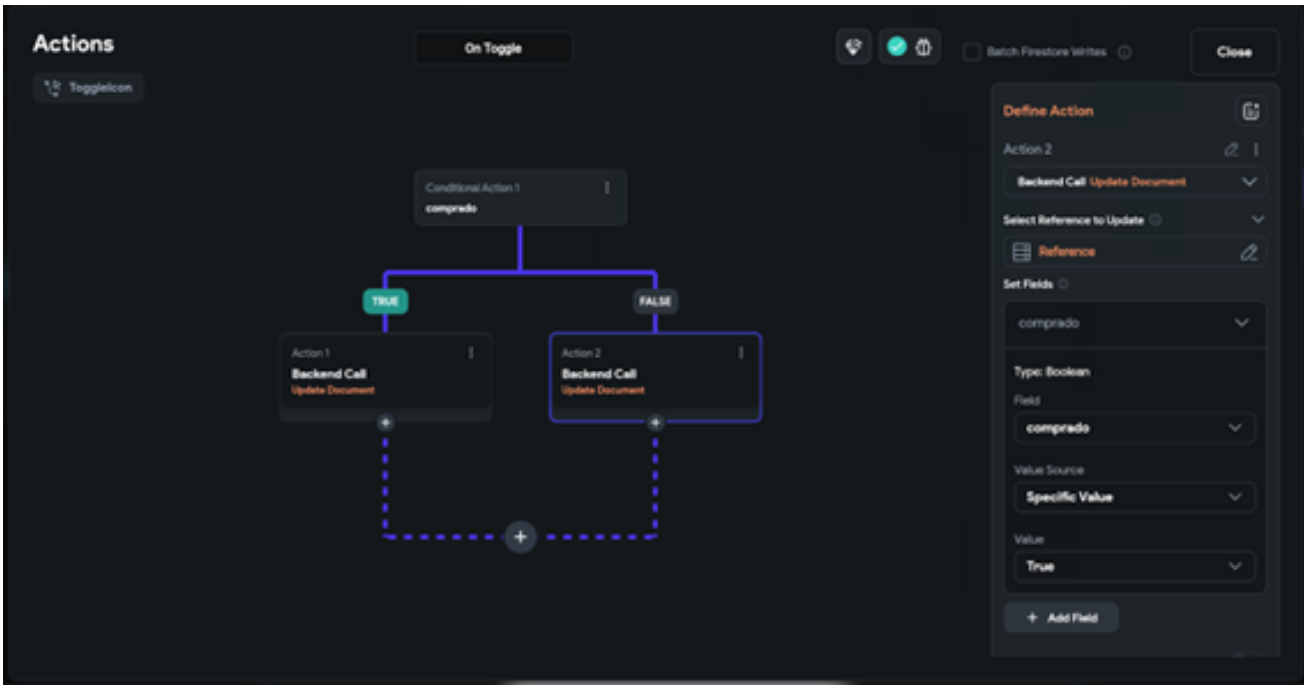
Agora basta criar a ação caso o valor de comprado seja **True** e caso ele seja **False**. Clique no **+** logo abaixo de cada valor desejado e vamos adicionar a ação de **Update document**.



Assim adicionamos a referência ao produto e colocamos um valor fixo para o campo comprado, afinal, se true ele vira false.

E para o valor false, basta fazer o contrário.

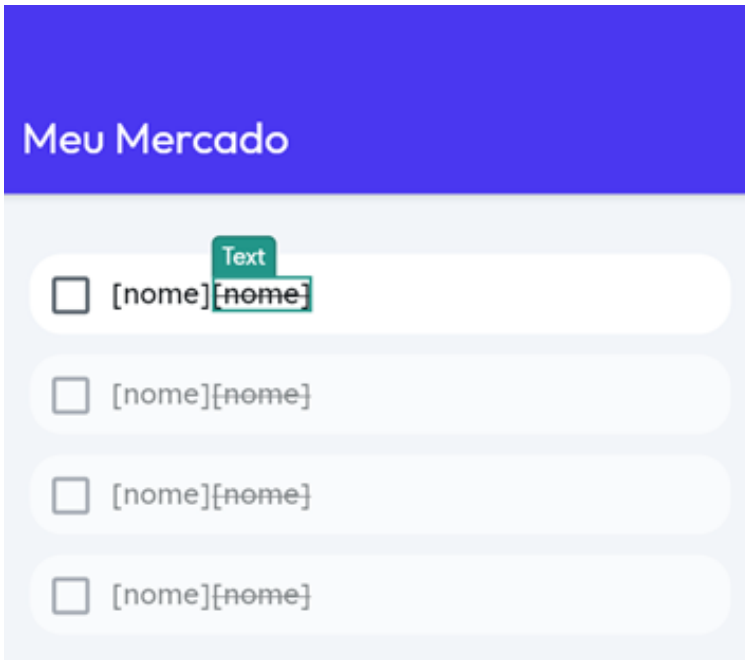




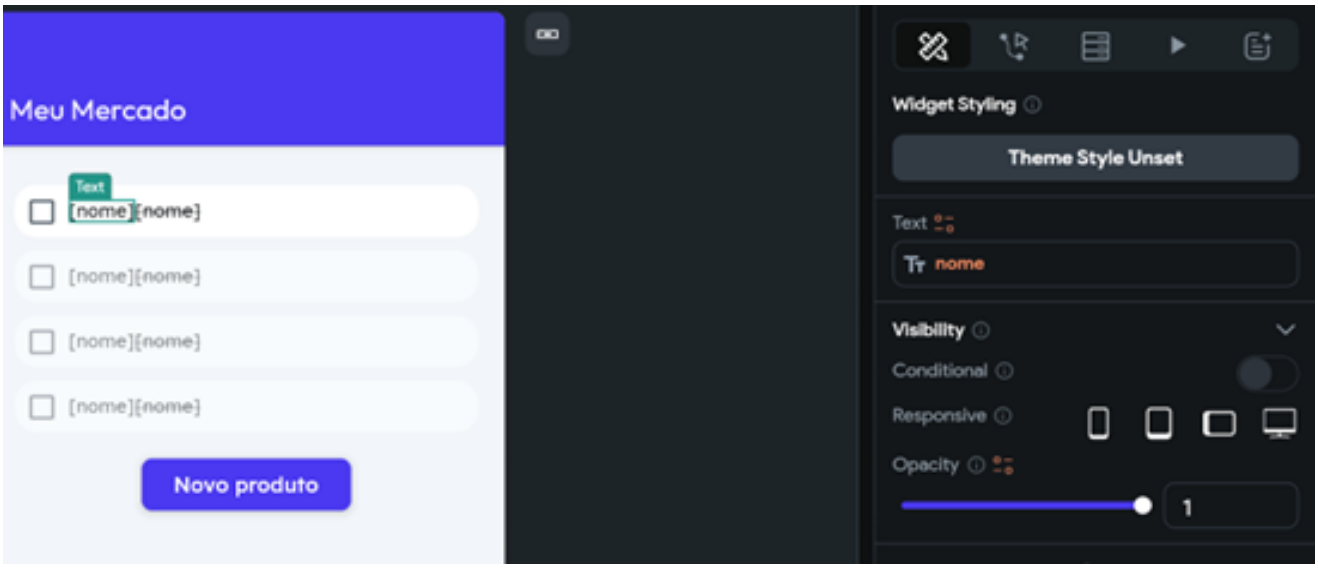
Pronto! Vamos testar!

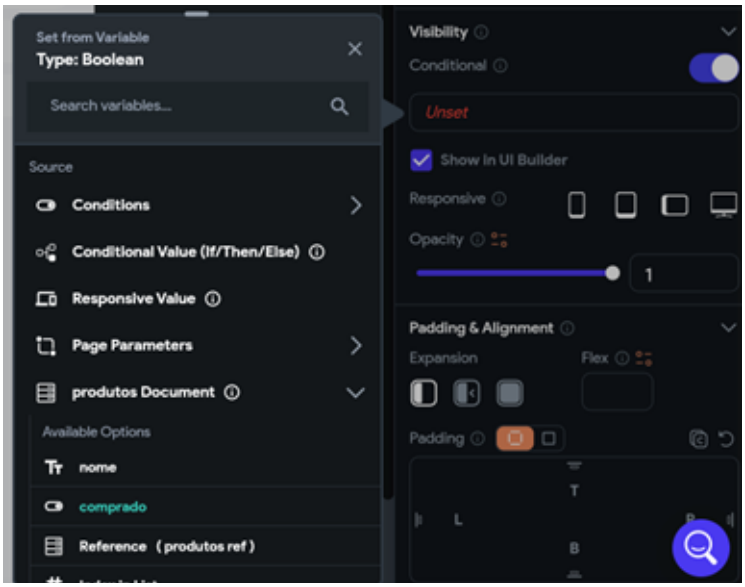
O que falta agora para nossa lista? Lembre-se que no nosso escopo queremos marcar como comprado riscando o nome do produto. Porém, no FlutterFlow não é possível tornar a estilização dinâmica, somente o tamanho da fonte e a cor.

Mas podemos criar uma forma diferente de fazer isso. Vamos duplicar nosso texto e colocá-los com estilização diferente. Se o item não tiver sido comprado, mostramos o texto normal. Se o item tiver sido comprado, mostramos o texto riscado. Então clique com botão direito em **Text** e em seguida **Duplicate**. Altere o segundo texto para estilo **Strikethrough**.

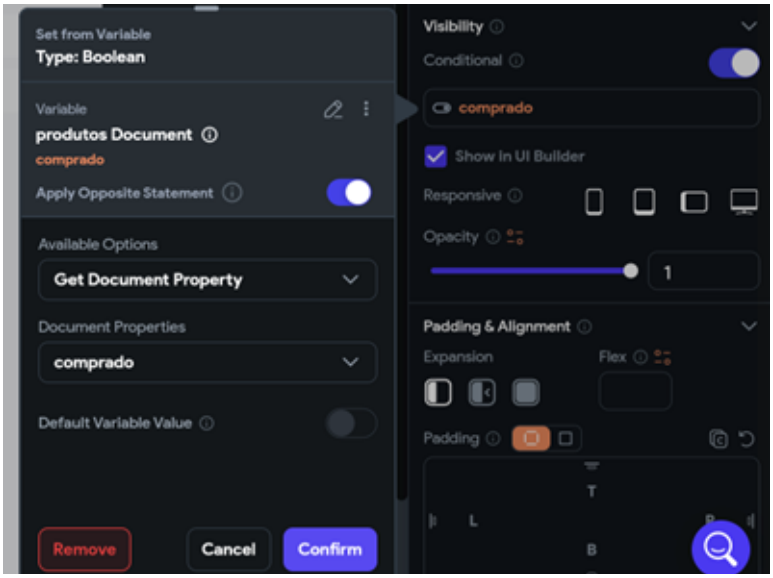


Agora vamos nas propriedades do texto e adicionar uma **visibilidade condicional**.





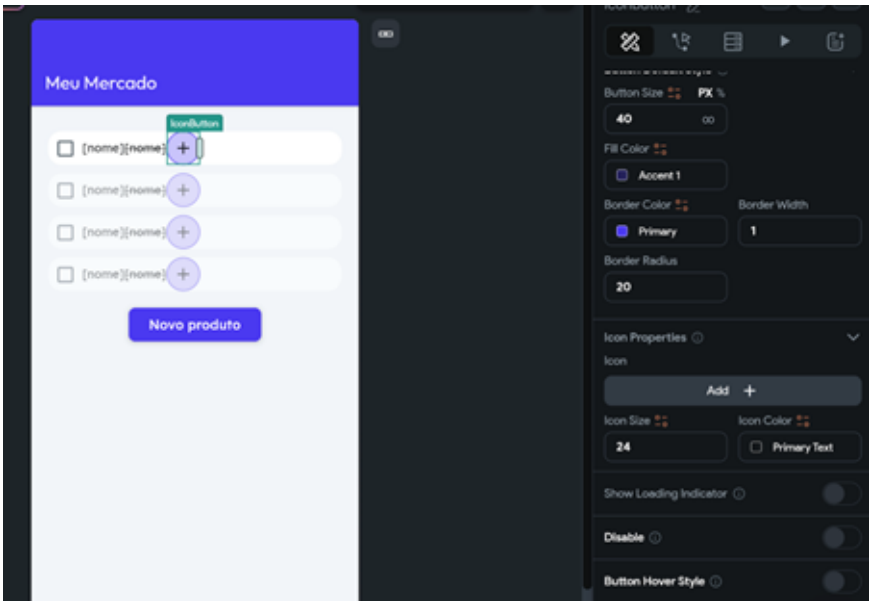
Assim pegamos o valor de comprado. Agora marque **Apply Opposite Statement**, pois mostramos o texto normal quando comprado for false.

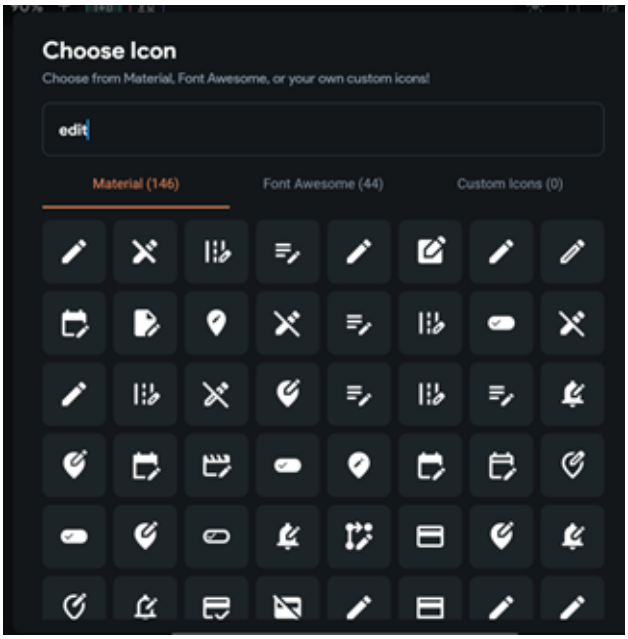


Pronto! As atualizações necessárias para a tela **home** estão feitas.

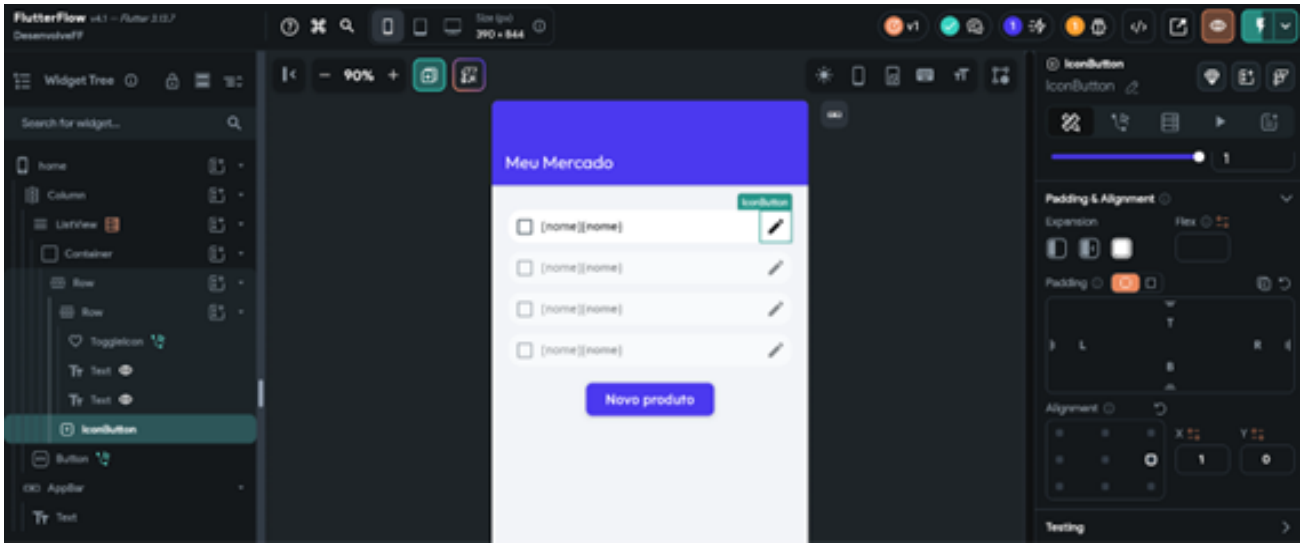
CRUD - Update da tela de edição

Vamos preparar agora as atualizações para a tela de edição. Para isso, nós vamos criar um ícone de edição para abrir a próxima tela e precisaremos passar os dados para essa nova tela. Vamos arrastar para a tela um **IconButton**. Por padrão, o FlutterFlow trás o ícone de adição, mas basta irmos ao final das propriedades, clicar em **Add +** (que é o nome do ícone) e escolher um novo. Modifique também a **Fill Color** e a **Border Color** para branco.

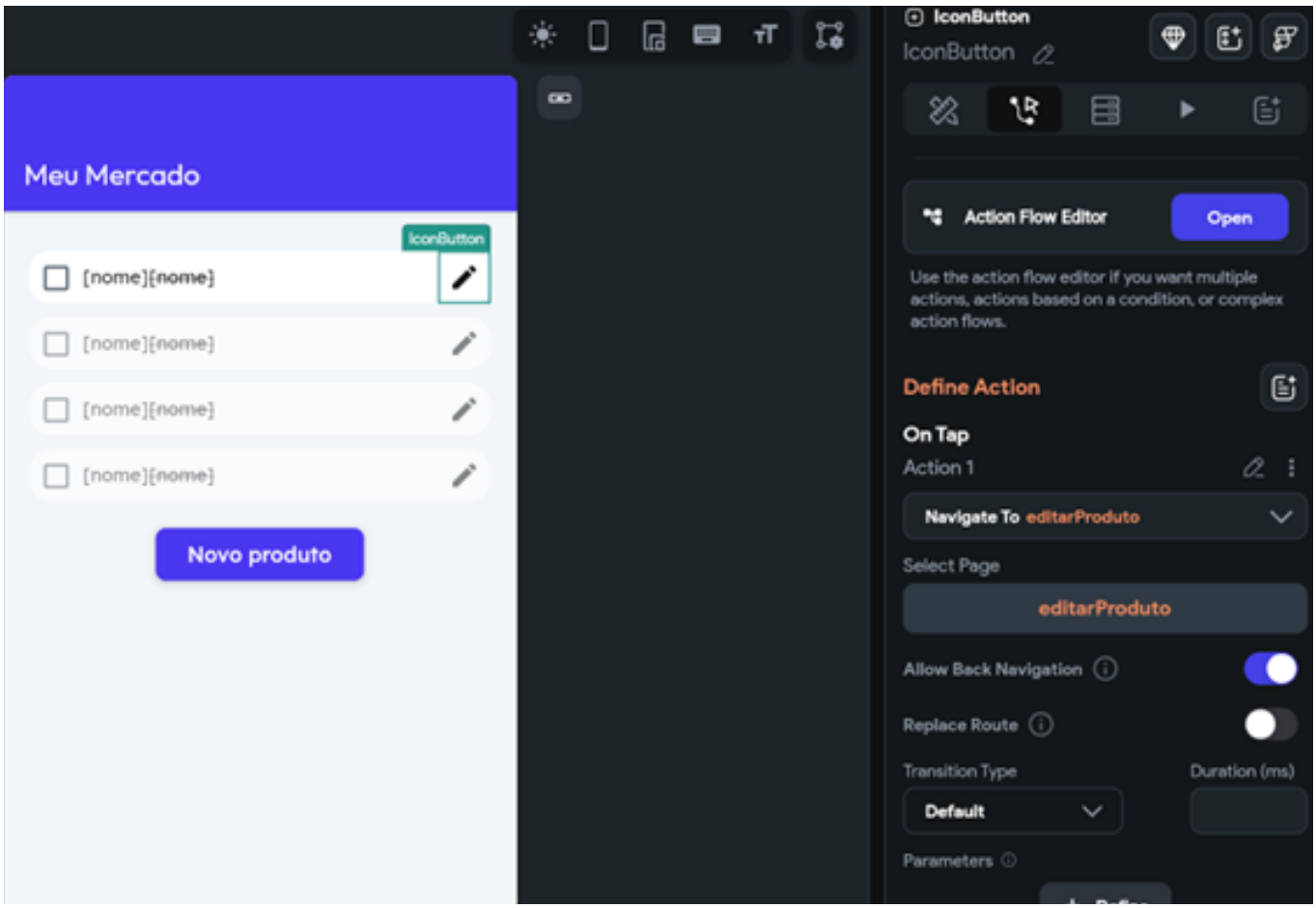




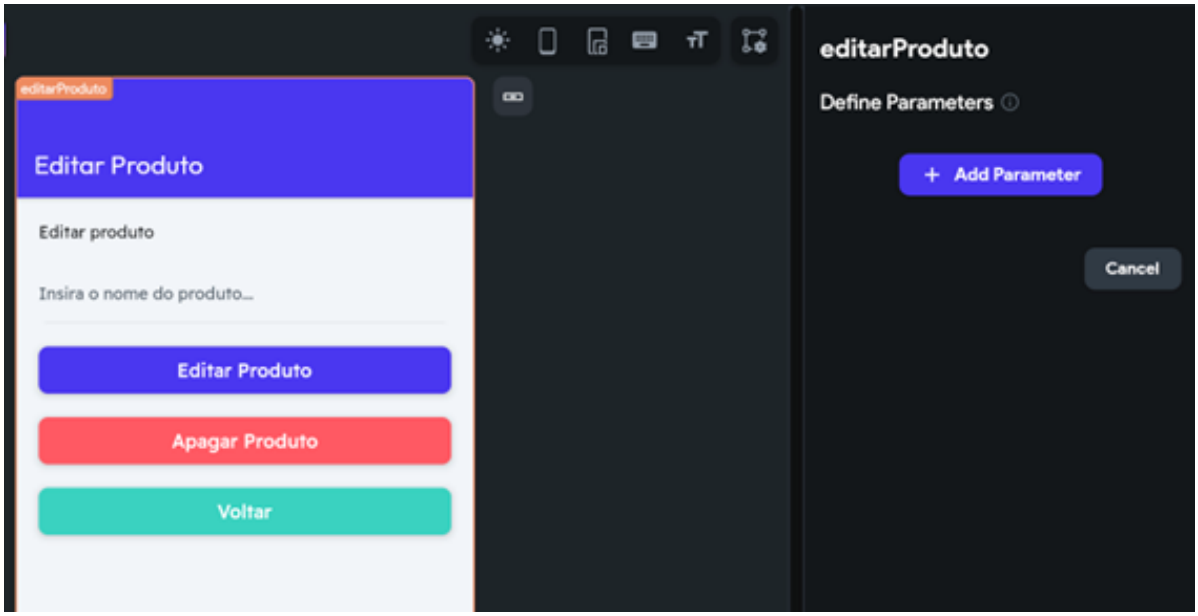
Ao executar nosso aplicativos, vamos reparar que a disposição dos elementos no nosso container não está boa. O ideal é que esse ícone de edição fique mais à direita e todos na mesma direção vertical. Para isso, vamos rearranjar nosso container com mais uma linha. Além disso, vamos alterar o alinhamento do nosso **Icon Button**.

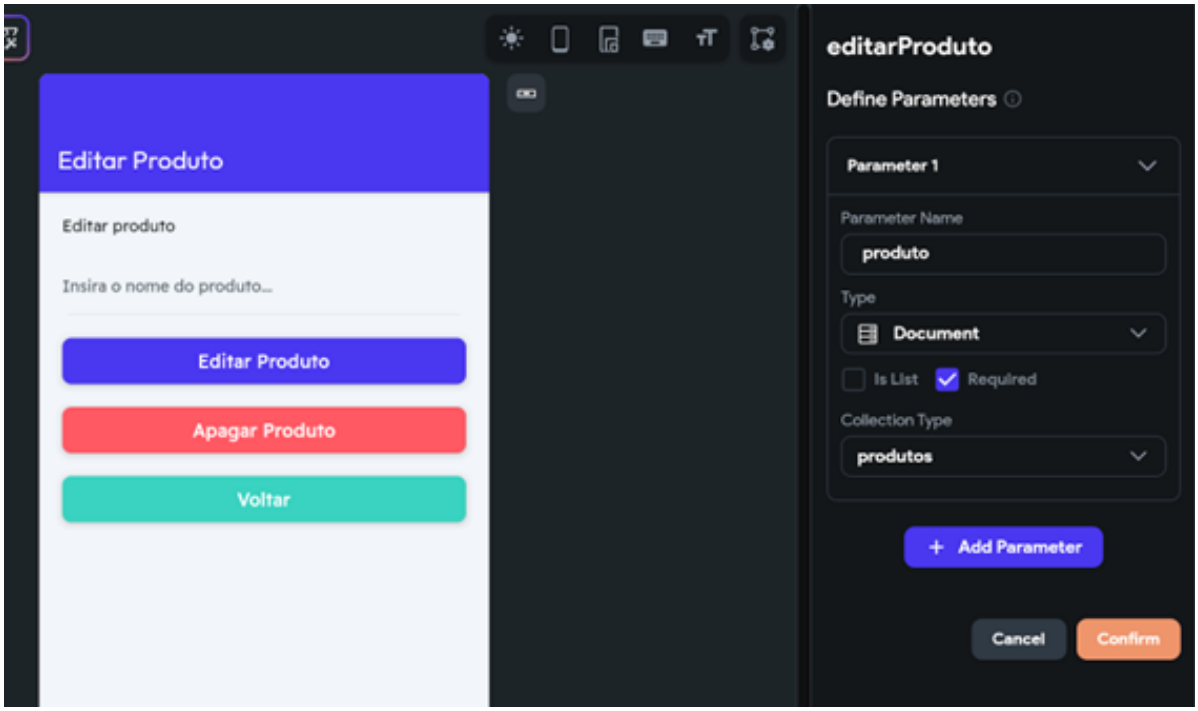


Feito isso, vamos criar a ação do nosso botão para navegar para a tela de edição.

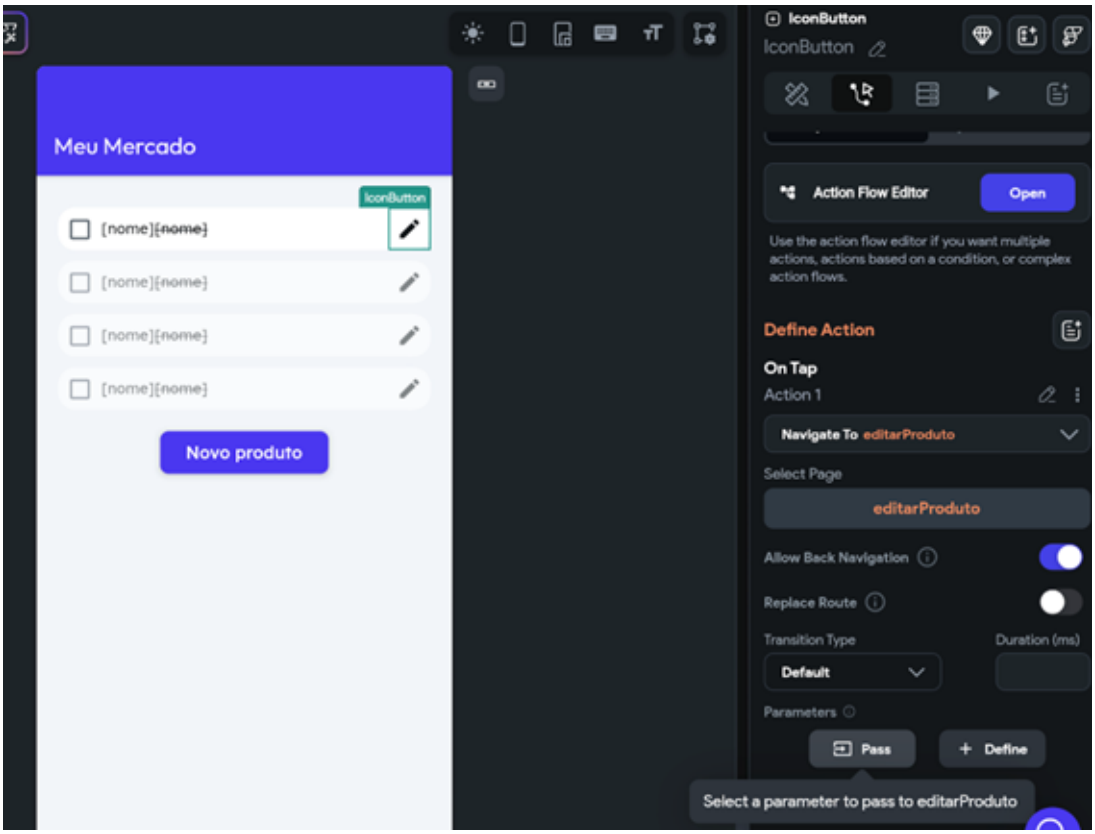


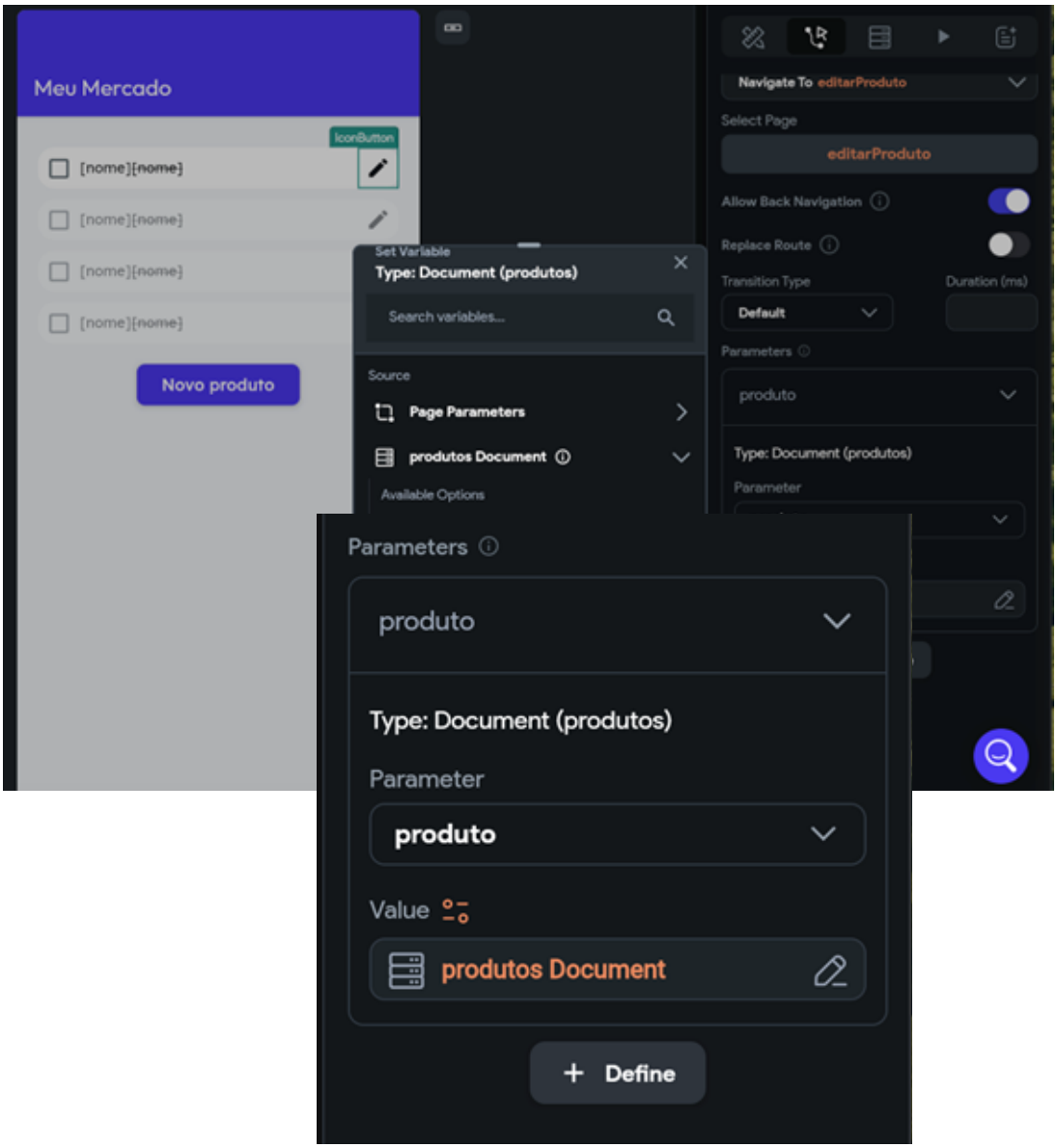
Como comentamos, precisamos passar informação para a nossa tela. Assim, vamos definir parâmetros. Clique em **Define.** Ele te jogará para a tela de edição. Clique em **Add Parameter**.

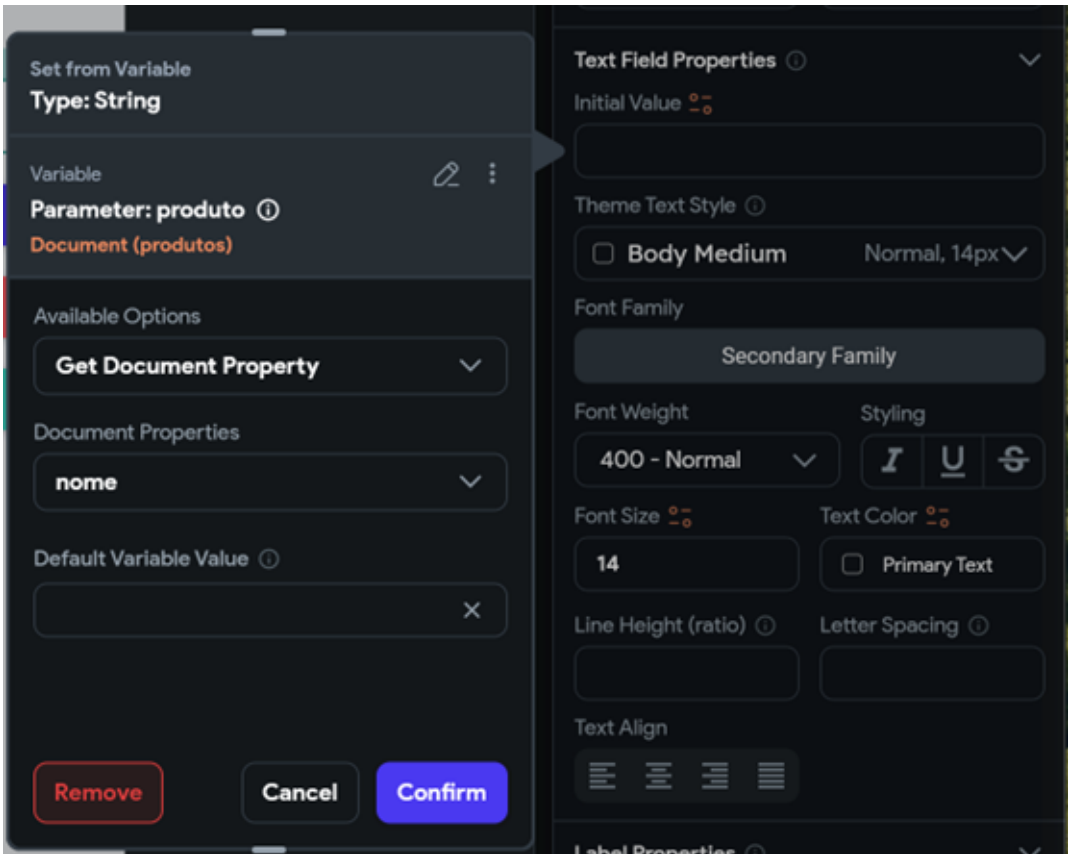




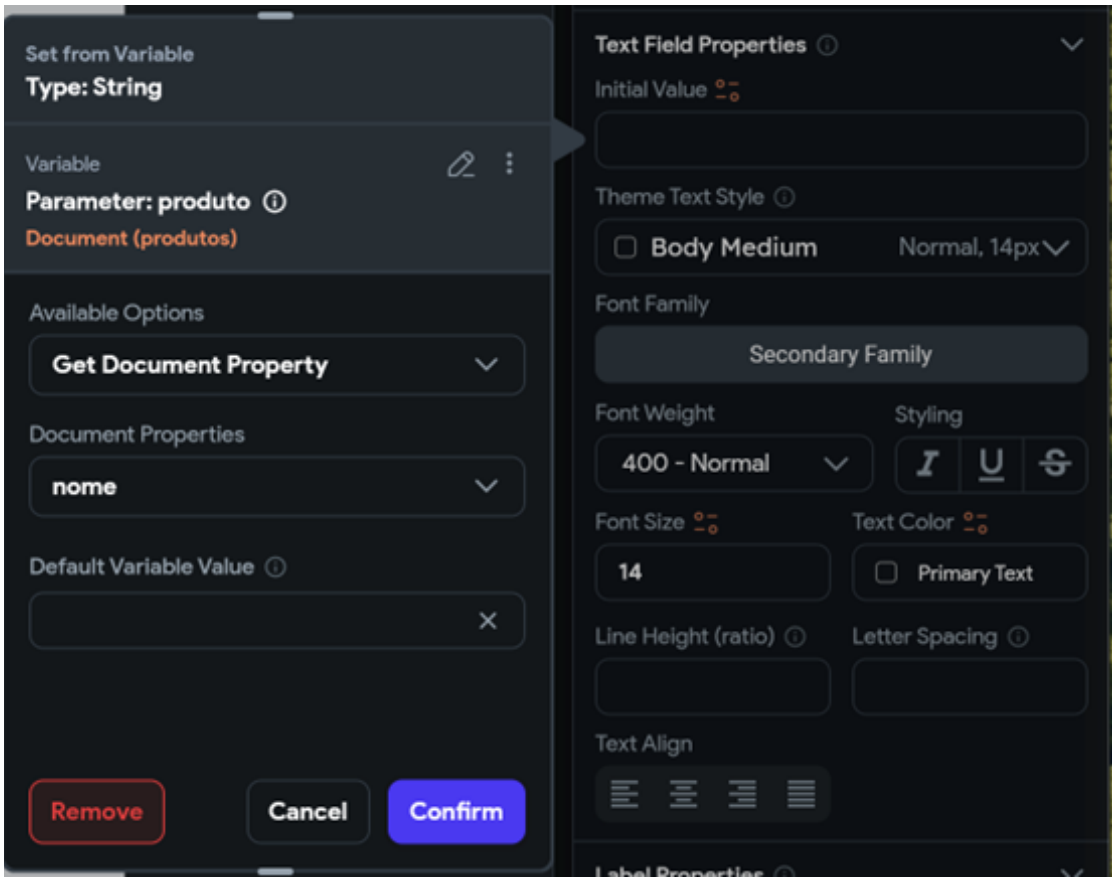
Ao clicar em **Confirm**, voltaremos para a tela home. Agora vamos terminar de configurar informando que queremos passar um documento para a nova página.


Meu Mercado
IconButton
[nome][nome]
[nome][nome]
[nome][nome]
[nome][nome]
Novo produto
IconButton
IconButton
Action Flow Editor
Open
Use the action flow editor if you want multiple actions, actions based on a condition, or complex action flows.
Define Action
On Tap
Action 1
Navigate To editarProduto
Select Page
editarProduto
Allow Back Navigation
Replace Route
Transition Type
Duration (ms)
Default
Parameters
Pass
Define
Select a parameter to pass to editarProduto

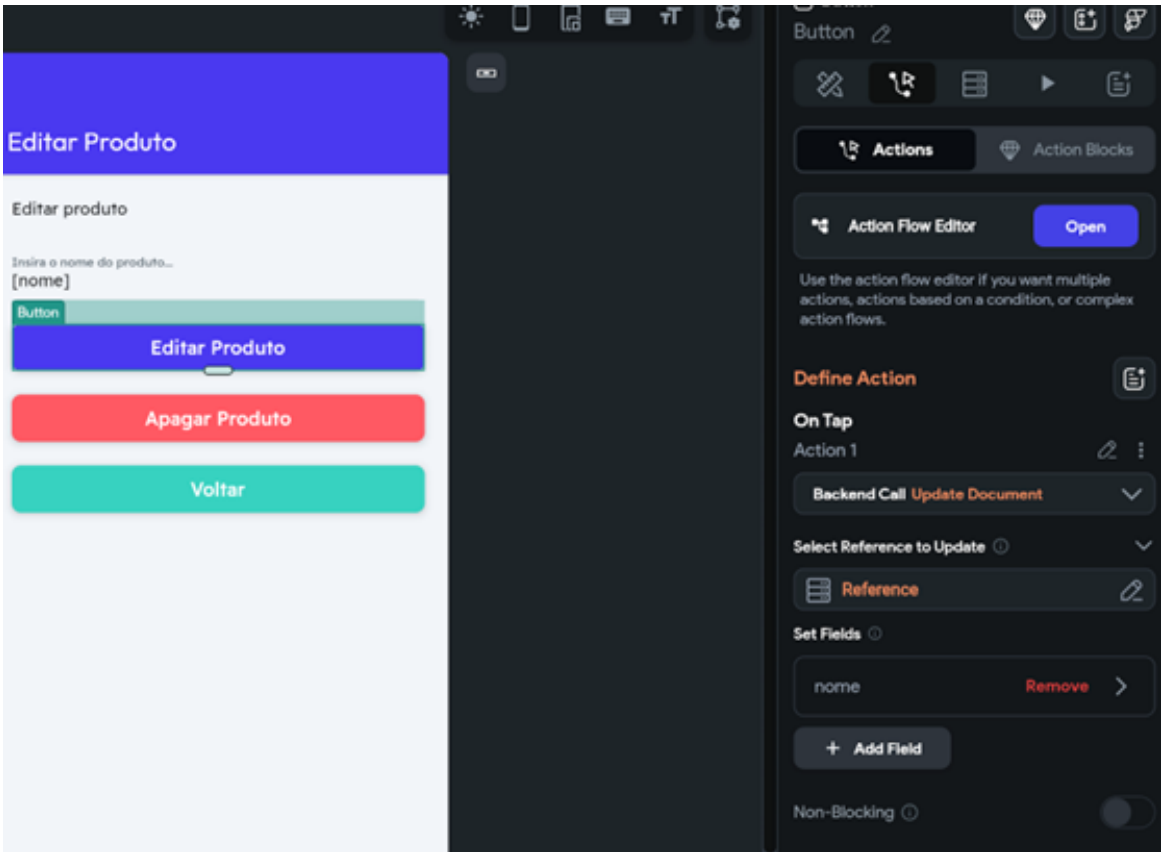

Meu Mercado
IconButton
[nome][nome]
[nome][nome]
[nome][nome]
[nome][nome]
Novo produto
Set Variable
Type: Document (produtos)
Search variables...
Source
Page Parameters
produtos Document
Available Options
Navigate To editarProduto
Select Page
editarProduto
Allow Back Navigation
Replace Route
Transition Type
Duration (ms)
Default
Parameters
produto
Type: Document (produtos)
Parameter
Parameters
produto
Type: Document (produtos)
Parameter
produto
Value
produtos Document
Define

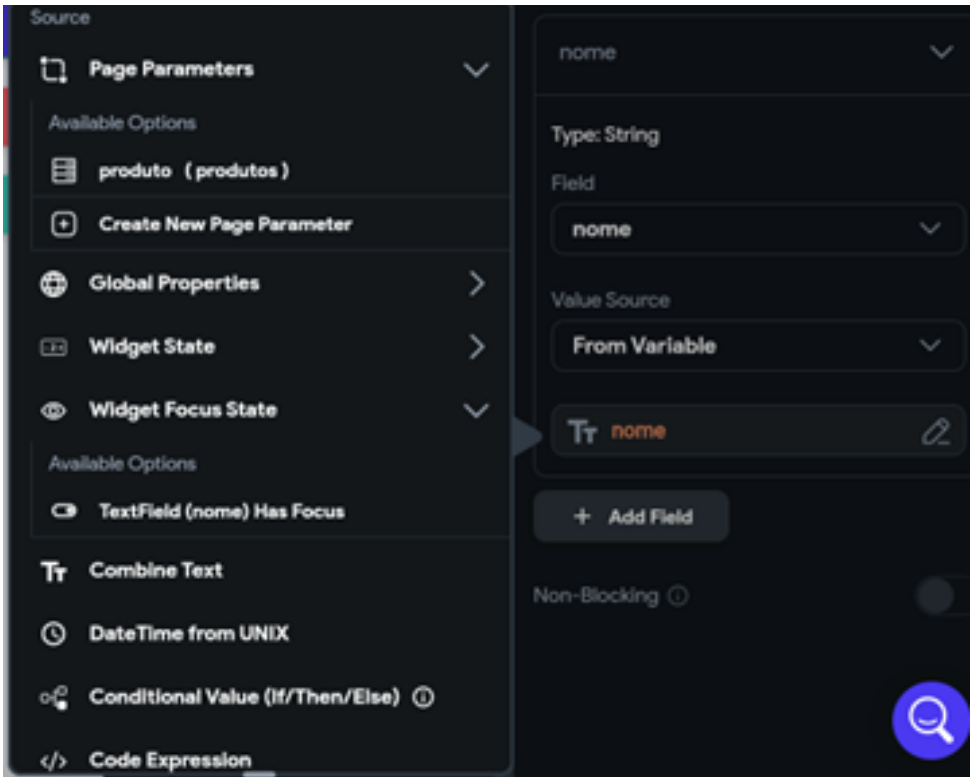


Vamos para a tela de edição para usar esse valor. Clique no **TextFiled** e em seu **Initial Value** vamos inserir um dado dinâmico.

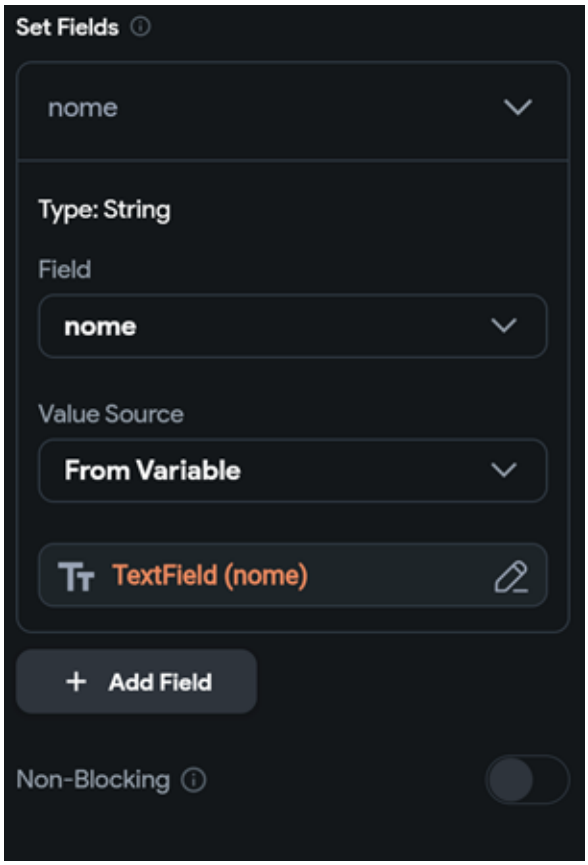


Com a navegação e a passagem de parâmetros feitas, podemos agora efetivamente editar o produto no banco de dados.



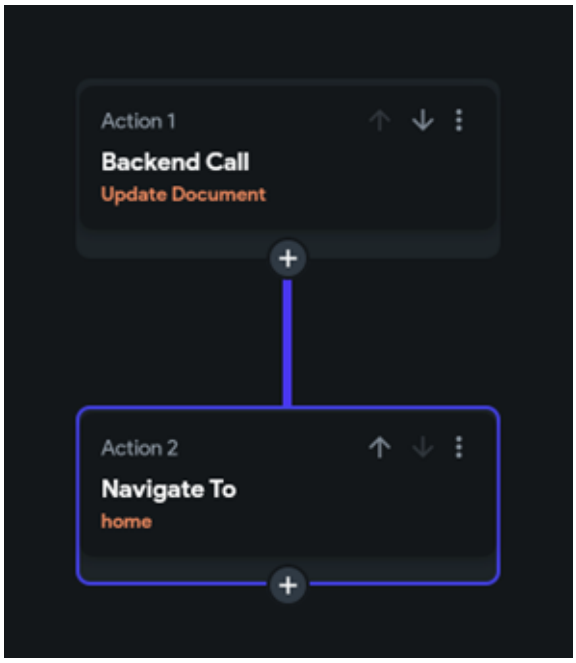


Selecione o **TextField.**

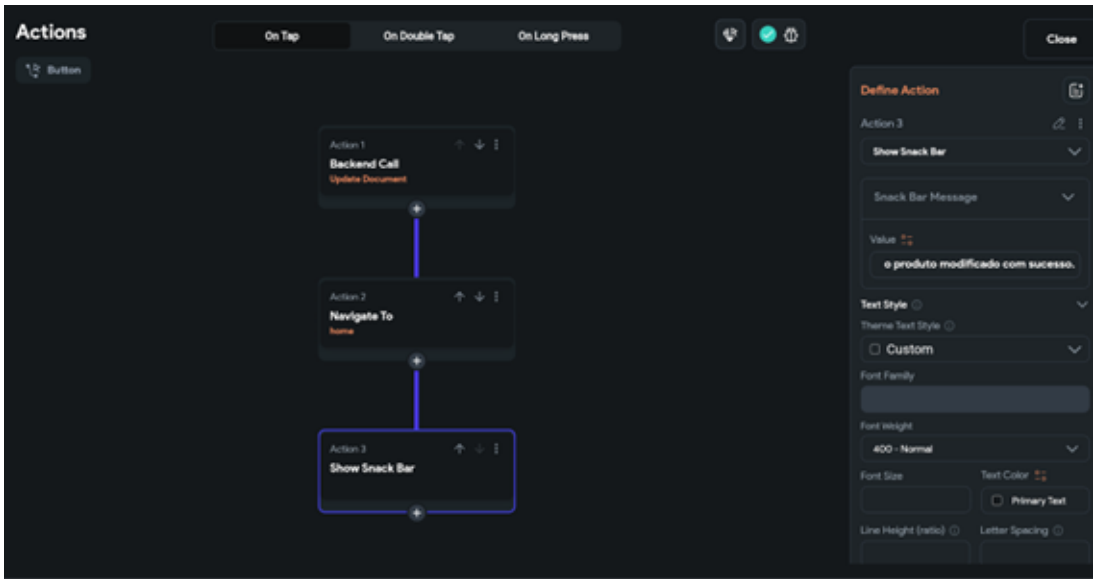


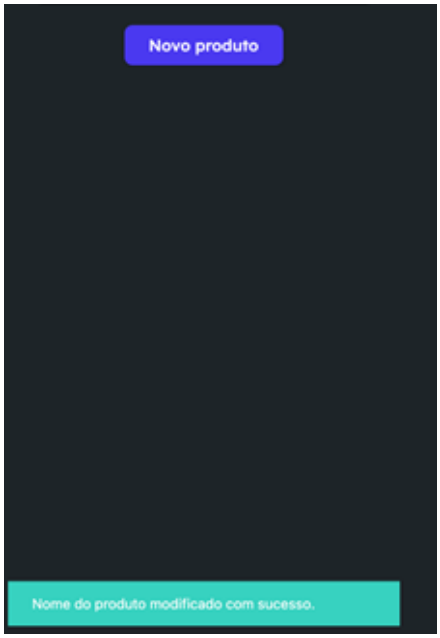
Ele fica em **Widget State.**

Assim que fizermos a atualização, vamos criar a ação de voltar para a home.



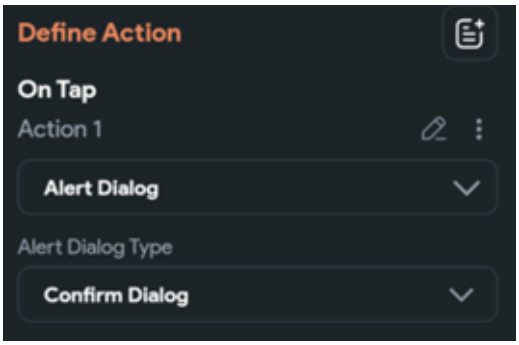
No Bubble, colocamos alertas quando o produto era adicionado e quando ele era alterado. No FlutterFlow essa ação se chama **Show Snack Bar**.

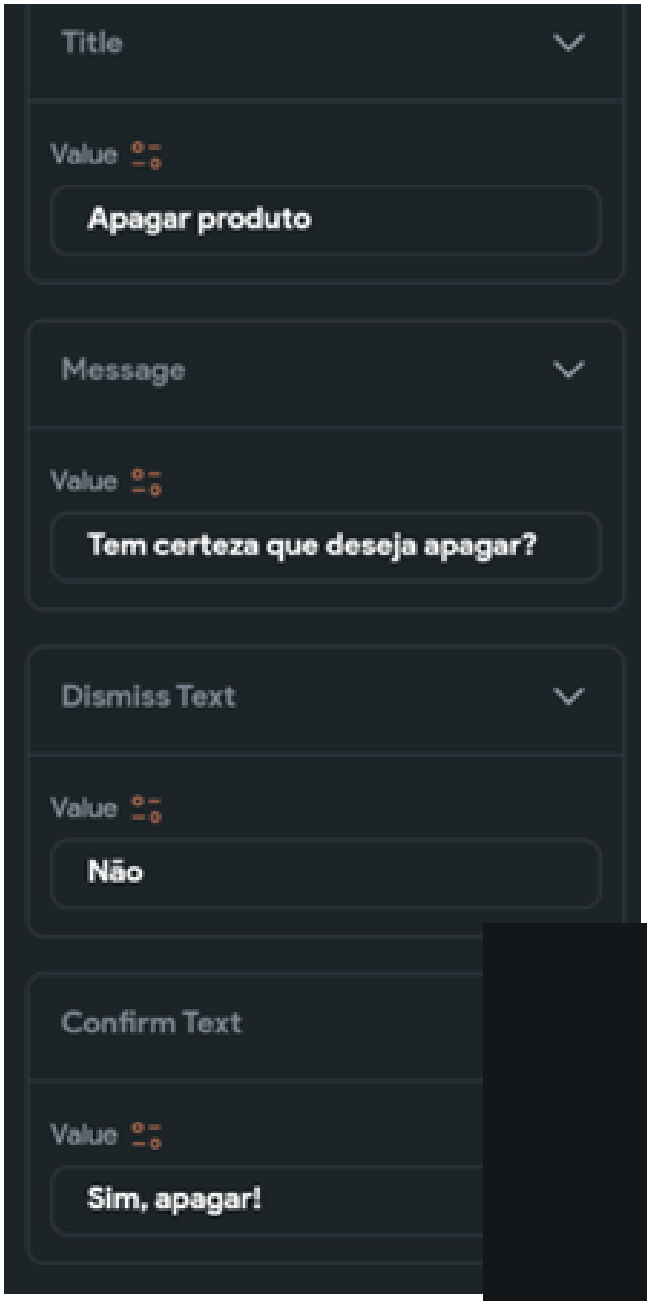




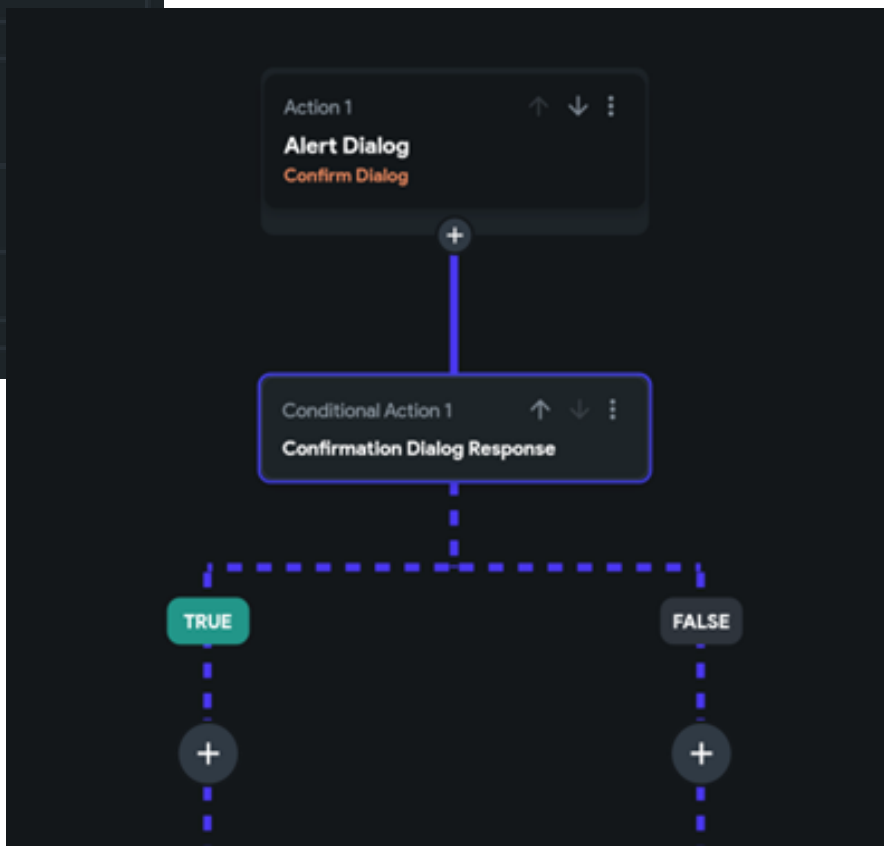
CRUD - Delete

Enfim chegamos a última letra do nosso CRUD, vamos definir a ação de apagar o produto. Como boa prática de experiência do usuário, sempre quando este vai realizar uma ação irreversível, se faz necessário pedir uma confirmação. Então nosso botão para **apagar produto** terá uma confirmação antes de executar a ação solicitada. Clique eu seu fluxo de ação e crie a ação **Alert Dialog > Confirm Dialog**.

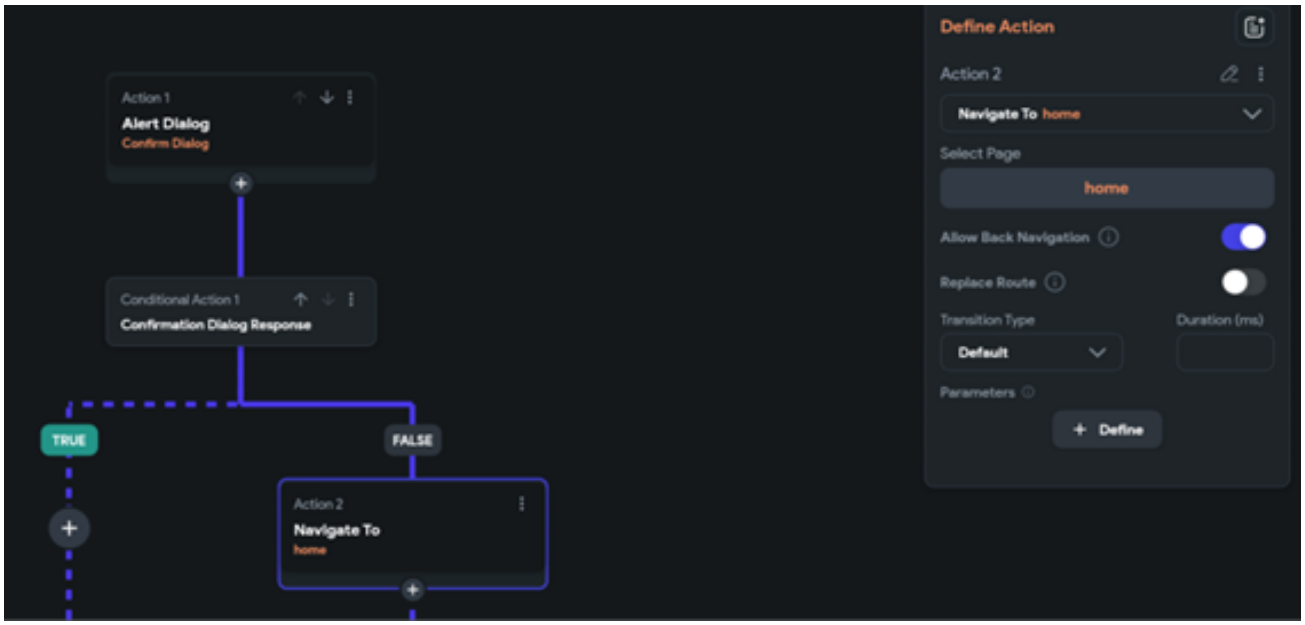




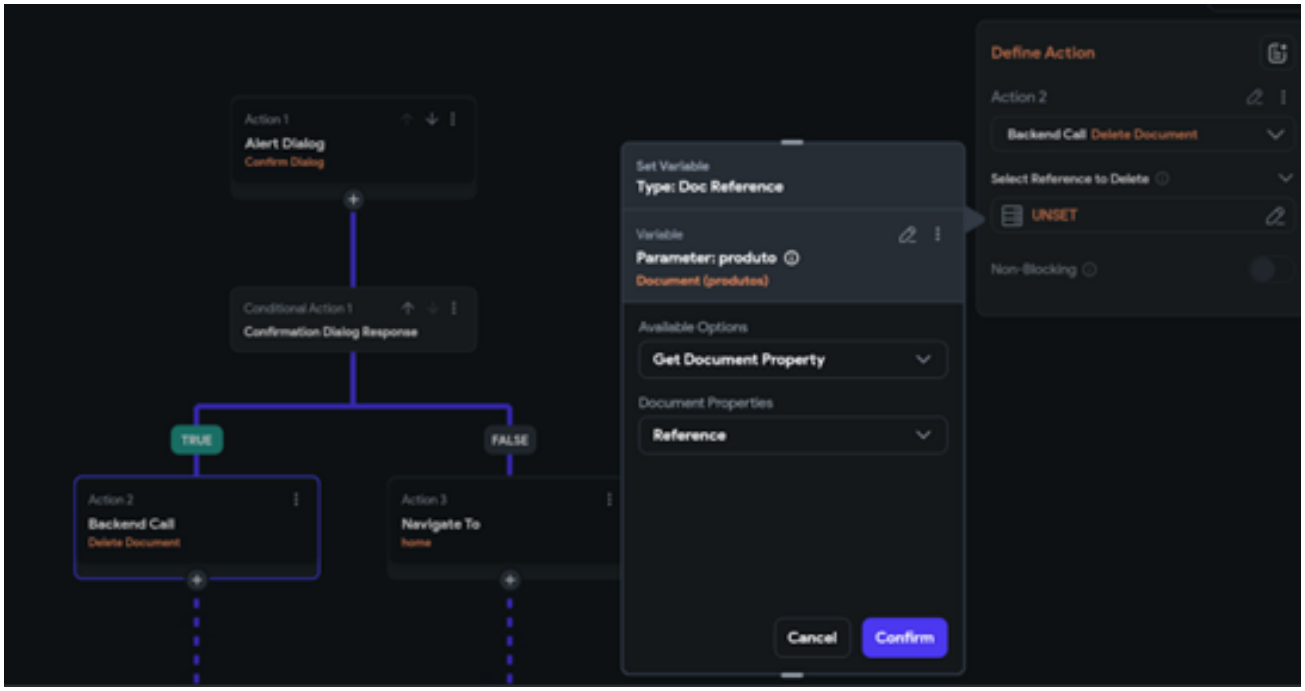
Logo depois adicionamos uma ação condicional, que o FlutterFlow já puxa como sendo a resposta da caixa de alerta.



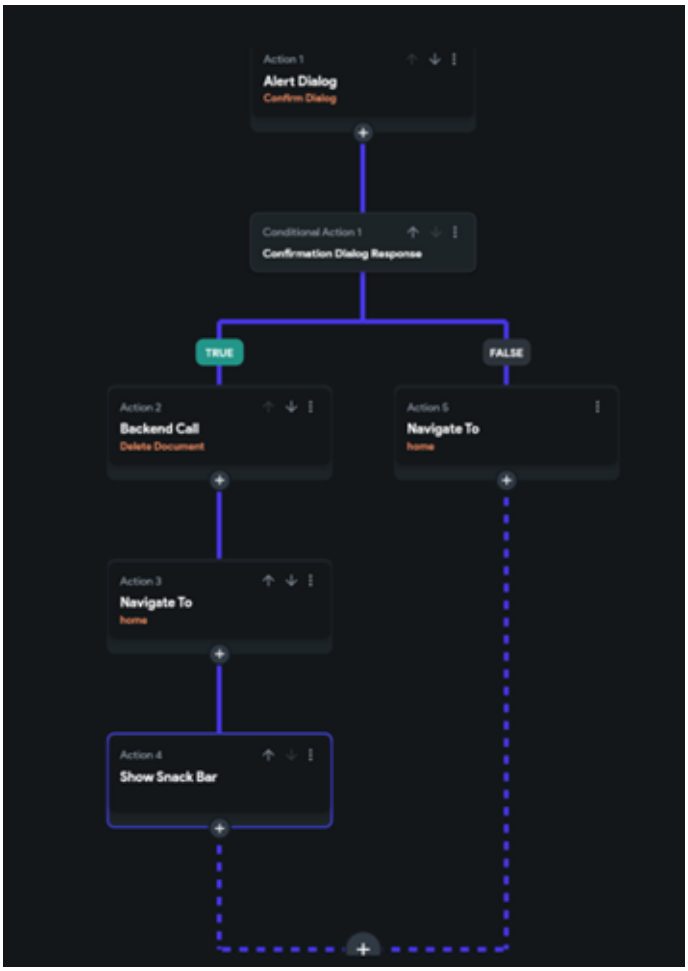
Se o usuário disser que não quer apagar, simplesmente voltamos para a home.



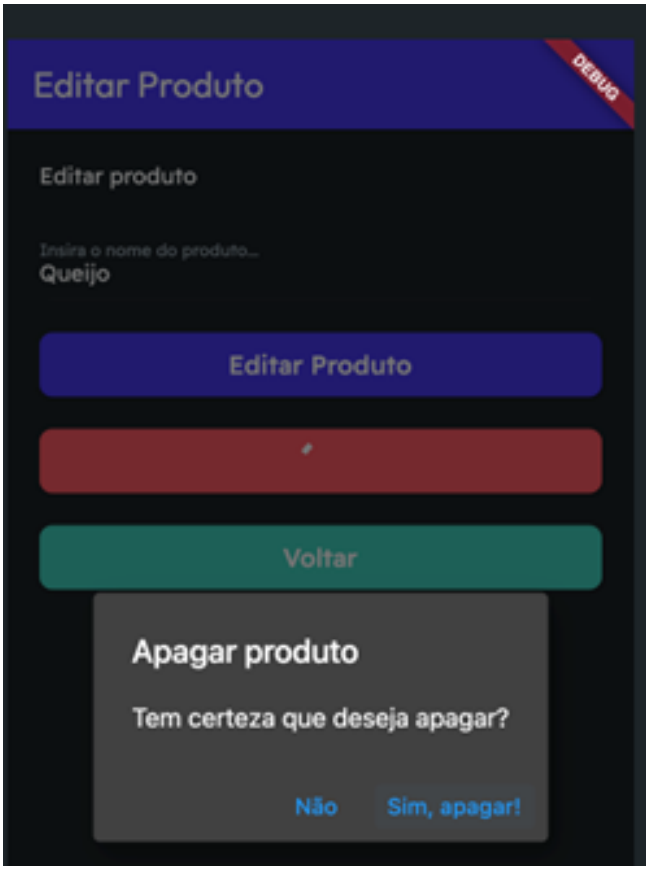
Se o usuário disser que quer apagar, vamos fazer a ação de **Delete Document**.



Depois, vamos criar uma ação para navegar de volta à home e mostrar o alerta de produto apagado.



E nosso aplicativo fica:

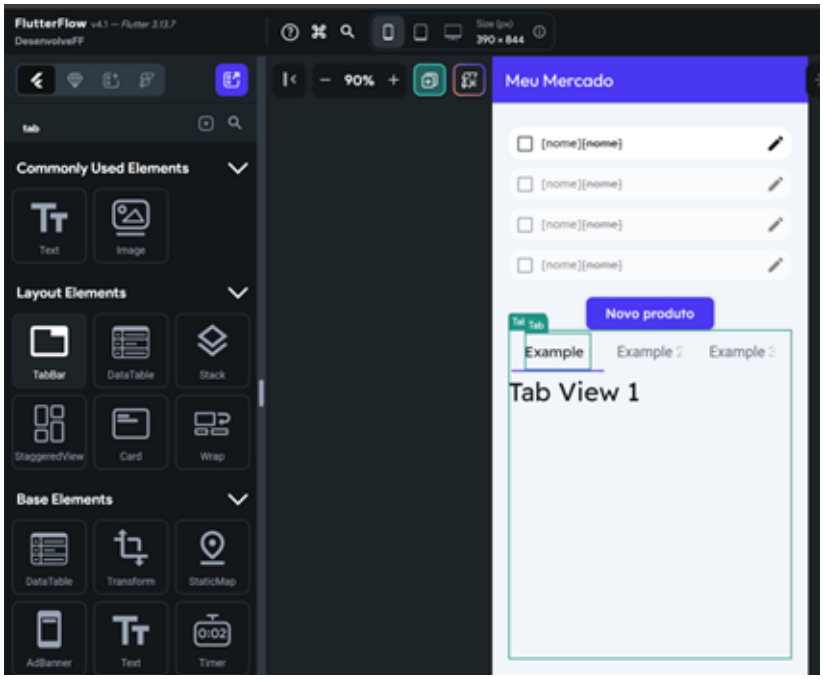


Estudo de caso: Aprimorando nosso aplicativo

Retomando o escopo que planejamos, já temos que:

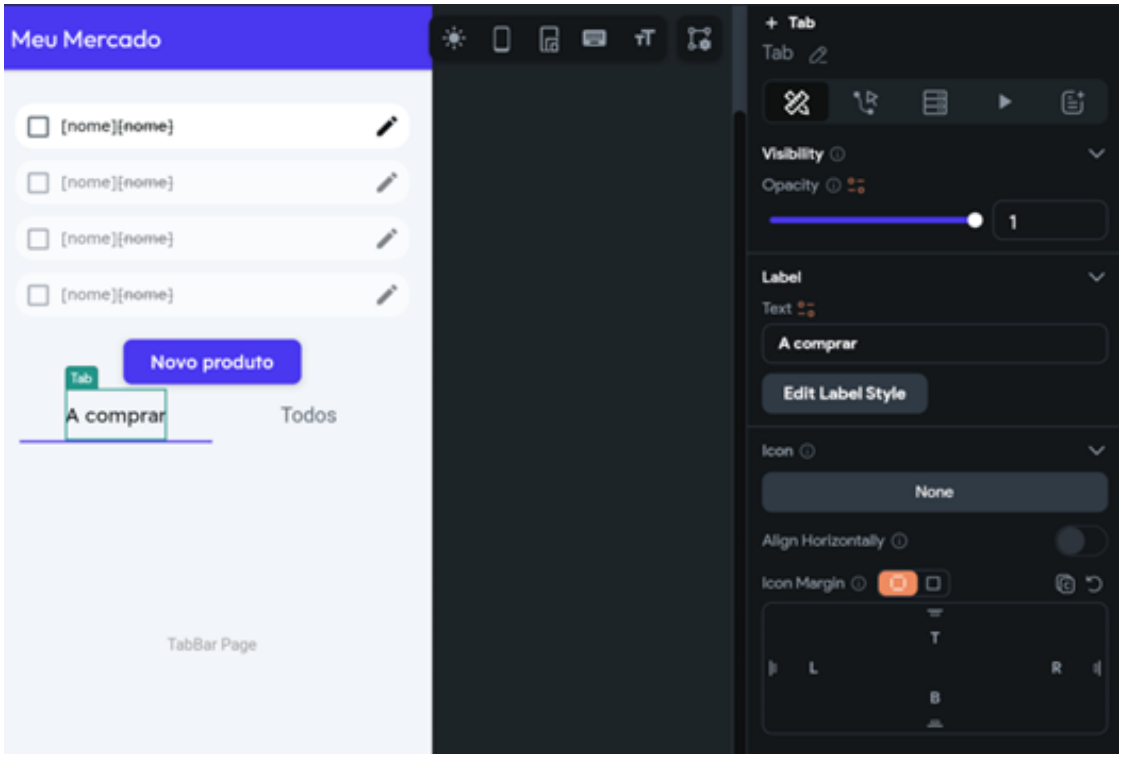
• Um campo de formulário para preencher com o nome do produto e um botão adicionar: Feito!
• Uma lista com todos os produtos já adicionados: Feito!
• Uma função para quando clicar no item, vai marcar como comprado: Feito!
• Uma função para quando clicar no item já comprado, desmarcá-lo: Feito!

Agora é a hora dos nossos aprimoramentos. Vamos colocar um elemento na tela que separa os elementos a comprar e todos: **TabBar**.

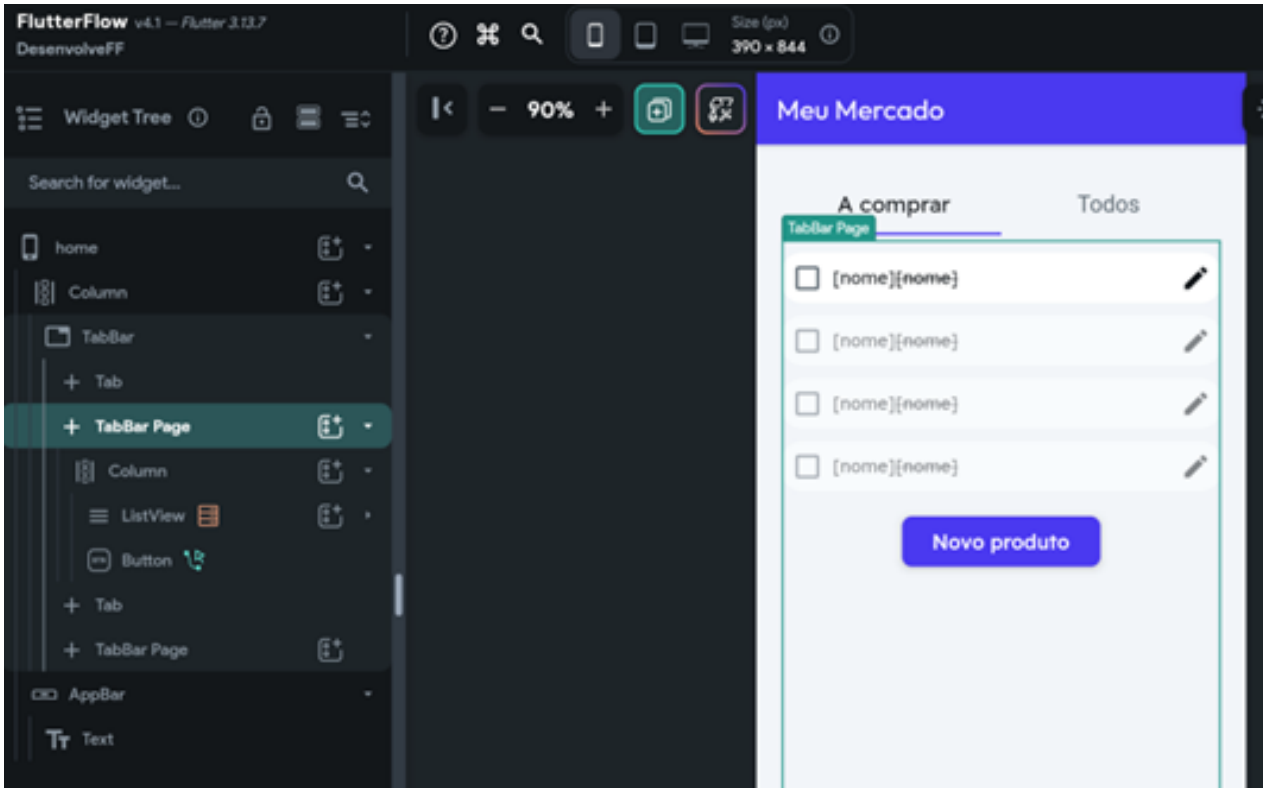


A TabBar no FlutterFlow é um widget usado para implementar uma barra de guias na interface do usuário de um aplicativo. Ele permite que os usuários alternem entre diferentes telas ou seções de conteúdo, geralmente organizadas horizontalmente na parte superior da tela. Cada guia na TabBar representa uma categoria específica de conteúdo ou funcionalidade dentro do aplicativo. Ao tocar em uma guia, o conteúdo correspondente é exibido na tela principal do aplicativo. A TabBar é uma maneira eficaz de organizar e navegar entre várias partes de um aplicativo.

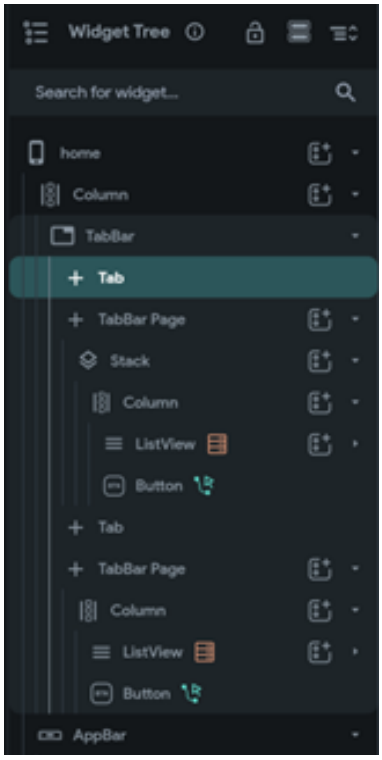
Vamos apagar a terceira aba criada e deixar só as duas primeiras. Altere o nome das abas e pode apagar o texto que vem por padrão em cada uma delas.



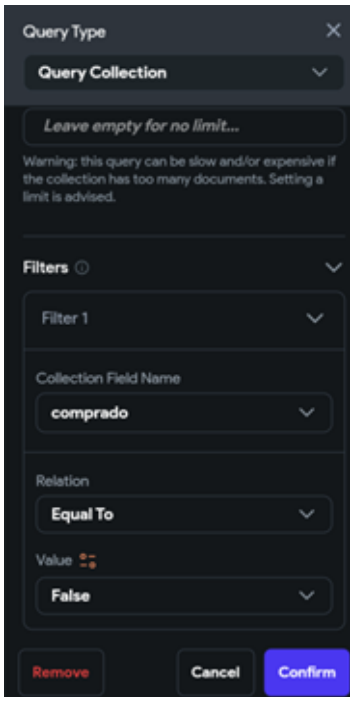
Agora vamos arrastar os widgets que criamos para dentro na **TabBar**.



Agora duplique esses elementos e arraste também para a segunda aba. Nossa árvore de widgets ficará:

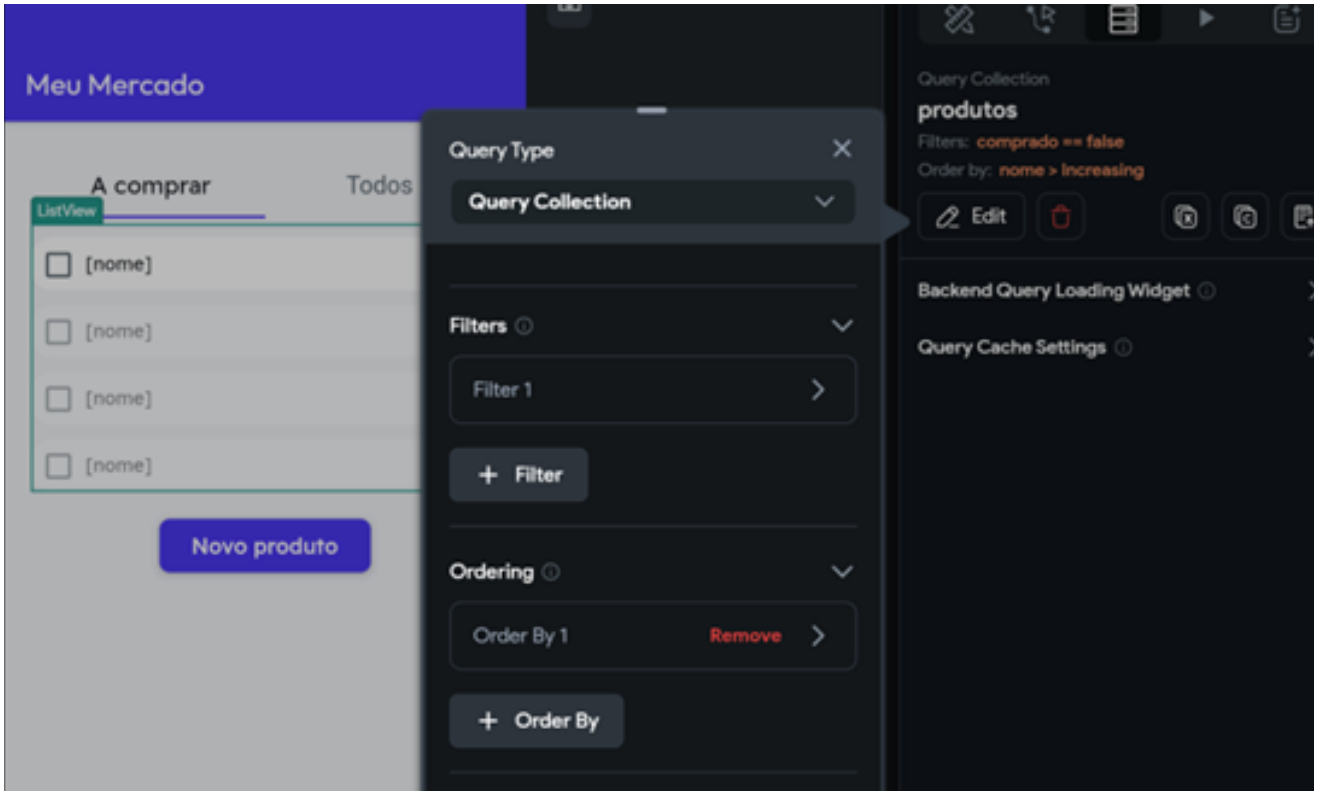


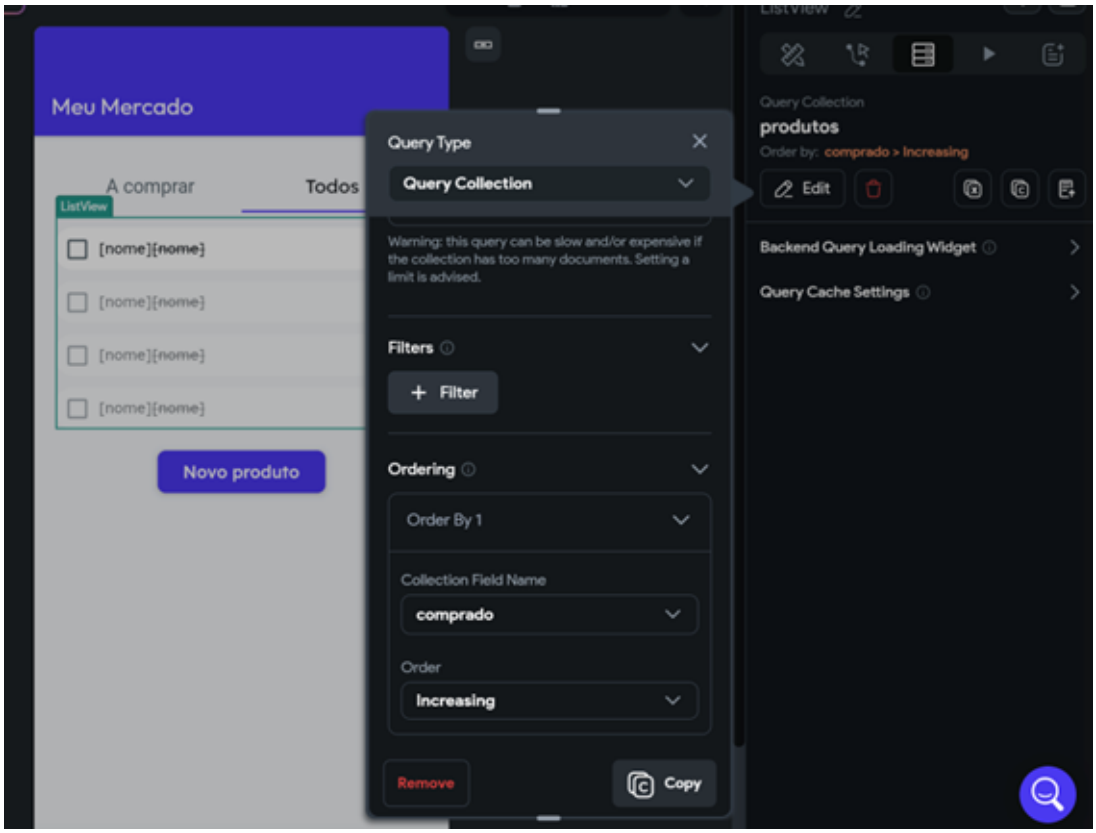
Podemos então Filtrar e Ordenar nossos dados de acordo com a aba que estamos. Na aba **Todos** podemos deixar como está, pois a finalidade dela é mostrar toda a nossa lista. Agora a aba A comprar precisa de um filtro para mostrar somente aqueles itens com **comprado false**. Para isso vamos na **ListView** da aba **A comprar**. Vamos na **Backend Query** e vamos editá-la. Vamos adicionar um filtro onde só buscamos o produto cujo campo comprado é falso.



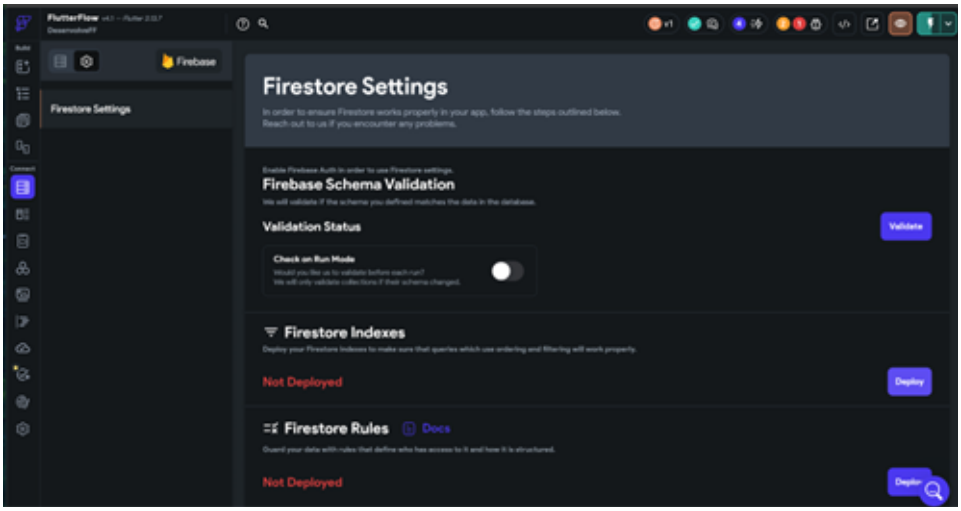
Com isso podemos até retirar da aba a comprar o texto riscado e a visibilidade condicional do texto normal, afinal, ele sempre vai aparecer.

Vamos melhorar mais um pouco essa visualização? Para facilitar para a pessoa que está utilizando, vamos ordenar a lista pelo nome do produto. Para isso basta irmos na mesma **Backend Query** e adicionar uma **Ordering** (ordem). Em a comprar ordenamos por ordem alfabética e na aba todos vamos ordenar por comprado.



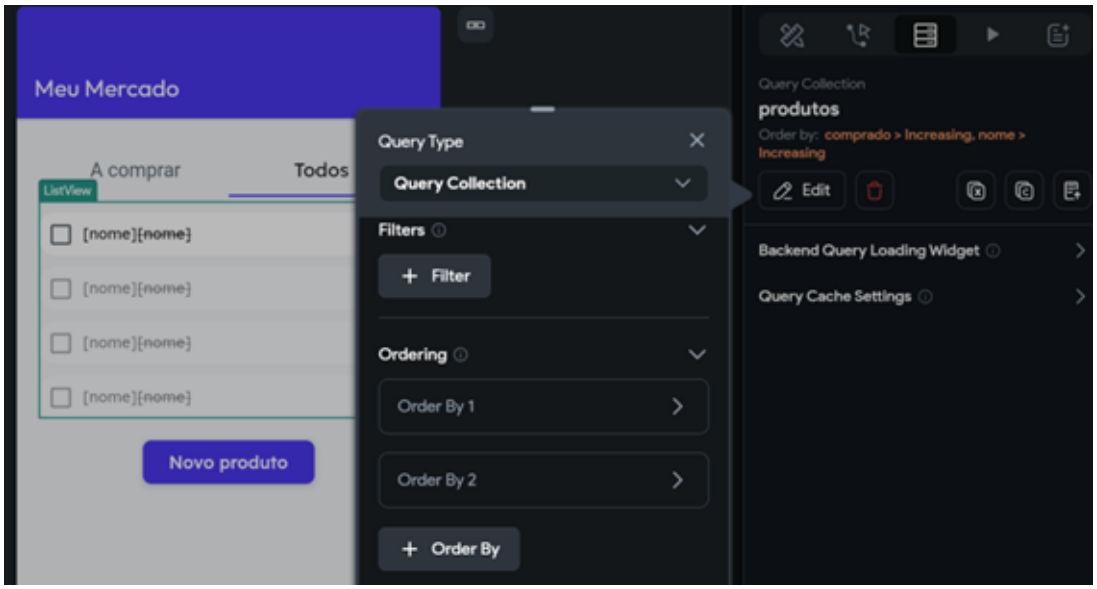


Na hora de executar, pode ser que apareça um erro no Firebase. Para resolvê-lo, basta clicar no bug e aparecerá a tela seguinte.



Aqui ele está dizendo que os dados do nosso Firestore não estão indexados, ou seja, não tem uma possibilidade de ordenação. Basta darmos **Deploy** em **Firestore Indexes** para resolver o problema. Feito isso, tudo volta a funcionar.

Podemos até acrescentar mais uma ordenação na aba Todos para ordenar tanto pelo comprado, quanto por ordem alfabética.



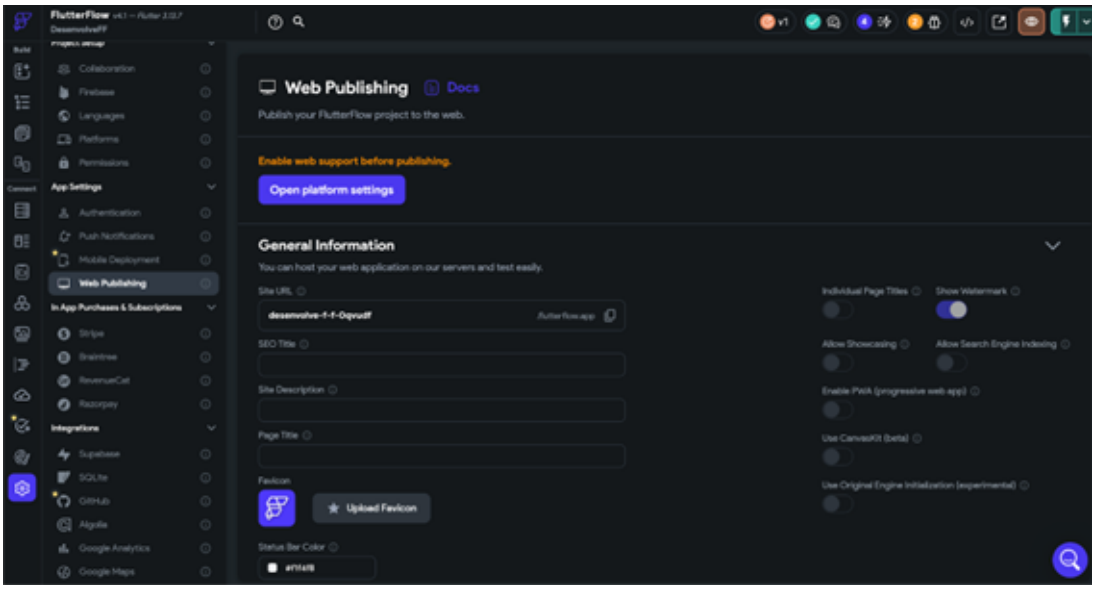
Ainda existem muitos aprimoramentos que podem ser feitos no nosso aplicativo, mas nosso MVP já está prontinho.

Explore bastante os widgets do FlutterFlow pois é uma plataforma muito robusta!

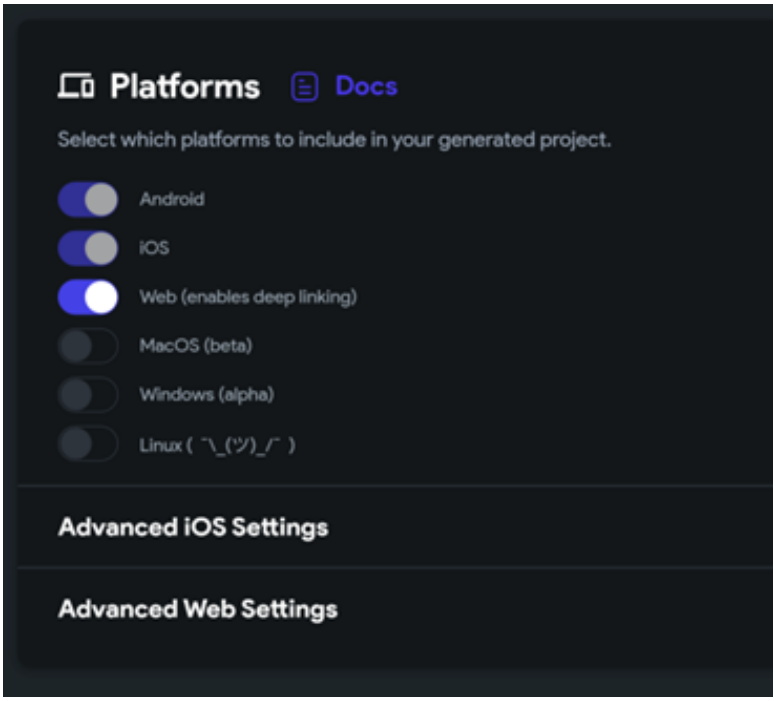
Implantação (Deploy)

Vamos agora implantar nosso aplicativo e entender como podemos entregar os nossos projetos com FlutterFlow. Aqui teremos duas formas de deploy: Para Web e exportando o código Dart/Flutter.

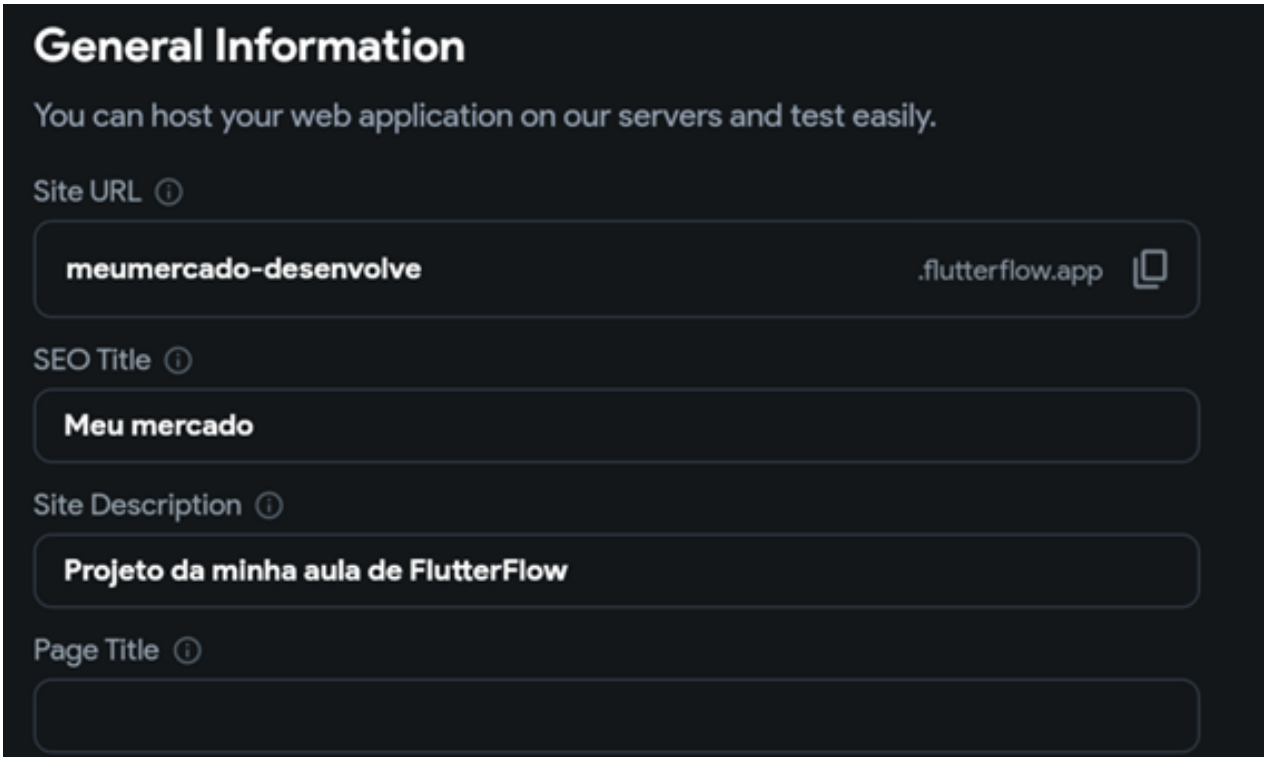
Para implantarmos na Web, vá ao menu esquerdo de configurações e clique em **Web Publishing**.



Agora clique em **Open platform settings** e habilite para web.

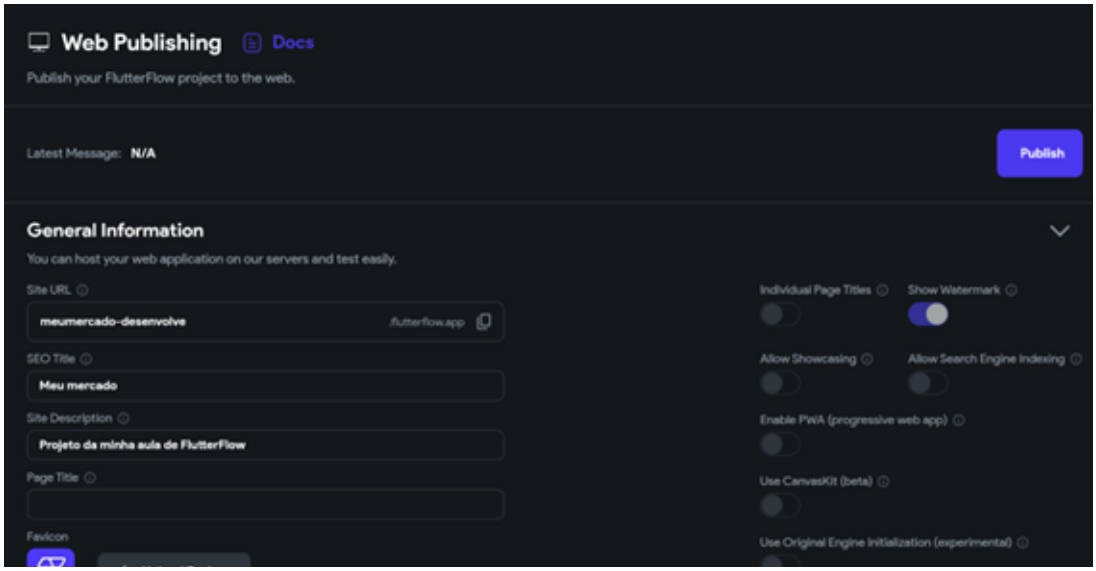


Agora voltamos para Web Publishing e vamos configurar as informações gerais.

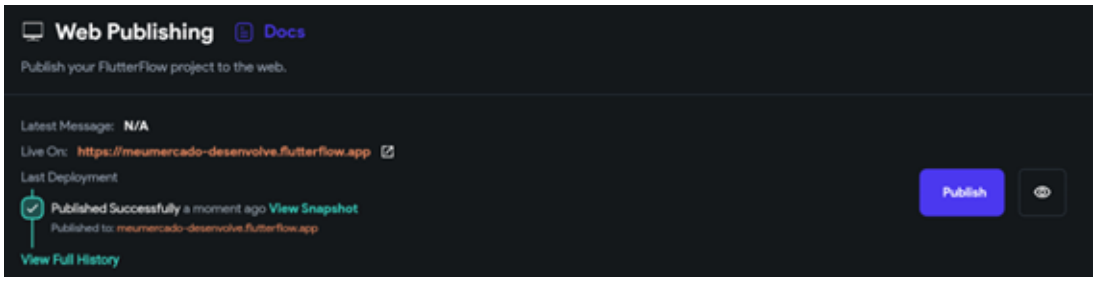


Aqui já podemos ver qual será a URL do nosso aplicativo.

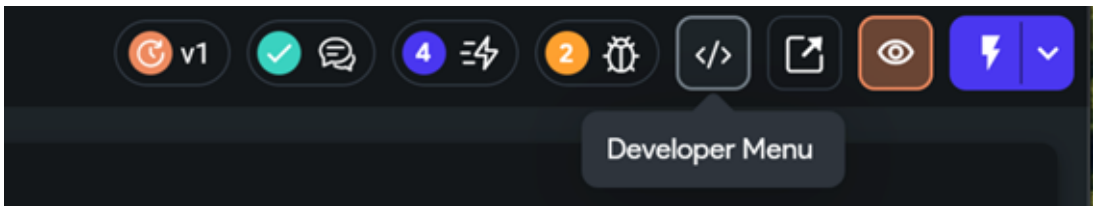
Depois, basta clicarmos em **Publish.**



E é essa URL criada que vocês enviarão para vermos os projetos de vocês!



E como fazemos para exportar o código Dart? Afinal, FlutterFlow é uma ferramenta low-code. No canto superior direito, temos o botão **Developer Menu**.



Porém, infelizmente, essa parte é paga na plataforma. Como comentamos no Bubble, o principal problema de plataformas no-code e low-code é que qualquer coisa além da criação de um MVP, precisa ser paga.

Mas não fique triste, mesmo com somente a publicação Web liberada, podemos testar diversas ideias com o FlutterFlow! Explore tudo!

Exercício

Agora é com você! Envie o link da sua lista de supermercado.

PROJETO
DESENVOLVE