FLUTTER FLOW

SUPABASE

Autora: Terezinha Márcia de Carvalho Lino

JOÃO PESSOA
2024

Sumário

1.INTRODUÇÃO

Flutter Flow é uma plataforma de desenvolvimento, sem código, que permite criar aplicativos móveis nativos para iOS e Android. Com o Flutter Flow, você não precisa saber codificar para criar aplicativos.

O Flutter Flow fornece uma interface visual e widgets de arrastar e soltar para projetar a interface do aplicativo, eliminando a necessidade de escrever o código você mesmo. O que torna o desenvolvimento de aplicativos mais rápido e acessível para não programadores.

1.1 Principais Recursos do Flutter Flow

- Ter biblioteca de widgets de arrastar e soltar para projetar a interface do aplicativo;
- Conectar aplicativos a bancos de dados, APIs, sem código;
- Permitir, por meio, da colaboração em tempo real, que as equipes construam juntas, os seus projetos;
- Ter planos gratuitos e pagos para atender diferentes necessidades;
- Publicar os App na Web e/ou em lojas oficiais de aplicativos;
- Desempenho nativo e UI com Flutter;
- Permitir personalizar os temas e design;
- Permitir inserir ações, lógica condicional, animações e muito mais sem código;

1.2 Benefícios de usar Flutter Flow

O Flutter Flow oferece vários benefícios importantes, que o tornam uma das melhores escolhas para a criação de APP.

- **Velocidade de Desenvolvimento**: uma das principais vantagens do Flutter Flow é a rapidez com que você pode construir um aplicativo. A

interface intuitiva de arrastar e soltar permite criar aplicativos completos sem a necessidade de escrever código. O que agiliza e acelera o processo de desenvolvimento.

- **Suporte multiplataforma:** Flutter Flow permite criar aplicativos compilados nativamente para iOS e Android a partir de uma única base de código.
- **Grande biblioteca de widgets:** Flutter Flow fornece acesso a toda a biblioteca de widgets Flutter enquanto usa uma interface visual, como gráficos interativos, mapas, animações, arrastando e soltando em vez de codificação.

2. FLUTTERFLOW – APRESENTAÇÃO DA INTERFACE

2.1 Menu do lado esquerdo – Menu dos Elementos

O menu de elementos é o primeiro ícone da barra de menu horizontal do lado esquerdo da tela. Os elementos estão agrupados da seguinte maneira:

- **Commonly Used Elements:** elementos que são bastante usados: text, column, row, container, image, button;
- **Layout Elements:** Container, Stack, Card, ListView, GridView, Divider, PageView, Carousel, Wrap etc;
- **Base Elements:** Text, Rich Text, Button, Ico, IconButton, ListTile, Calendar, VideoPlayer, Image, CircleImage, YouTuberPlayer.
- **Page Elements:** AppBar, FAB (botão, que coloca no canto da página), Drawer (menu que é usado nas laterais);
- **Form Elements:** elementos que são usados para construir formulários: TextField, DropDown, RadioButton etc.

2.2 Menu do lado esquerdo - Menu dos Componentes

O Menu dos Componentes pode ser encontrado no segundo ícone da barra de menu horizontal do lado esquerdo da tela. Você consegue criar componentes do zero

2.2.1 Templates - terceiro ícone da barra de menu horizontal do lado esquerdo da tela.

- Tem alguns templates que podem ser usados para acelerar o processo de criação do projeto;
- Templates em dark mode, light mode e outros.

2.2.2 Widget Tree

- Mostra a organização em pastas, dos elementos que estão sendo utilizados;
- Para adicionar um novo elemento, basta clicar no ícone e escolher o elemento. Cada página tem a sua Widget Tree.

2.2.3 Principais Widgests - Ferramentas

São várias as opções. Quando tiver dúvida sobre a utilização de cada uma, consulte a documentação do FlutterFlow. Alguns serão evidenciados aqui:

- **Linhas e Colunas: Colunas** Quando as informações são organizadas uma embaixo da outra. Sempre comece a organizar uma página usando uma coluna. **Linhas:** quando as informações são organizadas uma ao lado da outra.
- **Container:** elemento que serve para organizar outros elementos dentro do APP. Definir a altura e largura em pixel ou percentagem em relação ao espaço total. Pode deixar com a dimensão infinita, mas deve-se tomar cuidado, pois as linhas e colunas que estarão dentro dele vão tentar se adaptar a essa dimensão e provavelmente vai dar erro. Colocar cor, bordas, bordas arredondadas. Definir os espaçamentos. Definir a posição do container depende do elemento pai. Se o contêiner estiver dentro de uma coluna ou outro container, procure o elemento pai e defina a posição do elemento child. Colocar Elevation para evidenciar as bordas. Usar Box Shadow para sombreamento das bordas. Se preferir duas cores use o Gradient. Pode-se colocar também imagem no Background. Alinhar Child: se colocar um elemento dentro do elemento, é por aqui que se controla o seu alinhamento.

- **ListView:** é muito usado para colocar informações do Banco de Dados, quando precisar repetir os dados, de acordo com os que estão no Banco de Dados. Basta colocar um container dentro da ListView que os dados serão replicados, como uma lista.
- **GridView (visualização em grades):** colocando container dentro do gridview, dá para organizar cards um ao lado do outro.
- **TabBar:** barra de abas: serve para dividir a tela em 3 abas. Cada aba terá uma visualização diferente.
- **PageView:** para adicionar uma imagem na página.
- **Carousel**: para adicionar uma coleção de imagens.
- **Form Validation:** para validar formulários, basta colocar os elementos do formulário dentro dele.
- **Wrap**: para os container se acomodarem dentro das colunas.
- **CheckBox:** caixa para mostrar que algum item foi checado;
- **Switch:** caixa para trocar uma ação. Por exemplo: visualizar ou não.
- **Timer**: contador de tempo.
- **PDF View**: para visualizar um texto em pdf;
- **AppBar**: para colocar uma barra de navegação no App. Mas, o seu uso deixa o layout um pouco engessado, pois não permite inserir outros elementos.
- **FAB:** Botão que fica no final da página. Para ter acesso a este elemento, tem que clicar no nome da página, assim ele fica visível para colocar. Não aceita colocar dentro de uma coluna, por exemplo.
- **Drawer:** serve para ter um menu que recolhe e se torna visível quando clicado;
- **Stack:** para organizar um elemento sobre o outro:
  - Criar página;
  - Acrescentar uma coluna;
  - Acrescentar dentro da coluna o stack;
  - Colocar dentro do stack, 3 container. Eles ficarão sobrepostos. Para mudar a propriedade de cada um, clique neles para identificar. Para mudar a posição, clique no container e arraste para o local desejado.

2.2.4 Storyboard

Aqui você consegue visualizar quais páginas se conectam com outras páginas.

2.2.5 Firestore

Banco de Dados do Firebase, onde se faz o controle do banco criado.

2.2.6 DataTypes - App State

- A Appstate é como se fosse o cache de um site (evita que o app fique buscando dados no Banco de Dados, dessa forma economiza espaço);
- Uma variável AppState pode ser acessada e modificada em todo o aplicativo, em todas as páginas e componentes;
- Esse tipo de variável pode ser útil para armazenar dados que precisam ser compartilhados entre diferentes partes do aplicativo, como preferências do usuário e tokens de autenticação;
- Os dados salvos no appstate ficam armazenados de forma temporária. Quando o usuário sai do aplicativo aquele dado não é salvo. Então não deve ser usado para armazenar dados que precisam ficar pra sempre no banco de dados. Um exemplo: página de Consulta de CEP, dados de localidade podem ser armazenados em App State. Outro exemplo: Lista de agendamento de horário de um médico. Aqueles horários ficarão armazenados em App State. Pois, serão temporários.

2.2.7 API Calls

Onde se faz as chamadas às APIs, por exemplo: API do CEP.

2.2.8 Media Assets

Onde se faz a gestão das mídias para tela. Pode-se fazer o upload de uma foto que sempre será utilizada no APP, assim ela ficará disponível. **OBS: a foto de perfil não será guardada aqui, e sim no Banco de Dados.**

2.2.9 Custom Code

Pode criar Custom Functions, Custom Widget, Custom Actions, Custom Files.

2.2.10 Cloud Functions

Desenvolve funções para rodar na nuvem do Firebase.

2.2.11 Automated Tests

Para fazer testes automatizados.

2.2.12 Theme Settings

Onde define as fontes, tamanho da fonte, cores e design que serão usados no projeto.

2.2.13 Configurações

Diversas configurações do APP são definidas aqui. Por exemplo, quando o APP tiver entrada de usuário tem que habilitar no App Setting o **Authentication.**

2.3 Menu Superior

- Ícones de celular, tablet e desktop: clique nos ícones, para verificar se o APP está responsivo;
- Ícone do Besouro: se estiver laranja indica que tem algum erro, clique para saber qual o erro;
- Ícone de comentário;
- Ícone </>, clique para visualizar o código;
- Ícone da seta: quando tiver alguma dúvida sobre o desenvolvimento do projeto, você pode compartilhar o mesmo, clicando em Make Public e depois, basta copiar o link e compartilhar;
- Ícone do olho: para visualizar a parte de design do APP. Aqui não consegue ver a inteligência, conexão com o Banco de Dados;

- Ícone do Raio: Abre o modo teste do APP para ver o funcionamento do mesmo.

2.3.1 FlotterFlow IA

- Na tela central do Flutter, no menu superior clique no ícone de AI Generated Page, para criar uma nova página pela IA;
- Para criar um novo componente pela IA, clique no ícone +, New Component, Create with AI.
- No menu lateral à esquerda, vá até Theme Setting Colors e clique em Generate with AI, para gerar uma paleta de cores.
- No menu lateral, clique no ícone de Banco de Dados Firestore, clique para criar uma nova coleção, dê o nome da coleção e clica, aparece a opção de Generate with AI:
  - Entre no Firestore, Clique em criar nova coleção;
  - Clique em Generate AI;
  - Escreva o nome da nova coleção. Deve-se colocar o nome em inglês, pois em Português está dando erro;
  - Dar enter para criar a nova coleção. Se não estiver de acordo, pode-se repetir o processo que é criada uma nova coleção.

2.4 Menu do lado direito

Mostra as ferramentas para mudar as propriedades do elemento selecionado. A seguir será feita uma breve apresentação dessas ferramentas.

2.4.1 Primeiro Ícone: Propriedades

- Visibilidade condicional: você pode impor uma condição para que o elemento seja visualizado;
- Responsividade: escolhe se o elemento será visível, apenas no celular, tablet, desktop;
- Padding: controlar os alinhamentos;
- Propriedades: define a largura, altura, font, cor, bordas, sombreamento da borda, etc.

2.4.2 Segundo ìcone: Actions

Aqui são definidas as ações para o elemento selecionado. Por exemplo: selecione um botão e coloque a ação de levar para outra página. Você pode colocar várias ações em sequência.

2.4.3 Terceiro Ícone: Backend Query

Aqui você vai fazer uma consulta ao Banco de Dados, faz a conexão do FlutterFlow com o Banco de Dados – Firebase.

2.4.4 Quarto ìcone - Animações

Você pode colocar alguma animação no elemento selecionado.

2.4.5 Quinto ícone: Documentação

Você pode fazer toda a documentação do seu APP, para futuras consultas. Do lado do nome do elemento, estão 3 ícones, o primeiro é para salvar a configuração do componente, o segundo para salvar o widget, para poder ser utilizado em outros projetos.

- Value: Widget State: uploaded File URL;
- **Add Action – terceira ação;**
- Refresh Database Request: pagPessoal

3. SUPABASE – BANCO DE DADOS

Banco de dados relacional, veloz, escalável e relacional, que se integra de forma nativa com o FlutterFlow. Construtor facilitado de APIs. Possui plano gratuito e pago (para aqueles App publicados nas lojas virtuais).

3.1 Supabase Funcionalidades

- Banco de Dados SQL: Banco Relacional PostgreSQL;
- Auntenticação: Serviço completo de Auth;

- Storage: Serviço de Armazenamento;
- Edge Functions: Funções completas;
- Banco Realtime: Dados em tempo real;
- APIs: Comunicação via API;

3.2 Criar conta no Supabase

- Acessar o Supabase: https://supabase.com/
- Você poderá ter dois projetos de forma gratuita;
- Clicar em New Project;
- Name: App de Cursos EAD ;
- Database Password: Clicar em Generate a password: eadAppCursos23
- Copiar e salvar em lugar seguro (se for conveniente);
- Region: South America (São Paulo);
- Create a New Project.

3.3 Criar Tabelas no SupaBase

- Clicar no Table Editor (segundo ícone do menu, lateral esquerda;
- New Table;
- Name: tarefas;
- Columns: campos da tabela. São apresentadas duas colunas padrão, automaticamente: id (chave primária) e created_at (mostra o momento exato que o registro foi salvo no Banco de Dados)
    - Add Column: nome: tipo: text;
    - Add Column: nota: Tipo: int2;
    - Add Column: descCurta; Tipo: text.

Após criar a tabela, deve-ser criar as Regras de Segurança.

3.4 Regras de Segurança das Tabelas

- Clicar no nome da Tabela (menu esquerdo), para mostrar todos os campos criados;

- Clique no campo: No Active RLS policies;
- New Policy;
- For full customization;
- Policy Comand: ALL;
- Using Expression, clicar e colocar no espaço: true;
- With Check expression.: Clicar e colocar True;
- Save.

3.5 API Docs

Clicar na tabela e depois em API Docs, ele mostra as APIs automáticas que são criadas, quando você gera uma nova tabela dentro do Supabase.Os códigos são mostrados em JavaScript ou em Bash, basta clicar no ícone Language do lado superior direito. Na linguage Bash, mostra as URL das APIs. Por exemplo: url de buscar todos os cursos:

URL: ‘https://wocoghssbzkygoxjrdcq.supabase.co/rest/v1/cursos?select=*' \

Chave: "apikey: SUPABASE_CLIENT_ANON_KEY" \

Chave: "Authorization: Bearer SUPABASE_CLIENT_ANON_KEY”

Clicando em **Show Keys** aparecem todas as chaves das APIs, que poderão ser coladas em seus projetos.

OBS: Para aparecer a chave completa, clique em Project API key e escolha (anon public), aí mostra a chave completa.

3.6 SQL Editor

No menu lateral esquerda, clique em SQL Editor. New Query, você pode criar campos do SQL.

Exemplo: Selecionar todos os campos da tabela Cursos:

```
select
  *
  from
   cursos
```

Exemplo 2:

Selecionar o nome da tabela Cursos:

```
select
  nome
  from
   cursos
```

Depois clicar em Run para aparecer o resultado.

OBS: Se você editar algum dado errado (vírgula, por exemplo), vai dar erro. Para descobrir qual o erro, basta clicar no ícone: Debug with Supabase AI, ele mostra o erro e como deve ser.

3.6.1 SQL Editor – Templates

Clicando emTemplates você terá acesso a vários templates prontos para diversas funcionalidades. Exemplo: criar uma tabela, criar uma View, Add Column e outros

3.7 Database

Clicando no Database, você visualilza:

- Suas tabelas: Table;
- Schema Visualizar: mostra como as tabelas estão relacionadas;

3.8 Authentication

Clicar no ícone Authentication;

- Add user;
- Create a ne user;
- User email: colocar o email do usuário;
- User Password: senha;
- Create User.

3.8.1 Políticas de Segurança

- Em Authenticatio, clicar em Policies;
- Escolhe a tabela;
- New Policy.

3.9 Conectar o Supabase ao FlutterFlow

- Entrar no Fluter Flow;
- Configurações;
- Supabase;
- Clicar em Enable Supabase;
- API URL: Voltar para o Supabase: Configurações; API, copiar a URL e colar nesse espaço
- Anon Key: a chave anônima está abaixo da API URL no Supabase, copiar e colar no FlutterFlow;
- Clicar em Get Schema.

3.10 Authenticated Supabase

Em primeiro lugar deve-se habilitar a tabela Authentication no Supabase:

- Entrar no Supabase;
- Authentication (menu lateral);
- Providers;

- Habilitar apenas o email, deixe marcado apenas a primeira opção. A opção Confirm email, se estiver assinalada, o login somente será liberado depois da confirmação do mesmo. Será enviado ao email do usuário uma mensagem para ele confirmar. Quando estiver usando aplicativos publicados nas loja, seria recomendável ativar esta opção.
- Save.

Depois disso, entrar no FlutterFlow

- Entrar no FlutterFlow;
- Configurações;
- Authentication;
- Habilitar a Enable Authentication;
- Escolha: Supabase;
- Initial Page: Login;
- Logged in Page: Home Page.

Depois disso, em configurações clicar em Integrations, Supabase

- Clicar em Get Schema para atualizar as configurações

3.11 Página de Login

Depois de criar a página de criar conta e login, colocar inteligências nos botões.

3.11.1 Botão Criar Conta

- Clicar no botão criar conta;
- Action.
- **Add Action – primeira ação**
- Supabase Create Acount;
- Auth Provider: email;
- Email Field: email_created;
- Password Field: password_create;
- Confirm Password Field: confirm_password;

- **Add Action – segunda ação**
- Wait (delay); (esta ação é para evitar que usuário cadastrado tenha acesso imediato à página de destino0
- 1000 ms;
- **Add Action – terceira ação;**
- Navigation To;
- Home Page.

Depois disso, ir para a Home Page, clicar no Canva e habilitar a opção Requires Authentication.

3.12 Tabela Usuários

O Supabase na tabela Users nativa, não registra o nome, telefone e outros dados, assim você precisará criar uma tabela de Usuário, para suprir essa deficiência.

- Abrir o Supabase;
- Clicar em Tabelas;
- New table;
- Create a new table;
- Name: usuários;
- Description: pode deixar em branco;
- Columns: ele já traz o id r o create_at;
- O id é criado automaticamente. Mude o seu nome para user_id
- Na coluna user_id, Clicar no ícone de corrente do lado, para definir que ele será uma chave estrangeira: Select a schema: auth; Select a table to reference to: auth users; Select a column from auth: id uuid; Action if reference row is removed:cascade; Action if referenced row is removed: cascade
- Save;
- Add Column;
- email: text;

- nome: text;
- foto_perfil: text;
- tipo: text (tipo da pessoa: professor ou aluno)
- Save

(OBS: você pode colocar quantas informações quiser: cargo, telefone....)

- **Depois disso colocar a política de segurança**
- Clicar no nome da Tabela (menu esquerdo), para mostrar todos os campos criados;
- Clique no campo: No Active RLS policies;
- New Policy;
- For full customization;
- Policy Comand: ALL;
- Using Expression, clicar e colocar no espaço: true;
- With Check expression.: Clicar e colocar True;
- Save.

  Depois disso colocar os dados da Página Pessoal, de forma dinâmica.

3.12.1 Criando um usuário no Supabase

O usuário criado na tabela User não vai automático para a tabela Usuário. Para isso, faça:

- Vá para tabela Usuario, clicar em Insert row;
- User_id: clicar na opção Select record. Ele entra na tabela auth.users, clique sobre o id do usuário que deseja enviar para a tabela Usuários;
- Created_at: é criado automaticamente;
- Nome: coloque o nome do usuário;
- Email: coloque o email do usuário;
- Foto: coloque a foto;
- Tipo:professor (exemplo);
- Save.

3.12.2 Fazer uma relação da Tabela Cursos com o Professor

- Clicar na tabela Cursos;
- Editar;
- Add Column;
- Prof_id: tipo: int8;
- Clicar na Foreign Keys, que está localizado na parte de baixo da tabela:Add Foreign Key Relations. Shema: Public; Select a table to reference to: usuários. Select a column from: prof_id = users_id. No action; Save; Save.

3.13 Criar Views

Pode-se entender View, como se fosse uma janela para visualizar dados de uma determinada tabela. Só para facilitar, pois a view vai retirar da tabela, apenas os dados que você definiu.

- Entrar no Supabase;
- SQL Editor;
- Templaste;
- Add View. Exemplo:

```
--Criar view para nova tabela cursos
create
  view cursosNomes as
  select
    id,
    nome
  from
  cursos
```

Como resultado da View criada, você terá uma tabela com apenas os nomes dos cursos.

Para apagar a view, basta colocar: drop view cursosnomes.

3.14 Policies – Políticas de Segurança para Tabelas

- No Supabase, clicar em Athentication;
- Policies;
- Clicar em RLS enable, para habilitar;
- Policy name: Liberar Tudo;
- Allowed operation: ALL;
- Target role: Defaults to all (public) roles f none selected;
- 1: true;
- 1: true;
- Review;
- Save policy

3.15 Database Wrappers

É usado para conectar com outras Plataformas.

- Entrar no Supabase;
- Database;
- Wrappers;
- Enable Wrappers;
- Create a new wrapper;
- Firebase (como exemplo);
- Wrapper Name: Firebase (conectar com um projeto existente no Firebase)
- Project Id (entrar no Firebase e copiar o código do projeto escolhido), colar;
- Voltar no Firebase, em Configurações, clicar em Contar de Serviço, Clicar em Gerar nova Chave privada, Gerar Chave, Salvar, Copiar toda chave privada e colar no Supabase em Service Account Key;

- Add foreign tables: clicar. Edit foreign table: Clicar em Select a target ... selecione: Firestore Collection: Table name: teste_firebase; Object: colocar a coleção do Firebase que está usando: Save, Save;
- Volte na Table Editor que a tabela teste_firebase foi criada lá.

3.16 Importar Dados CSV para o Supabase

É recomendável que você faça a importação dos dados antes de criar as tabelas, para não dar erros.

- Baixar a tabela que será importada no formato CSV;
- Entrar no Supabase;
- Table Editor;
- New Table;
- Name: cursos;
- Import data via spreedSheet;
- Clicar em Drag and drop, or beower your files;
- Escolha o arquivo CSV que será importado;
- Save;
- Confira os dados e colunas, colocando as chaves estrangeiras necessárias;
- Crie uma chave primária: Add a neew column; id; type: int8; Is identity;Is Primary Key, Desabilite: Allow Nullable;
- Save

4. FUNDAMENTOS SQL PARA SUPABASE

4.1 Criar Tabela

- Entrar no Supabase;
- Clicar em SQL Editor;
- Template;
- Create a Table

- Apagar a função que se apresenta;
- Clicar do lado da Create Table e Renomear: Name: FunçõesSQL; Rename Query:
- Exemplo:

Calcular média das notas:

```
select
  avg(nota)
from
  cursos
```

Calcular a nota máxima:

```
select
  max(nota)
from
  cursos
```

Calcular a soma das notas:

```
select
  sum (nota)
from
  cursos
```

Contar o número de cursos:

```
select
  count (nome) as contagem_cursos
from
  cursos
```

Criar uma View

```
create
  view view_contagem_cursos as
select
  count(nome) as contagem_cursos
from
  cursos
```

A view será criada e disponibilizada em Table Editor. A vantagem da View é mostrar um novo resultado, sempre que os dados da tabela forem alterados.

4.2 Create enum types

Este tipo de dado te permite qualificar o dado, por exemplo: iniciante, intermediário, avançado.

- Entrar no Supabasr;
- Editor Table;
- Tabela Cursos;
- Inserir uma coluna;
- Name: nível;
- Data Type: clicar em +create enum types. Você é direcionado para uma nova página:
  - Database;
  - Enumerated Types;
  - Create Types;
  - Name: nível;
  - Values: iniciante; avançado;
  - Create type.

Depois disso, volte para Tabela Curso;

- Dta Types;
- Type: selecione: nível (enum type criado); Default Value: iniciante (valor padrão, se outro dado não for colocado, começará com iniciante)
- Save

  Depois pode criar uma função para saber a soma das cargas horárias, por nível e por curso.

```
select
nivel,
  sum(carga_horaria) as carga_horaria_por_nivel
from
```

```
  cursos
group by
  nivel
```

4.3 Join no SQL – União de Tabelas

Criar uma View para visualizar o nome do curso, com descrição curta, com o nome do professor e seu e-mail. A Join irá juntar as duas tabelas, cursos e usuários.

```
--View para juntar tabelas cursos + professores

select
    c.nome as curso_nome,--chamar o curso de c. O nome da tabela, no
resultado será curso_nome,
    c.id as curso_id,
    c."descCurta" as curso_descCurta, -- a desc_Curta, no resultado será
curso_desc_Curta
    u.nome as prof_nome,
   u.email as  prof_email
from
  cursos c
join
  usuarios u  on c.prof_id = u.user_id
```

Você pode acrescentar uma View, que será disponibilizada nas Tabelas

```
create
  view dados_curso_prof as
select
    c.nome as curso_nome,--chamar o curso de c. O nome da tabela, no
resultado será curso_nome
    c."descCurta" as curso_descCurta, -- a desc_Curta, no resultado será
curso_desc_Curta
    u.nome as prof_nome,
   u.email as  prof_email
from
  cursos c
join
  usuarios u  on c.prof_id = u.user_id
```

5. APP CURSO EAD

5.1 Criar Tabela Cursos

- Entar no Supabase;
- Table Editor;
- Create a New Table;
- Name: cursos;
- Description: App de Cursos;
- Enable Row Level Security: habilitar;
- Columns: id e created_at: já vêm criados;
    - Add Columns: nome, text; nota do curso: int4; descCurta: text; desc_longa: text; capa_Url: text; carga_horaria: int4
- Save.

Depois disso, entrar no Supabase

A tabela cursos, já estará criada. Você pode inserir uma Row, uma Column, Import data from CSV, dentro dessa tabela, clicando em Insert.

Exemplo de inserir uma linha:

- Clicar em Insert Row;
- O id e created_at, são gerados automaticamente;
- Nome: Curso de Inglês Avançado;
- Nota: 5;
- DescCurta: Aulas avançadas para conversação.
- Save;
- Voce também pode deletar os dados, clicando em Delete.

5.2 Planejamento do APP

- **Problema**: Entender o problema existente no mercado e a persona envolvida;
- **Solução:** Conceitualizae a tecnologia que irá solucionar o problema;

- **Design de Telas:** Elaborar telas por meio de um wireframe e fluxo de usuário;
- **Banco de Dados:** Modelar o banco de dados do aplicativo e relação entre tabelas;
- **Desenvolvimento:** Contruir o App conectando o frontEnd com o backEnd

5.3 Clonar o Projeto:

https://app.flutterflow.io/project/supabase-app-de-cursos-iv9juo?tab=uiBuilder&page=MeusCursos

Como o objetivo é aprender sobre o Supabase, a parte de design não será trabalhada. Então, clique no link acima e depois que abrir no FlutterFlow, clicar em Clone, assim, você poderá editar as páginas.

OBS: A IA do NoCode Start Up, responde as dúvidas sobre o FlutterFlow, Supabase e Firebase, basta perguntar: https://app.flutterflow.io/project/supabase-app-de-cursos-iv9juo?tab=uiBuilder&page=MeusCursos

Depois que criar as Policies no Supabase e ativar o Authenticated, começar a colocar inteligência nos elementos das páginas.

6. PÁGINA DE LOGIN

6.1 Botão Criar Conta

- Clicar no Botão Criar Conta;
- Action;
- Add Action;
- Open;
- Supabase Authentication;
- Create Account;

- Provider: email;
- Email Field: emailAddressCadastro;
- Senha: passwordCadastro;
- passwordConfirm;
- Add Action – segunda ação;
- Navigate to Meus Cursos.

6.2 Botão Login

- Clicar no botão Login;
- _Login;
- Add Action;
- Supabase Authentication Log In;
- Provider: email;
- Email Field: email_Adress;
- Password_login.

7. USE A IA COMO SUPER FUNCIONÁRIO

Aprenda como usar IA da forma correta para te ajudar a criar seus projetos no Supabase.

7.1 IA do Supabase

- Entrar no Supabase;
- Clicar em SQL Editor;
- Clicar no ícone Assistant;
- Coloque as informações sobre o que você quer que a IA crie. Por exemplo: tabela cursos, nome do curso, carga horária, professor, data de criação, quantidade de alunos.
- Depois disso, você pode aceitar (clicando em replace code) copiar, ou descartar. A IA mostra a tabela que foi criada;

- Depois disso, você pode pedir: preencha a tabela com dados aleatórios em todos os campos. Ela cria os dados.

Você pode também pedir para a IA fazer correções, em suas funções, para isso:

- Clicar em FunçõesSQL;
- Clique na Função que você quer corrigir;
- Clique em Debug with Supabase AI;
- Ela irá mostrar onde está o erro.

7.2 NoCode. Ais – IA da NoCode

- Entrar na Comunidade NoCode;
- Escolha NoCode Ais. Estão disponibilizadas 6 IAs: WeWeb, Xano, AppGyver, Bubble, Firebase, Supabase e FlutterFlow;
- Clicar na IA do Supabase;
- Faça as seguintes perguntas:
    - Você é meu consultor de modelagem de dados e tabelas. Estou criando um aplicativo de cursos on line, por favor me diga quais tabelas e relações precisarei ter.

8. API

As APIs servem para buscar dados de um sistema para outro. São utilizadas também para que o Front End se comunique com o BackEnd.

8.1 Postman API Plataform

“O Postman é um API Client que facilita aos desenvolvedores criar, compartilhar, testar e documentar APIs. Isso é feito, permitindo aos usuários criar e salvar solicitações HTTP e HTTPs simples e complexas, bem como ler suas respostas”.

Você, na maioria da vezes, consegue testar as suas APIs sem ter que pagar.

Entrar na Plataforma: https://www.postman.com/ e faça o download, clicando em Download the desktop app for. Escolha o Windows. Será instalado no seu computador. Depois de abrir preencha o dados solicitados.

- Entrar na Home;
- No menu lateral, você consegue compartilhar seus trabalhos (depende do plano); Fazer integrações com outras plataformas, como GitHub e outras.
- No menu superior, tem o My Workspace, onde você poderá gerenciar as suas APIs.

8.1.1 My Workspace

- Clicar no My Workspace;
- Collections (menu lateral), clicar:
  - New: clicando em New, você consegue fazer novas chamadas: Http e outros tipos de criações;
  - Import: clicando em importar você poderá importar outras APIs e/ou arquivos.

Criar nova coleção

- Clicar em Collections;
- + New Collection;

Cria-se uma nova coleção; Clicar nos três pontinhos ao lado, para escolher a ação que deseja. Clicar em Add Request (adicionar uma chamada). Tem vários tipos de chamadas: GET (puxar informações), POST (fazer novos registros no Banco de Dados), PUT e PATCH (para atualizar dados), DELETE (para apagar dados no BackEnd), HEAD, OPTIONS e outras.

- Você pode criar uma coleção do zero, ou então, escolher um template.

Criar Novas Variáveis – Environments

- Clicar em Environments;
- New Environments;
- Variable.

8.1.2 APIs no Supabase - GET

- Entrar no Supabase;
- Escolher uma tabela que foi criada;
- Cada tabela terá a API Docs;
- Clicar na API Docs da tabela escolhida;
- Clicar e Read rows, para mostrar os códigos;
- Os códigos são apresentados em JavaScript;
- Para facilitar pode clicar em Bash, ele mostra as API das rows;
- Clicar em Show Keys, para disponibilizar as chaves;
- Entrar no Postman;
- Clicar em New Collection, Add Request;
- No Get, cole a URL que foi copiado do Supabase, como a que está abaixo: https://wocoghssbzkygoxjrdcq.supabase.co/rest/v1/cursos?select=*
- Clicar em Header,
    - clique em Key e coloque: apikey;
    - E value, coloque a chave: eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6IndvY29naHNzYnpreWdveGpyZGNxIiwicm9sZSI6ImFub24iLCJpYXQiOjE3MTU5Nzk1NjksImV4cCI6MjAzMTU1NTU2OX0.tlPOOZeUTkFoEur0anCeFCYtu56YvsFsPeomTcrVZpY
- Clicar em Params: Inicialmente no Value está *, quer dizer que vai mostrar toda a tabela. Se você quiser pode selecionar por elemento, por exemplo “nome”, ai o resultado mostrará apenas os nome dos cursos.
- Clicar em Send: aparece o resultado.
- Salve com o nome GetCursos;

- Clicar nos 3 pontinhos para duplicar e dê o nome de GetCursos com Filtro

Usando o Filtering na API Docs - GET

- No Supabase, escolha uma tabela e entre em sua API Docs;
- Ir até Filtering;
- Copie a URL;
- No Postman, clicar na request GetCursos com Filtro;
- No Get, colar a URL copiada do Supabase;
- Neste modo, você poderá procurar o determinado dado de acordo com o id. Clicando em Params, na coluna id, Value, coloque eq.3, por exemplo, vai trazer o curso que tem o id = 3.

Usando Filtro por nome - GET

- No Supabase, escolha uma tabela e entre em sua API Docs;
- Ir até Filtering;
- Copie a URL;
- No Postman, clicar na request GetCursos com Filtro;
- No Get, colar a URL copiada do Supabase. Faça a seguinte mudança na URL:
    - https://wocoghssbzkygoxjrdcq.supabase.co/rest/v1/cursos?id=eq.1&select=*
    - https://wocoghssbzkygoxjrdcq.supabase.co/rest/v1/cursos?NOME=ilike.*nome*&select=*
    - O que alterou: no lugar do id, coloquei a variável “NOME”; no lugar do eq (procurar alguma dado igual), coloquei “ilike” (significa que a busca não precisa estar escrita igual ao que está no código, a API vai entender); No lugar do 1, coloque *nome* (precisar estar entre asterisco);
    - Depois disso, clique em Params, para testar. Key: nome; ilike*crochê*, Send. O resultado será cursos que têm o nome que traga a palavra crochê.

8.1.3 APIs no Supabase - POST

O Método POST é utilizado para criar novos dados.

- Entrar no Postman;
- Tabelas;
- Cursos;
- Duplicar a Get Filtro Nome Cursos, denominando: POST Inserir Curso
- Depois disso, entrar no Supabase. Entre na tabela Cursos, API Docs; Insert Row, Copiar a URL e colocar no Postman;
    - POST: https://wocoghssbzkygoxjrdcq.supabase.co/rest/v1/cursos
    - Copiar a última linha e colar no Postman, clicando em raw `{ "some_column": "someValue", "other_column": "otherValue" }`
    - Para inserir um novo curso, faça as seguintes mudanças: em some_colunm: insira: “nome”; “some "value” insira Curso de Postman; “other_colums”: insira: “descCurta”; “otherValue” insira: O melhor curso de Postman.
    - OBS: você pode inserir quantas colunas você quiser; basta acompanhar a tabela original (nota, desc_Longa, professor, e outros) e preencher os dados como explicado acima.
    - Depois disso: Save e Send. Os dados serão inseridos na Tabela no Supabse.

8.1.4 APIs no Supabase – UP Date – Método PATCH - Atualizar dados da tabela

- Entrar no Supabase;
- Escolher a tabela;
- Entrar na API Docs;
- Clicar em Update rows;

- Copiar a URL:
  https://wocoghssbzkygoxjrdcq.supabase.co/rest/v1/cursos?some_column=eq.someValue
- Colar no Postman;
- Fazer as alterações na URL para:
- https://wocoghssbzkygoxjrdcq.supabase.co/rest/v1/cursos?id=eq.12
  Significa que: você que alterar algum dado do curso cujo id = 12 (aqui você coloca o id do curso que queira alterar).

8.1.5 APIs no Supabase – Delete – Método PATCH - Deletar dados da tabela

- Entrar no Supabase;
- Escolher a tabela;
- Entrar na API Docs;
- Clicar em Delete rows;
- Copiar a URL:
  https://wocoghssbzkygoxjrdcq.supabase.co/rest/v1/cursos?some_column=eq.someValue
- Colar no Postman;
- Fazer as alterações na URL para:
  https://wocoghssbzkygoxjrdcq.supabase.co/rest/v1/cursos?id=eq.12*

  Significa que: você que deletar o curso, cujo id = 12 (aqui você coloca o id do curso que queira deletar).
- Save;
- Send.

8.2 APIs de Autenticação -

- Entrar no Supabase;
- Clicar em API Docs, menu lateral;
- Clicar em User Management;

- Colocar em Bash;
- Clicar em Sign Up;
- Copiar a URL:
  `https://wocoghssbzkygoxjrdcq.supabase.co/auth/v1/signup`
- Colar no Postman
- Copiar a última linha e colar no Postman no body:

```
{
  "email": "someone@email.com",
  "password": "TqSPSLOcLQSWoNbOMFhm"
}
```

Para inserir um novo usuário, coloque os dados do mesmo. Como:

```
{
  "email": "maya@email.com",
  "password": "maya123"
}
```

Depois disso, o sistema gera um token de acesso. Se você quiser visualizar quais as informações estão sendo pedidas, faça o seguinte.

- Copie o token


```
{
    "access_token":
"eyJhbGciOiJIUzI1NiIsImtpZCI6IkhUQzE1MDZpUFZKVTYxc08iLCJ0eXAiOiJKV1QifQ.
eyJhdWQiOiJhdXRoZW50aWNhdGVkIiwiZXhwIjoxNzE3NjIwNzkzLCJpYXQiOjE3MTc2MTcx
OTMsImlzcyI6Imh0dHBzOi8vd29jb2doc3Niemt5Z294anJkY3Euc3VwYWJhc2UuY28vYXV0
aC92MSIsInN1YiI6IjNmMTI0NDAzLWJmZjgtNDE5MC05NWUzLWIwMzFjMmQ3Y2ViOSIsImVt
YWlsIjoibWF5YUBlbWFpbC5jb20iLCJwaG9uZSI6IiIsImFwcF9tZXRhZGF0YSI6eyJwcm92
aWRlciI6ImVtYWlsIiwicHJvdmlkZXJzIjpbImVtYWlsIl19LCJ1c2VyX21ldGFkYXRhIjp7
ImVtYWlsIjoibWF5YUBlbWFpbC5jb20iLCJlbWFpbF92ZXJpZmllZCI6ZmFsc2UsInBob25l
X3ZlcmlmaWVkIjpmYWxzZSwic3ViIjoiM2YxMjQ0MDMtYmZmOC00MTkwLTk1ZTMtYjAzMWMy
ZDdjZWI5In0sInJvbGUiOiJhdXRoZW50aWNhdGVkIiwiYWFsIjoiYWFsMSIsImFtciI6W3si
bWV0aG9kIjoicGFzc3dvcmQiLCJ0aW1lc3RhbXAiOjE3MTc2MTcxOTN9XSwic2Vzc2lvbl9p
ZCI6ImQ0NDI0MTJjLWEyZTctNGNmNy04Yjc5LTcwMmNjNzU5MzJjMCIsImlzX2Fub255bW91
cyI6ZmFsc2V9.EvACGOUOabAQlaDEQNJCyPtMSf9_uEXGJcbmzgFCFlk",
    "token_type": "bearer",
    "expires_in": 3600,
    "expires_at": 1717620793,
    "refresh_token": "-V-g0IDlEqDYVtDYJL0tdw",
    "user": {
        "id": "3f124403-bff8-4190-95e3-b031c2d7ceb9",
        "aud": "authenticated",
        "role": "authenticated",
        "email": "maya@email.com",
```

```
        "email_confirmed_at": "2024-06-05T19:53:13.037416137Z",
        "phone": "",
        "last_sign_in_at": "2024-06-05T19:53:13.041674403Z",
        "app_metadata": {
            "provider": "email",
            "providers": [
                "email"
            ]
        },
        "user_metadata": {
            "email": "maya@email.com",
            "email_verified": false,
            "phone_verified": false,
            "sub": "3f124403-bff8-4190-95e3-b031c2d7ceb9"
        },
        "identities": [
            {
                "identity_id": "b0956920-9677-49b0-94a9-c3fcc0d37e12",
                "id": "3f124403-bff8-4190-95e3-b031c2d7ceb9",
                "user_id": "3f124403-bff8-4190-95e3-b031c2d7ceb9",
                "identity_data": {
                    "email": "maya@email.com",
                    "email_verified": false,
                    "phone_verified": false,
                    "sub": "3f124403-bff8-4190-95e3-b031c2d7ceb9"
                },
                "provider": "email",
                "last_sign_in_at": "2024-06-05T19:53:13.030647951Z",
                "created_at": "2024-06-05T19:53:13.030697Z",
                "updated_at": "2024-06-05T19:53:13.030697Z",
                "email": "maya@email.com"
            }
        ],
        "created_at": "2024-06-05T19:53:13.020418Z",
        "updated_at": "2024-06-05T19:53:13.054426Z",
        "is_anonymous": false
    }
}
```


E cole em: https://jwt.io/
Ele irá mostrar as informações.

8.2.1 API Login

- Entrar no Supabase;
- API Docs – menu lateral;

- User Management;
- Log In With Email/Password;
- Coloque na linguagem Bash;
- Para login deve-se fazer um POST;
- Copie a URL;
- Entre no Postman;
- Duplicar o Post Auth Cadastro e Cole a URL;
- No Body cole a última linha do código:

```
{
  "email": "someone@email.com",
  "password": "EzaEtmLbpejEdiTlBmHF"
}
```

- Send. O resultado trará um token de acesso.

8.2.2 Como colocar seguraça nas Tabelas – POLICIES

- Entrar no Supabase;
- Authentication;
- Policies;
- Têm criadas as tabeças de Cursos e Usuários, com as respectivas regras, que foram criadas;
- Na política de Cursos, clicar nos 3 pontinhos e Edit;
- Em Target roles, escolha a opção autheticaded (somente o usuário logado terá acesso ao dados do Banco de Dados);
- Review;
- Save Policy;
- Se você voltar ao Postman, clicar jo Get Ler Cursos, não consiguirá ver nenhum dado, pois, não está logado;
- Para fazer o login, você precisa colocar o token. Vamos utilizar o token gerado, e atividade anterior;
- No Postman, clique em Authorization, Type: Barer Token;

- Inserir o token no local solicitado (o token foi gerado na operação cadastro);
- Quando finalizar o seu App é importante você restringir o acesso apenas para o usuário autenticado.

9. PLANEJAMENTO BACKEND SUPABASE

- Definir o Conceito Geral do App EAD: funções, telas e usuários;
- Modelagem Completa do Banco de Dados: detalhamento e relações das tabelas;
- Planejamento Projeto BackEnd: BackEnd pronto para conectar com qualquer tipo de Front;
- Setup Tabelas + Views: Criação das tabelas e views dentro do Supabase;
- APIs Variadas: Estrutura APIs para conexão ao frontend.

9.1 Conceito Geral do App

Utilizar o Miro para fazer o diagrama do App.

- **Conceito**: App de Cursos EAD de professores e alunos. Os professores podem cadastrar novos cursos e os alunos podem se matricular nos cursos disponíveis.
- **Funcionalidades Coren(essenciais) do App**:

  Professores

  - Visualizar Lista de seus Cursos criados;
  - Visualizar qte Alunos Total;
  - Adicionar, Editar e Deletar Cursos;
  - Adicionar, Editar e Deletar aulas dos cursos;
  - Adicionar Aulas aos Cursos.

  Alunos

- Visualizar Lista de Cursos Disponíveis;
- Visualizar Detalhes do Curso;
- Opção de se matricular no curso;
- Opção de dar o check de curso realizado.

Ambos

- Upload imagem de perfil.

Administradores do App

- Podem tornar um usuário em tipo professor.

- **Páginas que serão criadas**
  - **Tela 1:** Login e Cadastro;
  - **Tela 2:** Dashboard Home Aluno
    Total Cursos Matriculados;
    Total Cursos Finalizados;
    Cursos Matriculados;
    Cursos Disponíveis;
  - **Tela 3: Dasboard Home Professor**
    Totais Meus Cursos;
    Totais Alunos;
    Meus Cursos Criados;
    Popup Adicionar novo Curso
  - **Popups**
    Detalhes Cursos – Editar e Deletar
    Adicionar Novo Curso
    Popup meu Perfil

- **Tipos de Usuários**
  - Administradores do App;
  - Aluno;
  - Professor

10. METODOLOGIA DE MODELAGEM

- Análise de Requisitos: levantamento e análise de requisitos de negócio do software. Entender o que o App vai fazer e qual a solução ele propõe;
- Modelo Conceitual: etendimento geral dos dados do negócio (visão de alto nível);
- Modelo Lógico: detalhamento das entidades, atributos e relacionamentos;
- Modelo Físico: implementação do banco de dados em SGBD, por meio de SQL.

10.1 Modelagem na Prática

Exemplo da No-Code Start-Up.

10.1.1 Análise de Requisitos

- Oferece capacitação para desenvolvimento de App;
- Os alunos podem comprar vários cursos;
- Os professores são: Celdo Neto e Matheus Castelo.

10.1.2.Modelo Conceitual

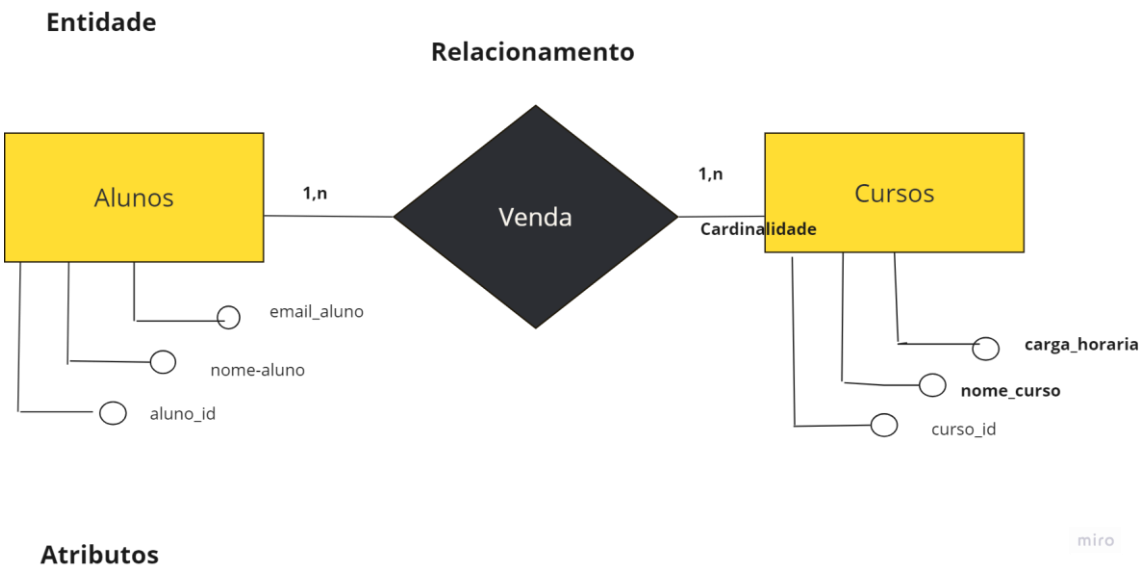
Entendimento geral dos dados no negócio (visão alto nível)

- Definir Entidades: são conceitos do negócio que precisamos armazenar informações. Exemplos: Alunos, Cursos, Professores, Vendas.
- Definir Atributos: sãrsoo as informações de cada Enditade (campos). Exemplos: email_aluno, nome_aluno, aluno_id, carga_horaria, nome_curso,
- Definir Relacionamentos e Carfinalidade: as tabelas do banco de dados poderão relacionar entre si.
    - Relacionamento um pra um. Exemplo (1 pra 1): 1 professor possui apenas 1 curso. Um curso tem apenas 1 professor. Você pode até juntar tabelas, ou seja, fazer a união delas, já que os dados não vão se repetir.

- Relacionamento 1 pra muitos (1 pra n): Exemplo: um curso pode ter várias compras; uma venda por curso. O lado n recebe uma chave estrangeira do lado do curso, que promoverá a conexão entre as tabelas.
- Relacionamento muitos pra muitos (n pra n): um aluno pode ter vários cursos, um curso pode ter vários alunos. Poderá criar uma nova tebela de conexão. Por exemplo> “alunos_cursos ou “vendas”.

- Construir Modelo Conceitual Final: Diagrama de Entidade Relacionamento(DER).

  Exemplo de Modelo Entidade de Relacionamanto: modelo para descrevermos as entidade, atributos e relacionamentos:



Exemplo das relações entre Imobiliária, Locador, Locatário e Imóvel.

- Um imóvel tem um locador, mas o locador pode ter vários imóveis;
- Uma locadora pode ter contatos vários locadores e o locador pode ter contatos comvárias locadoras;

- Uma Corretora gerencia vários locatários, mas o locatário, aluga um imóvel específico apenas uma Corretora (no caso deste imóvel);
- O Locatário pode alugar um imóvel (neste caso);
- O imóvel pode ser alugado por vários Locatários. (grupo de amigos, por exemplo).

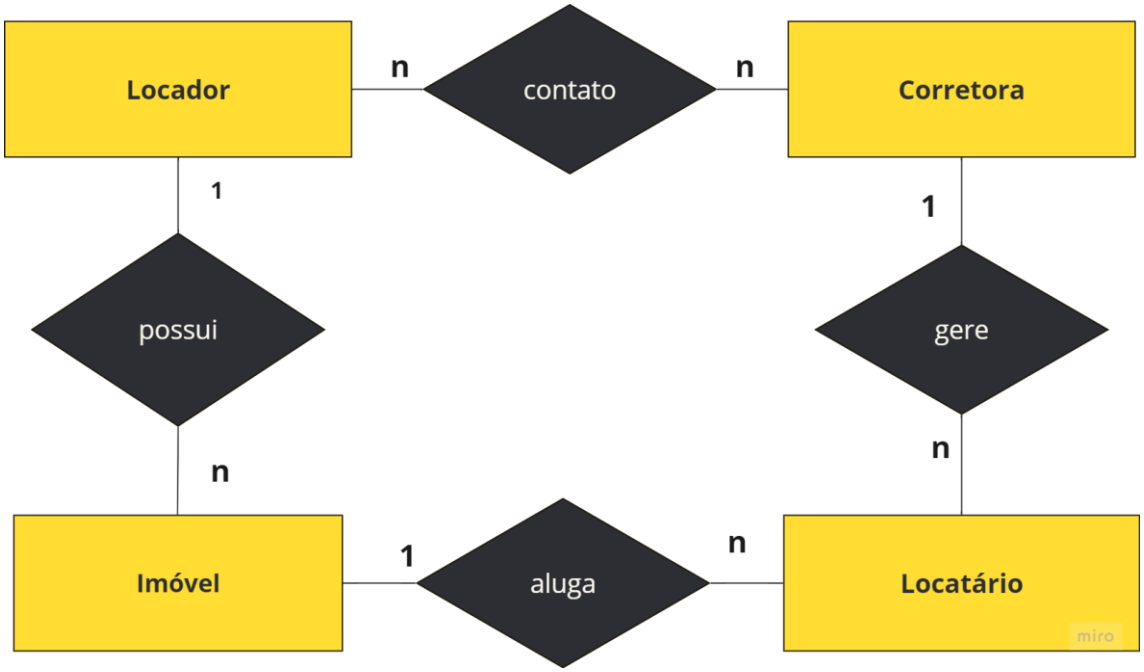
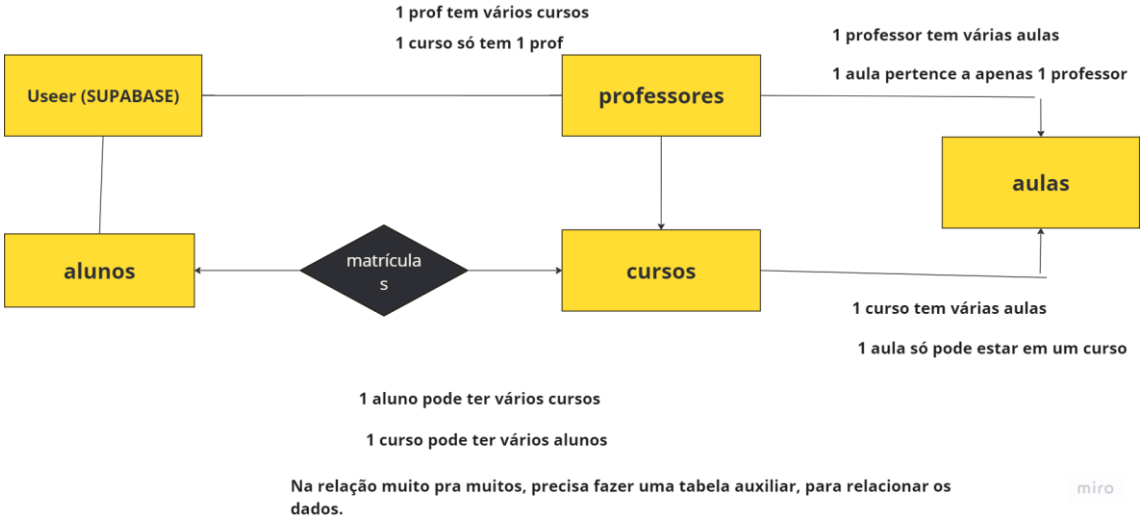


Diagrama de Entidade Relacionamento (DER) do modelo da NoCode StartUp



- Um aluno pode fazer vários cursos diferentes e os cursos podem ter vários alunos (relação n, n);
- O curso tem um professor, e o professor tem vários cursos (relação 1,n);
- Um professor tem várias aulas, uma aula pertence a apenas im professor;
- Um curso tem várioas aulas e uma aula só pode estar em um curso;
- Um aluno pode ter vários cursos, um curso pode ter vários alunos. OBS: Na relação muito para muitos, precisa criar uma tabela auxiliar, para relacionar os dados.
-

10.1.3 Modelo Lógico

Detalhamento das entidades, atributos e relacionamentos:

- Normalização das tabelas: é a otimização das tabelas para reduzir redundãnicas, duplicações e inconsistência dos dados. Com isso, terá um modelo de dados mais consistente, organizado e com maior performance.

- **Primeiro passo: Primeira Forma Normal: NF1** – a tabela deve possuir atributos únicos, não podem existir atributos multivalorados. No banco de dados, temos que ter um valor para cada campo. Por exemplo: endereço, não pode ter: rua, número, bairro e cidade no mesmo campo. Então, deve-se criar um campo para cada valor. O campo telefone, pode ser que um usuário tenha dois números e outros não. Para evitar deixar a tabela com campos vazios, a opção é criar uma tabela separada, só para telefones, e associar os mesmos ao id do usuário.
- **Segundo Passo: Segunda Forma Norma: NF2** - os atributos (não chave, aqueles que têm chave primária ou chave secundária) dependem apenas da chave primária. Ou seja, devem ficar numa mesma tabela, apenas os campos que dependem da chave primparia. Para os que dependem da chave entrangeira, deve ir para outra tabela Tem que seguir a NF1 também.
- **Terceiro Passo: Terceira Forma Normal: NF3 –** os atributos (não chave) devem ser independentes entre si. Tem que segir a NF1 e NF2 também. Por exemplo, tabela com campos: valor da compra e quantidade comprada. Se você colocar uma campo com total da compra, ela dependerá da quantidade comprada e valor da compra. Então, não precisa colocar. Pois, se você precisar do total, numa query, você pode usar as informações de valor da compra e quantidade para achar o total. Quanto mais limpa a tabela, melhor a performance do Banco de Dados.

- Detalhamento dos atributos e relacionamentos
  - Tipos de dados no SQL, veja todos os tipos, clicando no link: https://www.rlsystem.com.br/tipos-dados-sql-server#:~:text=Tipo%20de%20Dados,para%20processamento%20posterior

Veja link para outros cursos:
https://www.rlsystem.com.br/tipos-dados-sql-server

- Construir Modelo Lógico Final.

10.1.4 Ferramentas de Modelagem

MySQL Workbench

Miro: procurar o template: Diagrama ER

Lucidchart:

https://www.lucidchart.com/pages/?

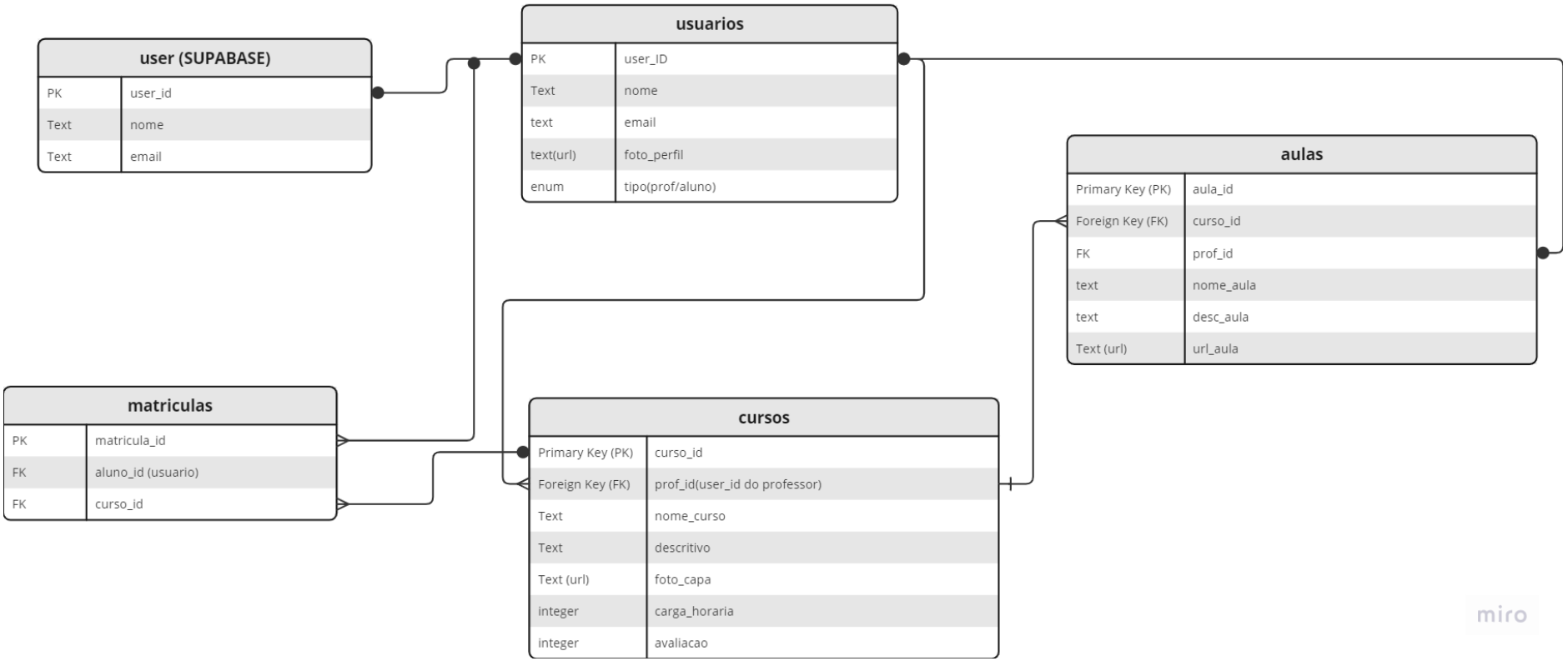
10.1.5 Modelo Físico

Implementação do banco de dados em SGBD por meio do SQL.

- Escolha da Tecnologia SGBD;
- Criar banco com SQL;
- Gestão e menutenção do Banco de Dados.

Pode ser criado nas tecnologias: PostgreSQL, MySQL e SQLite.

Diagrama de Relacionamento de Entidades



11. DESENVOLVIMENTO DO APP NO SUPABASE

As tabelas Cursos, Usuários já foram criadas. Temos que criar mais duas tabelas: Matrículas e Aulas.

11.1 Tabela Matrículas

- New Table;
- Name: matriculas;
- Habilitar Enable Row are required to query data;
- Columns: id e created_at já vêm inseridas;
- Add Column: user_id; colocar chave estrangeira:
    - Clicar na chave do lado;
    - Select a shema: public;
    - Select a table tp reference to: usuários;
    - Select a column from: user_id;
    - Save.
- Add Column: curso_id; colocar chave estrangeira
    - Clicar na chave ao lado;
    - Select a shema: public;
    - Select a table tp reference to: cursos;
    - Select a column from: id;
    - Save.
    - Save
- Add Column: finalizada;
    - Name: finalizada;
    - Data Tyoe: boleano;
    - Default Value: false;
    - Allow Nullable: desmarcar;
    - Save.
    - Save

Inserir dados na tabela matriculas para testar.

- Clicar na Tabela Matriculas;
- Inserir;
- Vá até user_id, clicar em Select record, para buscar o aluno matriculado;
- Colocar o nome e os demais dados, salvar.

11.2 Tabela Aulas

- New Table;
- Name: aulas;
- Deixar o id e created_at;
- Add Column: curso_id; colocar como chave estrangeira, clicando na chave:
    - Select a schema: public;
    - Select a table to reference to: cursos;
    - Curso_id = id
- Add Column: prof_id; colocar como chave estrangeira, clicando na chave:
    - Select a schema: public;
    - Select a table to reference to: usuarios
    - Curso_id = id
- Add Column: aula_nome; tipo: text;
- Add Column: aula_url; tipo: text;
- Save

11.3 Criar Polices para as Tabelas

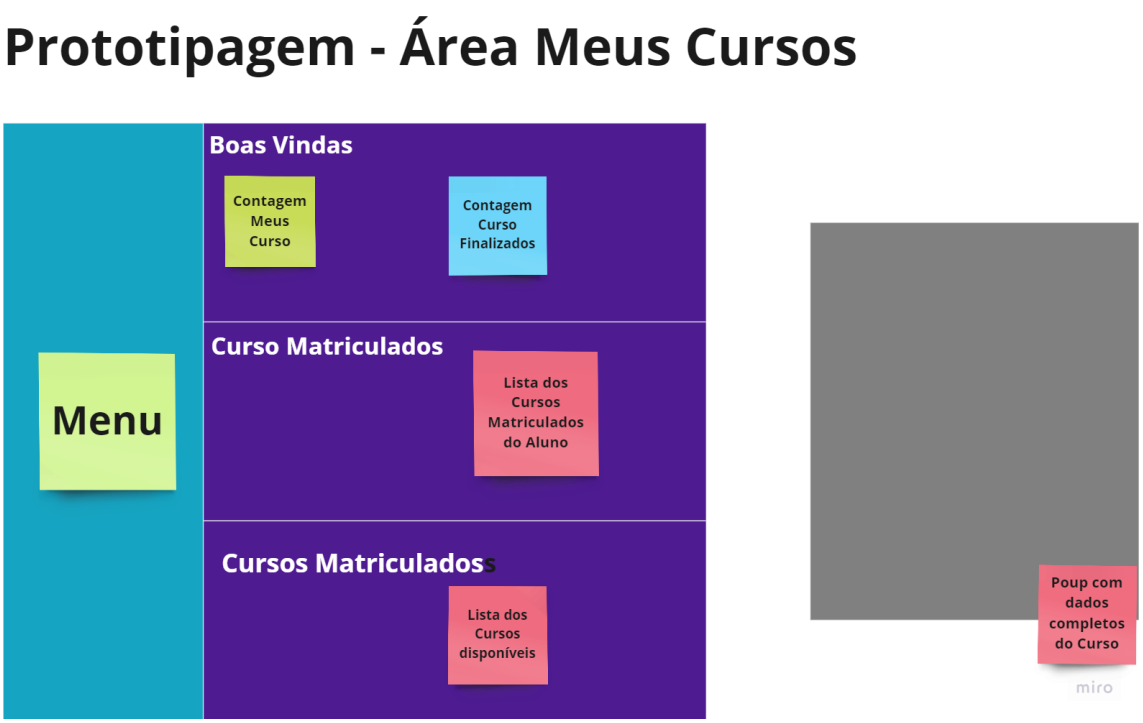
- Clicar na tabela;
- Polices;
- Policy Name: LiberarTudo;

- All;
- True;
- True;
- Save.
- OBS: Fazer isso para todas as tabelas. Na tabela usuário, depois do App pronto precisa ser mudado para usuário \Authe ticates.

12. VIEW

A View permite juntar dados de duas tabelas e visualizar em uma só. Veja a explicação de como fazer no tópico: **Join no SQL – União de Tabelas**

12.1 Prototipagem - Área Meus Cursos



12.2 View para juntar Tabelas:Matrículas, Alunos e Cursos

Fazer uma View para juntar cursos, com alunos e matrículas. Você pode usar a IA do Supabase para criar. Mas, vamos fazer manualmente, para aprender.

- Entrar no Supabase;
- Clicar em SQL Editor;
- New Query;
- --- Coloque esses 3 pontinhos e faça o comentário: view para juntar tabelas: Matrícula, Alunos e Cursos;
- Copie o código da View dados_cursos_prof e cole aqui, para facilitar;
- Faça as alterações necessárias, no código, colado;

```
---View para juntar tabelas: MATRICULAS + ALUNOS + CURSOS
create
  view matr_alunos_cursos as
select
    m.id as matricula_id,
    u.nome as aluno_nome,
    u.user_id as aluno_id,
    u.email as  aluno_email,
    u."fotoPerfil"as aluno_fotoPerfil,
    c.nome as curso_nome,--chamar o curso de c. O nome da tabela, no resultado será curso_nome
    c.id as curso_id;
    c."descCurta" as curso_descCurta, -- a desc_Curta, no resultado será curso_desc_Curta
    c."capaURL" as curso_capa,
    c."descLonga" as curso_descLonga,
    c."nota" as curso_nota

from
  matriculas m
join
  usuarios u  on u.user_id = m.user_id
join
cursos c on c.id = m.curso_id
```

13. SALVAR FOTOS DE PERFIL NO SUPABASE

https://www.youtube.com/watch?v=DNpvL2EW0tc

- Entrar no Supabase;
- Abrir a tabela usuários;
- Add Column;
- Name: foto;

- Tipo de dado: text; (vai armazenar uma url)
- Save

Depois disso, entrar no Storage:

- Create a storage bucket;
- Name: bucket1;
- Habilitar: Public bucket;
- Save;
- Dentro do novo bucket pode-se criar pastas;
- Selecione o bucket;
- Clicar em create folder;
- Name: fotos;

Depois disso criar as regras de segurança

- Na página do bucket criado, clicar em policies;
- No nome do bucket, clicar em New policy;
- For full customization;
- Policuy name: Acesso Geral;
- Allowed Operation: clicar em todas as opções;
- Clicar em Review;
- Clicar em Save Policy.

Depois disso, entrar no FlutterFlow

- Atualize os dados do Supabase: clicar em Get Schema, para atualizar os dados;
- Entrar na Página pagPessoal;
- Clicar no canva;
- Clicar no ícone de maior nível hierárquico, e clicar na BackEnd Query;
- Add Backend Query;
- Query Type: Supabase Query;
- Table: usuários;
- Single Row;
- Filters;

- Filter 1;
- Field Name: uid;
- Relation: Equal To;
- Authenticaded User;
- User ID;
- Confirm.

Depois disso, clique em propriedades

- Clicar no nome: usuarios Row: nome;
- Clicar no email: usuários Row: email;
- Clicar na imagem: Image Type: Network; Path; usuários Row: foto. Coloque um ícone de perpil, no valor padrão.

Depois disso, faze o upload da imagem.

Botão Salvar Imagem

- Clicar no Botão Salvar Imagem;
- Action;
- Open;
- **Add Action – primeira ação;**
- Upload/Save/Media;
- Upload Type: Supabase;
- Bucket Name: Value: bucket1; (nome do bucket criado no Supabase);
- Upload Folder Path: Value: fotos (nome da pasta criada no Supabase);
- Midia Source: Either Camera or Gallery;
- **Add Action = segunda ação**
- Update Row;
- Table: usuários;
- Matching Rows: Filter 1: Field Name: uid; relation: Equal To; Unset; usuários Row: uid;
- Set Field;
- Add Field;
- Foto;
- Value: Widget State: uploaded File URL;

- **Add Action – terceira ação;**
- Refresh Database Request: pagPessoal

14. INTEGRAÇÃO DO FLUTTERFLOW COM SUPABASE

Etapas do Projeto EAD 2.0 FlutterFlow w Supabase

- **Projeto Avançado com Supabase**: Plataforma EAD a partir do Backeend já planejado;
- **Integração Completa FlutterFlow**: funcionalidades avançadas e usuários com diferentes papéis;
- **Painel Administrativo**: professor poderá criar e gerenciar suas aulas;
- **Outras Operações Importantes**: Upload de Imagens e Função de Buscar no Supabase.
- **Link do projeto para clonar**:

  https://app.flutterflow.io/project/plataforma-e-a-d-cursos-d-e-s-i-g-n-qtwa67?

14.1 Integração do FlutterFlow com Supabase

- Entrar no FlutterFlow;
- Configurações;
- App Setting;
- Supabase: habilitar: Enable Supabase. O FlutterFlow vai pedir a API Url e a Chave de acesso;
- Entre no Supabase;
- Configurações;
- Settings;
- API, copie a URL e cole no FlutterFlow;
- No Supabase copie a Chave (Project API Keys: anon) e cole no FlutterFlow. Clique que Get Schema.

14.2 Authentication

- No FlutterFlow, clicar em Authentication;
- Habilitar: Enable Authentication: type: Supabase;
- Página de Entrada: boas-vindas;
- Pagina quando logado: meuCursos.

14.3 Página de Boas Vindas

14.3.1 Login

- Abrir a página boas-vindas;
- Entrar na Aba de Login;
- Clicar no botão Login;
- Action;
- **Add Action – primeira ação**
- Open;
- Supabase Log in;
- Unset: Email;
- Email Field: emailLogin;
- Password Field: senhaLogin;
- **Add Action – segunda ação**
- Navigate to: MeusCursos.

14.3.2 Cadastro

- Abrir a página boas-vindas;
- Entrar na Aba de Cadastro;
- Clicar no botão Começar;
- Action;
- **Add Action – 1ª Ação**
- Supabase: Create Account;
- Unset: Email;

- Email Field: emailCadastro;
- Password Field: senhaCadastro;
- Confirm Password Field: senhaConfirmação;
- **Add Action – 2ª Ação – Esta ação é para incluir o cadastrado na tabela usuário;**
- Supabase: Insert Row;
- Table: usuários;
- Set Field;
- Add Field: user_id;
- Value: Authenticatio User;
- User ID;
- Add Field: nome;
- Value: Widget Station;
- NomeCadastro (Nome);
- Add Field: email;
- Value: Authentication User;
- Email.

14.3.3 Página Meus Cursos – Drawer

Para abrir a Drawer, entrar na página Meus Cursos, Clicar na widget Tree e clicar na Drawer. Em propriedades, clicar em editar Drawer.

- Clicar no ponto mais alto do Componente, Container abaixo da Drawer;
- BackEnd Query;
- Add Query;
- Supabase Query;
- Table: usuários;
- Single Row;
- Filter: Filter 1: user-id; Relation: Equal To; Value: Authenticated User: User_id; Confirm

14.3.4 Componente MenuSideBarExpandido

- Entrarno Componente;
- Clicar no email do usuário;
- Propriedade: Text; Authenticated User: email

Para chamar os demais dados do usuário, precisa fazer uma BackEnd Query para buscar os dados no Banco de Dados.

- Clicar no ponto mais alto do Componente, Container abaixo doctn MeuSideBarExpandido;
- BackEnd Query;
- Add Query;
- Supabase Query;
- Table: usuários;
- Single Row;
- Filter: Filter 1: user-id; Relation: Equal To; Value: Authenticated User: User_id; Confirm

  OBS: Depois que fizer os procedimentos dos Tópicos 21 DATA TYPE, 22 – DATA TYPES E 23 API CALL da Página de Boas Vindas, você pode excluir a BackEnd acima, pois serão procedimentos idênticos, para não sobrecarregar o banco de dados.

Depois disso, colocar os dados de forma dinâmica:

- **Clicar em nome;**
- Text;
- Usuários Row;
- Nome;
- Default: sem nome (melhor não deixar este dado em branco);
- Confirm;
- **Clicar na Cicle Image;**
- Path;

- Usuários row;
- Foto de Perfil;
- Dafault: colocar uma padrão;
- Confirm.

OBS: Fazer o mesmo procedimento para o Componente MeuSideBarReduzido e para o Meu Perfil.

14.3.5 Página MeusCursos – Cursos Disponiveis

Colocar Inteligência na pagina para mostrar os cursos disponíveis.

- Clicar na Página Meus Cursos;
- Clicar na LIstView dos Cursos Disponíveis;
- Clicar em BackEnd Query;
- Add Query;
- Supabase Query;
- Table: cursos;
- List of Row; (não tem filtro, pois quer mostra todos os cursos.)
- Cofirm;
- Confirm.

Depois disso, colocar os dados de forma dinâmica

- **Clicar na image;**
- Path;
- Cursos Row;
- Capa Url;
- Default: deixe sempre uma foto padrão;
- Confirm;
- **Clicar no nome;**
- Text;
- Cursos Row;
- Nome.

Para mostrar os detalhes do curso, quando clicar no nome:

- Clicar no container abaixo da ListView;
- Action;
- Add Action;
- Bottom Sheet;
- Select Component: Aluno_CursosDisponiveis.

Para abrir Pop Up, para transferir informação para outra página, precisa passar o id do curso. Só assim, consegue fazer uma busca no banco de dados.

Entrar na página que você quer buscar os dados, nesse caso, seria o Componente Aluno-CursosDisponiveis:

- Clicar no segundo container do componente;
- + Component parameters;
- Add Parameter;
- Parameter Name: cursoId;
- Type: Interger;
- Required: habilitar;
- Confirm;
- **No mesmo Container colocar uma BackEnd Query;**
- Add Query;
- Type: Supabase Query;
- Table: cursos;
- Single Row;
- Filter;
- Filter name:id;
- Equal to;
- cursoId;
- Confirm.

Depois disso, voltar na Página Meus Cursos, para passar esse parâmetro;

- Clicar no container abaixo da ListView;
- Action; (que já havia sido criada);

- Pass;
- Clicar em cursoID;
- Value: cursos Row: id.

Depois disso, colocar os dados do Componente AlunosCursoDisponiveis, de forma dinâmica:

- Clicar no nome curso;
- Text;
- Cursos Row;
- Nome;
- Confirm;
- Fazer o mesmo com os demais dados;

14.3.6 Página Meus Cursos – Cursos Matriculados

- Abrir a Página Meus Cursos;
- Clicar na ListView Cursos Matriculados;
- BackEnde Query;
- Add Query;
- Supabase Query;
- Table: matr_alunos_cursos (View que fizemos no Supabase);
- List of Row;
- Filter;
- Field Name: aluno_id;
- Relation: Equal to;
- Value: Authentication: User_ID;
- Confirm;
- Confirm.

Depois disso, colocar os elementos de forma dinâmica

- **Clicar na image;**
- Path;

- matr_alunos_cursos Row;
- Url Capa;
- **Clicar no nome;**
- Text;
- Matr_alunos_cursos Row;
- Nome_curso.

14.3.7 Componente ProfAluno-DetalhesCursos

Colocar lógica, para que, quando clicar em Cursos Matrículados, na Página Meus Cursos, abra os detalhes do curso selecionado.

- Abrir o componente ProfAluno-DetalhesCursos;
- Clicar no elemento mais alto do nível hierárquico, para colocar um parâmentro;
- Add Parameter;
- Parameter Name: cursoID;
- Type: Integer;
- Confirm;

Depois disso, clicar no segundo container, para fazer a chamada:

- BackEnd Query;
- Add Query;
- Supabase Query;
- Table: dados_curso_prof;
- Single Row;
- Filter;
- Curso_id;
- cursoID;
- Confirm.

Depois disso, colocar os elementos de forma dinâmica:

- Clicar no nome do curso;

- Text;
- Dados_curso_prof Row;
- Curso_nome;
- Default: sem nome;
- Confirm;
- Fazer isso, com todos os elementos.

Fazer consulta das aulas

- Depois da descrição longa, tem as aulas disponibilizadas. Para visualizar, deve-se colocar uma BackEnd Query, para fazer uma consulta no Banco de Dados:
- Clicar na ListView das Aulas;
- BackEnd Query;
- Supabase Query;
- Table: aulas;
- List of Row;
- Filters;
- Curso_id;
- Equal To;
- cursoID;
- Confirm;
- Confirm.

Depois disso, vá até a Página Meus Cursos:

- Clicar no primeiro container abaixo da ListView;
- Action;
- Bottom Sheet;
- Select Component;
- ProfAluno-DetalhesCurso;
- Pass Parameter;
- cursoID;
- Value: matr_alunos_cursos Row;

- Curso_id;
- Confirm.

14.3.8 Página Meus Cursos – Botão Matricula

- Entrar na Página Meus Cursos;
- Clicar no botão matrícula;
- Action;
- Add Action;
- Bottom Sheet;
- Schow;
- Select Comppnent;
- Aluno-Detalhes_Matricula.

Depois disso, ir para o componente Aluno-DetalhesMatricula;

- Passar um parâmentro: define parameter;
- Parameter name: matriculaID;
- Type: Interger;
- Save;

Voltar à Página Meus Cursos, na Action do botão Matrícula, e passar o parâmetro do Componente Aluno-DetalhesMatricula:

- Clicar em Pass;
- Paramenter: matriculaID;
- Value: matr_alunos_cursos Row;
- Matricula_id;
- Confirm;

Em seguida criar uma BackEnd Query, para buscar os dados no Banco de Dados:

- Clicar no segundo container do Componente;
- BackEnd Query;

- Supabase Query;
- Table: matriculas;
- Single Row;
- Filter;
- Filter name: id (id da matrícula);
- Equal To;
- Value: matriculaID;
- Confirm.

Depois disso, colocar os dadis de forma dinâmica:

- **Clicar no número da matrícula;**
- Text;
- Matriculas Row;
- Id;
- Confirm;
- Clicar em Finalizada;
- Text;
- Matriculas Row;
- Finalizada;
- Default: false;
- Confirm.

14.3.9 Componente Alunos-DetalhesMatricula – Botão Marcar Matrícula como Concluída

- Clicar no botão;
- Action;
- **Add Action – primeira ação;**
- Supabase Update Row;
- Table: matriculas;
- Matching Rows;
- Filter: id;

- Equal to;
- matriculaID;
- Set Field;
- Field: finalizada;
- True;
- **Add Action – segunda ação;**
- Alert Dialog;
- Informational Dialog;
- Value: Curso Completado;
- Value: Matrícula Finalizada;
- Ok;
- **Add Action – Terceira ação;**
- Bottom Sheet;
- Dismiss.

OBS: O botão “Marcar Matrícula como Concluída”, só será mostrado para o usuário, se o curso ainda não tiver sido completado, para isso, você precisa colocar uma condicional:

- Clicar no botão, em propriedades, habilitar o Conditional;
- Unset;
- Single Condition;
- First Value: Unset: matriculas Row: finalizada;
- Equal To;
- False

14.3.10 Componente Aluno-CursosDisponíveis

Neste componente os dados já foram colocados de forma dinâmica, em procedimentos anteriores. Agora, vamos colocar ação no botão Realizar Matrícula.

- Clicar no botão;
- Action;

- **Add Action – primeira ação**
- Supabase Insert Row;
- Table: matriculas;
- Set Field:
    - Add Field:user_id; Authenticated User: User_id; Confirm;
    - Add Field: curso_id ; Value: cursoId
- **Add Action – segunda ação**
- Dialog Alert;
- Informational Dialog;
- Value: Matrícula Realizada!
- Value: Você já pode acessar seu curso!
- **Add Action – terceira ação**
- Bottom Sheet – Dismiss.

14.3.11 Componente Aluno-DetalhesAula

- Clicar no container Aluno-DetalhesAula, para criar um parâmetro;
- Component Parameters +;
- Parameter name: aulaID;
- Type: Integer;
- Confirm;

Clicar no segundo container para fazer uma BackEnd Query, consulta ao Banco de Dados:

- BackEnd Query;
- Add Query;
- Supabase Query;
- Table: aulas;
- Single Row;
- Filter;
- Id;

- Equal To;
- aulaID;
- Confirm.

Colocar os dados de forma dinâmica

- Clicar no nome da aula;
- Text;
- Aulas Row;
- Aula_nome;
- Confirm
- Clicar no vídeo;
- Propriedades;
- YouTube Player;
- URL;
- Aulas Row;
- Aula_url;
- Confirm.

Depois disso, voltar ao Componente ProfAluno-DetalhesCurso, para passar o parâmetro:

- Abrir o Componente;
- Clicar no botão “assistir”;
- Action;
- Add Action;
- Bottom Sheet;
- Show;
- Select Component: DetalhesAula;
- Pass;
- aulaID
- Value: aulasRow;
- Id;
- Confirm.

Depois disso, colocar os dados de forma dinâmica:

- Clicar no nome da aula;
- Propriedade;
- Text;
- Aulas Row;
- Aula_nome;
- Confirm

15. PÁGINA ÁREAPROFESSOR

Nesta página serão mostrados os cursos que o professor criou.

15.1 List View

- Abrir a página areaProfessor;
- Clicar na ListView dos Meus Cursos Criados;
- BackEnd Query;
- Add Query;
- Supabase Query;
- Table: dados_curso_prof;
- List of Row;
- Filter (colocar um filtro para mostrar os cursos do professor logado);
- prof_id;
- Equal To;
- Authenticaded User: User_id;
- Confirm.

Depois disso, clicar no primeiro container abaixo da ListView, para colocar ação de levar para o componente: ProfAluno-DetalhesCursos;

- Clicar no container;
- Action;
- Add Action;
- Bottom Sheet;

- Show;
- Select Component: ProfAluno-DetalhesCursos;
- Pass Parameter;
- cursoId;
- Value: dados-curso_prof Row;
- Curso_id.

Depois disso, entrar no Componente "MenuSideBarExpandido":

- Clicar na Área do Professor;
- Action;
- Add Action;
- Navigation To;
- AreaProfessor.

Em seguida, colocar os dados da Página AreaProfessor de forma dinâmica.

- Na ListView, clicar na image;
- Path;
- Dados_Curso_prof_Row;
- Curso_capa;
- Confirm;
- Clicar no nome do Curso;
- Text;
- Dados_curso_prof Row;
- Curso_nome;
- Confirm.

15.2 Botão + Novo - Criar Novo Curso

- Clicar no botão + Novo;
- Action;
- Add Action;
- Bottom Sheet;

- Select Component: Prof-AdicionarCurso;

Em seguida clicar no Componente Prof-AdicionarCurso.

15.3 Botão Editar (editar aulas do professor)

- Abrir a Página areaProfessor;
- Clicar no botão Editar;
- Action;
- Add Action;
- Bottom Sheet;
- Show: Select Component: Prof-Editar-Curso.
  Depois entrar no Componente Prof-Editar-Curso

16. COMPONENTE PROF-ADICIONARCURSO

16.1 Botão Adicionar

- Clicar no botão;
- Action;
  Add Action – primeira ação;
- Supabase Insert Row;
- Table: cursos;
- Set Field: nome; Value: Widget State: nomeCurso; Add Field: capaUrl; WidgetState: capaUrl. Faça o mesmo com> descrição curta, descrição longa, nota, carga horária para o prof_id: Value: Authenticated: User_id;
  Add Action – segunda ação:
- Informational Dialog;
- Value: Show; Curso criado com sucesso;
  Add Action – terceira ação:
- Bottom Sheet: Dismiss.

17. COMPONENTE PROF-EDITARCURSO

- Abrir o componente;
- Clicar no nome do componente, para criar um Parâmentro;
- Clicar em Component parameter:
- Name: cursoID;
- Type: Interger;
- Save;

Depois, criar uma BackEnd Query, para buscar os dados no Banco de Dados:

- Clicar no segundo container;
- BackEnd Query;
- Add Query;
- Supabase Query;
- Table: cursos;
- Single Row;
- Filter: id;
- Equal to;
- cursoID;
- Confirm.

Depois, colocar os elementos do componente de forma dinâmica:

- Clicar Nome do Curso – Propriedades;
- Initial Value;
- cursoRow: nome;
- Clicar na descrição curta;
- Initial Value;
- cursoRow: descCurta;
- Repetir o mesmo processo para os demais dados.

17.1 Botão Deletar

- Clicar no botão deletar;
- Action;
- Add Action;
- BackEnd Call Delete;
- Action Type: Delete Row;
- Table: Cursos;
- Matching Rows;
- Add Filter;
- Field Name: id;
- Relation: Equal To;
- cursoId;
- Add Action – segunda ação
- Dialog Confirm;
- Value: Você realmente quer deletar todas as informações do Curso?
- Ok;
- **Add Action - segunda ação**
- Informational Dialog;
- Value: Curso deletado!
- Feito!

17.2 Botão Atualizar

- Clicar no botão Atualizar;
- Action;
- Open;
- Add Action;
- Supabase Update Row;
- Table: cursos;
- Matching Rows;
- Add Filter;
- Id;

- Equal To;
- cursoID;
- Set Field;
- Nome: Widget State: nomeCurso;
- CapaURL: Widget State: capaUrl;
- Repetir para todos os elementos: descCurta, descLonga, nota e carga_horária.
- **Add Action – segunda ação;**
- Informational Dialog;
- Value: Curso Editado;
- Feito.
- **Action – terceira ação**
- Bottom Sheet
- Dismiss.

18. COMPONENTE PROF-ADICIONARCURSOAULA

Antes de criar as ações no FlutterFlow, precisa criar as Polices no Supabase, permitindo alterar tudo

Antes de entrar no componente, faça o que se manda:

- Entrar na Área do Professor Adicionar uma Aula ao Curso

- Entrar na Área do Professor;
- Clicar em Aulas;
- Action;
- Add Action;
- Bottom Sheet;
- Show;
- Select Component;
- Prof-AdicionarCursoAula;

  Depois disso abrir o componente: “Prof-AdicionarCursoAula:

- **Clicar no nome para criar um Parâmetro:** Parameter Name: cursoID; Type: Integer; Confirm.
- **Depois disso,**
- Clicar no botão Adicionar;
- Action;
- **Add Action – primeira ação;**
- Supabase Insert Row;
- Table: aulas;
- Set Field;
- Aula_nome;
- Value: Widget State: aula_nome;
- Url_aula: Widget State: vídeo_url;
- Prof_id: Authenticated User: prof_id;
- Curso_id: Value: cursoID;
- **Add Action – segunda ação**
- Informational Dialog;
- Aula criada com sucesso!
- Mandou bem

19. PÁG MEUSCURSOS: TOTAL CURSOS E CURSOS FINALIZADOS

- Abrir a Página Meus Cursos;
- Clicar no primeiro container, abaixo do container ResultadosCards;
- BackEnd Query;
- Add Query;
- Supabase Query;
- Table: matriculas;
- List Rows;
- Filter;
- Field name: user_id; Equal To; Authenticated User: User_ID
- Confirm;

Depois disso, colocar os dados de forma dinâmica

- Clicar no número de cursos;
- Text;
- matriculasRows;
- Avaible Options: Number of Items;
- Confirm.
- Abrir a Página Meus Cursos;
- Clicar no primeiro container ctnCursoFinalizados;
- BackEnd Query;
- Add Query;
- Supabase Query;
- Table: matriculas;
- List Rows;
- Filter;
- Field name: user_id; Equal To; Authenticated User: User_ID;
- Add Filter;
- Filter 2: finalizada; Equal: True;
- Confirm;

Depois disso, colocar os dados de forma dinâmica

- Clicar no número de cursos;
- Text;
- matriculasRows;
- Avaible Options: Number of Items;

Confirm.

20. BLOQUEAR O ACESSO DO ALUNO À ÁREA DO PROFESSOR

Se o usuário for do tipo aluno, não pederá acessar à área do professor, que é uma área do Administrativo. Para isso, será necessário criar uma API.

- Clicar na API Calls, para buscar os dados do usuário;
- API Call Name: GET Detalhes Usuario;

Entrar no Supabase:

- Planilha: usuários;
- API Doc;
- Read Row;
- Language: Bash;
- Filtering (que será o user_id do usuário logado);
- Copiar a curl e colar no FlutterFlow na linha GET:
  https://wocoghssbzkygoxjrdcq.supabase.co/rest/v1/usuarios?user_id=eq.[userID]
  No get o filtro tem que ser do usuário logado, por isso mudou o id=eq. Para:user_id=eq.[userID]. Pode tirar também o select que vem no final do código.
- Depois clicar em Variable para criar a variável: [userID]:
  - Clicar em Variable;
  - Add Variable;
  - Name: userID;
  - Type: string;
  - Is list: False
- Clicar em Headers; Add Header: escreva: apikey: e dar espaço e copiar a chave API do Supabase e colar aqui:
  apikey:
  eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6IndvY29naHNzYnpreWdveGpyZGNxIiwicm9sZSI6ImFub24iLCJpYXQiOjE3MTU5Nzk1NjksImV4cCI6MjAzMTU1NTU2OX0.tlPOOZeUTkFoEur0anCeFCYtu56YvsFsPeomTcrVZpY

- Add Header para passar o outro header de Authorization:
  Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6IndvY29naHNzYnpreWdveGpyZGNxIiwicm9sZSI6ImFub24iLCJp

YXQiOjE3MTU5Nzk1NjksImV4cCI6MjAzMTU1NTU2OX0.tlPOOZeUTkFoEur0anCeFCYtu56YvsFsPeomTcrVZpY

- Antes de salvar precisar testar se deu certo:
- Entre no Supabase e copie o número de um usuário;
- Colar esse número no Value: ao lado da Variable userID;
- Clicar em Response Test;
- Test API Call
- Se o teste der certo siga;
- Em Response Test, clicar em JSON Paths;
- Clicar em + Add JSON Path, faça isso para toda a lista que se apresenta. Assim, você terá acesso a esses dados;
- Faça a alteração nos names:
    - Add Name: userID;
    - Add Name: createdAt;
    - Nome;
    - Email.
    - fotoPerfil;
    - tipo
- Add Call.

20.1 Área do Professor – limitando acesso

- Abrir Componente Menu SideBarExpandido;
- Clicar na Área do Professor;
- Action;
- Add Action;
- API Call;
- GET Detalhes Usuário;
- Variable:
- Unset: userID;
- Value: Authenticated: User ID;

- Action Output Vairable Name: apiResultGetUsuarioLogado;
- **False**: Add Action: Informational Dialog: Value:Action Ouputs: apiResultGetUsuarioLogado; Unset: Row Body Text; Fechar
- **TRUE: Add Condition:**
  - Single Condition;
  - First Value: Unset: Action Output: apitResultGetUsuarioLogado: Avaible options: API Response Options: Unset: JSON Body: Predifined Path: Path: tipo; Save
- Second Value: professor;
- Confirm;
- **TRUE**
- Add Action: Navigation to: Área professor;
- Add Terminate
- **FALSE: Add Action;**
- Informational Dialog: Você não tem permissão para acessar;
- Value: Área exclusiva para professores;
- Add Terminate.

OBS: Você deve repetir essa ação nos Menus:

- MenuSideBarReduzido, no container MeuTime;
- Menu SideBarTablet, no containerMeu Time;
- Drawer, no container Meu Time.

21. DATA TYPES

Pode ser usado para agrupar dados. Exemplo: você cria uma variável chamada “endereço” e dentro dessa variável você coloca vários campos: rua, bairro, número, cidade, estado. Você também pode criar uma variável “usuário” e colocar vários campos, que identifiquem o usuário, mas, precisará fazer apenas uma chamada.

Criando o data type: user:

- Clicar em Data Type – menu lateral do FlutterFlow;
- + Create a New Data Type;
- Name: user;
- Create;
- Clicar em Add Field, para definir os campos:
    - Field name: nome;
    - Type: string;
    - Is List: não habilitar;
    - Add Field. Field name: email;
    - Type: String;
    - Add Field. Field name: fotoPerfil;
    - Type: Image Path;
    - Is List: não habilitar;
    - Add Field. Field name: tipo;
    - Type: String;
    - Is List: não habilitar;

22. API STATE

App State são formas de você conseguir acessar dados, sem precisar fazer nova chamada.

Depois de criar o Data Type **user**, entrar no API State, para fazer a chamada.

- Clicar em App Valuer, menu lateral;
- Add App State Variable;
- Field Name: user;
- Date Type: Data Type (o que foi criado no item anterior);
- Type: User;
- Persisted: habilitar;
- Create.

-

23. PÁGINA DE BOAS VINDAS

Voltar à págima de Boas Vindas para fazer algumas otimizações.

- Abrir a página de login e clicar no botão login;
- Action;
- Delete a segunda ação que já foi feita, deixando apenas a de login;
- **Add Action – segunda ação;**
- API CALL: Unset: GET Detalhes Usuario;
- Action Output Variable; name: apiResultDetalhesUser;
- Set Additional Variable: userID;
- Value: Authenticated User: User ID;
- **FALSE: se o resultado der falso;**
- **Add Action – terceira ação**
- Informational Dialog;
- Value: Erro;
- Value: Action Output;
- apiResultDetalhesUser;
- Unset: Raw Body Text;
- Confirm;
- **Terminate;**
- **TRUE**
- **Add Action – quarta ação**
- Update App State;
- Add Field;
- User;
- Unset: Set Value;
- Unset: user (data type);
- Add Field: nome; Action Output; apiResultDetalhesUser; Uset: JSON Body; Predefined Path: nome; Confirm;
- Repetir o processo para email, fotoPerfil e tipo;

- Confirm;
- **Add Action – quinta ação;**
- Navigation to;
- Meus Cursos.
- Terminate.

Depois disso, entrar no componente, MenuSideBarExpandido:

- Clicar no nome do usuário. Tirar o que estava definido antes, nas propriedades e colocar o que ser segue;
- Text;
- App State;
- User;
- Data Structure Field;
- Unset: nome;
- Confirm;
- Fazer o mesmo com o email e imagem.

24. PÁGINA MEUS CURSOS – DADOS DE FORMA DINÂMICA

- Entrar na Página Meus Cursos;
- Clicar no texto de bem-vindo, na parte superior do App;
- Text;
- Combine Text;
- Text 1: Bem-vindo, (precisa colocar espaço depois da vírgula);
- Text 2: App State; user; Avaible Options; Date Structure Field: unset: nome; Confirm.

25. COMPONENTE MENUSIDEBARREDUZIDO – Dados dinâmicos

- Clicar na circleimage;

- Path:
- App State;
- User;
- Data Structure Field;
- Unset: fotoPerfil.

26. COMPONENTE MENUSIDEBARTABLET – Dados dinâmicos

- Clicar na circleimage;
- Path:
- App State;
- User;
- Data Structure Field;
- Unset: fotoPerfil.

OBS: Fazer o mesmo procedimento no componente Meu Perfil.

27. UPLOAD DE MIDIAS

- Para fazer uploa de mídia, você precisar ativar o Storage, no Supabase:
  - Entrar no Supabase;
  - Criar um bucket;
  - Criar uma pasta, clicando em Create Folder, como o nome fotos, para guardar as imagens;
- Voltar para o FlutterFlow;
- Abrir o componente Meu Perfil;
- Clicar no botão Upload Foto;
- Action;
- **Add Action – primeira ação**
- Upload Save Media;

- Upload Type: Unset: Supabase;
- Bucket Name: bucket2; Upload Folder Path: Value: foto (o nome que você deu no Supabase);
- **Add Action – segunda ação**
- Update Raw;
- Table: usuários;
- Matching Rows: Filter 1: Field Name: user_id
- Equal To;
- Value: Authenticated User: User ID;
- Set Fields:
- FotoPerfil;
- Widget State;
- Value: Upload File Url;
- **Add Action – terceira ação**
- Update App State;
- Add Field: user;
- Select Update Type: Update Field;
- Add Field;
- fotoPerfil;
- Value: Widget State: Uploaded File URL;
- Done.

Colocar os dados do Componente Meu Perfil de forma dinâmica.

As propriedades criadas para o nome e email, anteriormente, podem ser deletadas e colocar as que se seguem. Isso, para não ter procedimentos repetidos:

- Clicar no nome;
- Text;
- App State;
- User;
- Avaible Options: Data Structure Field;

- Select Field: nome;
- Confirm.

Repetir para o email.

28. COMPONENTE BUSCARCURSO

Aprenda a buscar os cursos que estão sendo oferecidos.

- **Entrar no Supabase:**
- Abrir a Tabela Cursos;
- Clicar em API Docs (nas APIs você pode filtrar por: igual, maior que, menor que, like e ilike. Neste procedimento será usado o “ilike”)
- Clicar em Filtering;
- Colocar em Bash;
- Copiar a URL:
- https://wocoghssbzkygoxjrdcq.supabase.co/rest/v1/cursos?id=eq.1&select=*

- **Entrar no FlutterFlow:**
- Clicar em API Calls;
- Clicar na API criada anteriormente: GET Detalhes Usuario e duplicar
- Dar o nome na API copiada: GET Buscar Cursos;
- Copiar a URL no Supabase e colar e fazer as adequações, ficando assim: https://wocoghssbzkygoxjrdcq.supabase.co/rest/v1/cursos?nome=ilike.*[nomeCurso]*

Nos Headers, já estão prontos, pois foi duplicado. Ficando assim:
Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6IndvY29naHNzYnpreWdveGpyZGNxIiwicm9sZSI6ImFub24iLCJpYXQiOjE3MTU5Nzk1NjksImV4cCI6MjAzMTU1NTU2OX0.tlPOOZeUTkFoEur0anCeFCYtu56YvsFsPeomTcrVZpY

apikey:
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpc3MiOiJzdXBhYmFzZSIsInJlZiI6IndvY29naHNzYnpreWdveGpyZGNxIiwicm9sZSI6ImFub24iLCJpYXQiOjE3MTU5Nzk1NjksImV4cCI6MjAzMTU1NTU2OX0.tlPOOZeUTkFoEur0anCeFCYtu56YvsFsPeomTcrVZpY

- Criar a variável: nomeCursos; Type: string;
- Testar: clicar em Response Test e Test API CALL;
- Se der tudo certo, salve os JSON Paths, deixar apenas:
  - createdAt;
  - nome;
  - capaURL;
  - descCurt;
  - descLonga;
  - nota;
  - cargahoraria;
  - profid;
  - curso_id (onde está “id”, colocar cursoid);
- Save.

Depois disso, entrar no Componente Buscar Curso

- Clicar na ListView;
- BackEnd Query;
- Add Query;
- API Call;
- GET Buscar Curso;
- Variable: + Set Additional Variable: Unset; nomeCurso; Value: Widget State: BuscarCursoInput;
- Confirm.

OBS: como é uma chamada API e trabalhando com lista, deve-se usar o Generate Dynamic Children (ícone do lado do BanckEnd Query):

- Clicar no Generate Dynamic Children;

- Variable Name: BuscarCurso;
- Value: Unset: GetBuscarCursosReponse;
- JSON Body; JSON Path, Custom, JSON Path: $;
- Confirm;
- Save;
- Ok.

Depois disso, colocar a imagem de forma dinâmica:

- **Clicar na imagem;**
- Path;
- BuscarCurso Item;
- Available Options: JSON Path;
- JSON Path: $.capaURL;
- Confirm;
- **Clicar no nome do curso;**
- Text;
- BuscarCurso Item;
- Available Options: JSON Path;
- JSON Path: $.nome;
- Confirm.

Depois disso, clicar no primeiro Container abaixo da ListView, para colocar uma ação pra levar à página DetalhesCursos:

- Clicar no container;
- Action;
- Add Action;
- Bottom Sheet: Show;
- Select Component: Aluno-CursosDisponíveis;
- Passa parâmetro: clicar em Pass;
- cursoID;
- Value: Buscar Curso Item;

- JSON Path;
- JSON Path: $.id;
- Confirm

OBS: Clicar na linha da busca e certificar qua a opção Update Component On Text Change esteja clicado. Senão não irá funcionar.

29. PUBLICAÇÃO DO APP

- Entrar no FlutterFlow;
- Configurações;
- Clicar em Pataforms;
- Habilitar: Android, IOS e Web;
- Clicar em Web Publishing (O plano pago permite: tirar a marca d'água, desbilitando: Show Watermarke, inserir um Favicon, de sua marca;
- Site url: marcialino.flutter....
- SEO Title: Título que identifique o seu site;
- Site description: descrição do site;
- Page Title: Nocoflix;
- Deixar ativado: PWA (o que possibilita o App rodar no celular);
- Publish.