17 — Primeira tela

17 — Primeira tela

> Receita end-to-end: uma tela de busca + cadastro de `Fornecedor`, tocando todas as camadas. Copie, troque `Fornecedor` pela sua entidade e o id do caso de uso. Ao final há um checklist de verificação — a tela só está pronta quando ele passa.

O que vamos construir

Uma tela que:

1. Abre um caso de uso no backend
2. Busca fornecedores por filtro
3. Exibe o resultado em tabela
4. Salva um fornecedor novo

Camadas tocadas: `constants` → `interfaces` → `hooks` → `schemas` → página → `paths` → `routes` → `menuTree`.

Pré-requisitos

Do backend, você precisa saber:

- **Id do caso de uso** (ex.: `"4821"`)
- **Nomes dos requests** (ex.: `RM_INCLUI_OBJETO`, `RM_BUSCA_FORNECEDORES`, `RM_SALVA_OBJETO`)
- **Formato dos payloads**

Não invente. Pergunte a quem implementou o caso de uso ou descubra com `enableLogs` (06).

Sem esse contrato ainda? Construa com placeholders

Se o backend real não existe ou ainda não foi definido, **não pule esta receita** — construa-a mesmo assim, com todo id de caso de uso e nome de RM como placeholder explícito (`"SUBSTITUA_USE_CASE_ID"` , `"RM_SUBSTITUA_SALVAR"`), numa entidade que não possa ser confundida com domínio real (`Exemplo` , não `Fornecedor`). Isso é o passo 11 do bootstrap (02) — prova, com `tsc` / `lint` / `build` reais, que a cadeia inteira compila e roteia antes de qualquer feature de negócio existir, e dá ao time algo executável para copiar. Comente no topo do arquivo que é placeholder, registre a rota normalmente, e apague quando a primeira tela de negócio nascer.

1. Constantes

```
// src/pages/Cadastro/Fornecedor/service/constants.ts
export const FORNECEDOR_RMS = {
  USE_CASE: "4821",
  OBTEM_DADOS: "RM_INCLUI_OBJETO",
  BUSCAR: "RM_BUSCA_FORNECEDORES",
  SALVAR: "RM_SALVA_OBJETO"
};
```

Ver 06. Nenhum id ou nome de request literal fora deste arquivo.

2. Interfaces

```
// src/pages/Cadastro/Fornecedor/service/interfaces.ts
export interface FornecedorXML {
  _OID: string;
  _Nome: string;
  _CNPJ: string;
```

```
  _Ativo: boolean;
  TipoFornecedor?: TipoFornecedorXML;
}
export interface TipoFornecedorXML {
  _OID: string;
  _Titulo: string;
}
// abertura: backend devolve objeto novo + listas de apoioexport interface
FornecedorInicialResponse {
  Fornecedor: FornecedorXML;
  TiposFornecedor: TipoFornecedorXML[];
}
export interface BuscaFornecedoresRequest {
  OBJECTID: {
    _Nome: string;
    _Ativo: boolean;
  };
}
export interface BuscaFornecedoresResponse {
  Response: FornecedorXML[];
}
export interface SalvaFornecedorRequest {
  Fornecedor: {
    _OID: string;
    _Nome: string;
    _CNPJ: string;
    TipoFornecedor?: { _OID: string };
  };
}
```

3. Hooks de caso de uso

```
// src/pages/Cadastro/Fornecedor/service/hooks.tsimport { useCurioMutation } from
```

```
"@/hooks/useCurioMutation";
import { FORNECEDOR_RMS } from "./constants";
import {
  BuscaFornecedoresRequest,
  BuscaFornecedoresResponse,
  FornecedorInicialResponse,
  SalvaFornecedorRequest
} from "./interfaces";
export const useIncluiFornecedor = () => useCurioMutation<FornecedorInicialResponse,
void>(FORNECEDOR_RMS.OBTEM_DADOS);
export const useBuscaFornecedores = () =>  useCurioMutation<BuscaFornecedoresResponse,
BuscaFornecedoresRequest>(FORNECEDOR_RMS.BUSCAR);
export const useSalvaFornecedor = () => useCurioMutation<void,
SalvaFornecedorRequest>(FORNECEDOR_RMS.SALVAR);
```

4. Schemas

```
// src/pages/Cadastro/Fornecedor/schemas.ts
import { z } from "zod";
export const fornecedorSchema = z.object({  _Nome: z.string().min(1, "Informe o nome").max(120,
"Máximo de 120 caracteres"),
  _CNPJ: z
    .string()
    .min(1, "Informe o CNPJ")
    .regex(/^\d{2}\.\d{3}\.\d{3}\/\d{4}-\d{2}$/, "CNPJ inválido"),  _TipoFornecedor:
z.string().min(1, "Selecione o tipo")
});
export type FornecedorFormData = z.infer<typeof fornecedorSchema>;
export const filtroFornecedorSchema = z.object({
  _Nome: z.string(),
  _Ativo: z.boolean()
});
export type FiltroFornecedorData = z.infer<typeof filtroFornecedorSchema>;
```

5. Estilos

```
// src/pages/Cadastro/Fornecedor/FornecedorPage.styles.tsimport { SxProps, Theme } from
"@mui/material";
export const containerStyle: SxProps<Theme> = {
  display: "flex",
  flexDirection: "column",
  gap: 2,
  p: 3
};
export const headerBarStyle: SxProps<Theme> = {
  display: "flex",
  justifyContent: "flex-end",
  gap: 1
};
export const filtroStyle: SxProps<Theme> = {
  display: "flex",
  gap: 2,
  alignItems: "flex-start",
  flexWrap: "wrap"
};
export const formSectionStyle: SxProps<Theme> = {
  display: "grid",
  gridTemplateColumns: { xs: "1fr", md: "1fr 1fr" },
  gap: 2
};
```

Nomes de constante nomeada, não um objeto `styles` — ver 12.

6. A página

```
```

```
// src/pages/Cadastro/Fornecedor/FornecedorPage.tsximport React, { useEffect, useMemo, useRef, useState } from "react";
import { Box, Button, Divider, Typography } from "@mui/material";import { Save, Search } from "@mui/icons-material";
import { UseCaseManager } from "@curio/client/react";
import { useAuth } from "@context/AuthProvider";import { useUseCaseControls, useValidatedForm } from "@hooks/index";import { DataTable, FormField, SelectInput, type Column, type SelectOption } from "@components/common";
import {
  fornecedorSchema,
  filtroFornecedorSchema,
  type FornecedorFormData,
  type FiltroFornecedorData
} from "./schemas";
import { FORNECEDOR_RMS } from "./service/constants";import { useBuscaFornecedores, useIncluiFornecedor, useSalvaFornecedor } from "./service/hooks";import type { FornecedorXML } from "./service/interfaces";import { containerStyle, filtroStyle, formSectionStyle, headerBarStyle } from "./FornecedorPage.styles";
const colunas: Column<FornecedorXML>[] = [
  { id: "_Nome", label: "Nome", minWidth: 200 },  { id: "_CNPJ", label: "CNPJ", minWidth: 160 },  { id: "_Ativo", label: "Ativo", align: "center", format: (value) => (value ? "Sim" : "Não") }
];
const FornecedorContent: React.FC = () => {
  const { open, status } = useUseCaseControls();  const [fornecedores, setFornecedores] = useState<FornecedorXML[]>([]);
  const { data: dadosIniciais, mutate: incluirFornecedor, isPending: isIniciando } = useIncluiFornecedor();  const { data: resultadoBusca, mutate: buscarFornecedores, isPending: isBuscando } = useBuscaFornecedores();  const { mutateAsync: salvarAsync, isPending: isSalvando } = useSalvaFornecedor();
  // refs impedem reabrir/reinicializar a cada render  const hasAttemptedOpenRef = useRef(false);
  const hasInitializedRef = useRef(false);
  // maquina de estados: idle -> abre; open -> inicializa
  useEffect(() => {
    if (status === "idle" && !hasAttemptedOpenRef.current) {      hasAttemptedOpenRef.current = true;
```

```
      open();
    } else if (status === "open" && !hasInitializedRef.current) {      hasInitializedRef.current
= true;
      incluirFornecedor();
    }
  }, [status, open, incluirFornecedor]);
  useEffect(() => {
    if (resultadoBusca) setFornecedores(resultadoBusca.Response ?? []);
  }, [resultadoBusca]);
  const tiposFornecedor: SelectOption[] = useMemo(    () => (dadosIniciais?.TiposFornecedor ??
[]).map((tipo) => ({ value: tipo._OID, label: tipo._Titulo })),
    [dadosIniciais]
  );
  const form = useValidatedForm({
    schema: fornecedorSchema,
    defaultValues: { _Nome: "", _CNPJ: "", _TipoFornecedor: "" }
  });
  const filtroForm = useValidatedForm({
    schema: filtroFornecedorSchema,
    defaultValues: { _Nome: "", _Ativo: true }
  });
  const handleBuscar = (data: FiltroFornecedorData) => {    buscarFornecedores({ OBJECTID: {
_Nome: data._Nome, _Ativo: data._Ativo } });
  };
  const handleSalvar = async (data: FornecedorFormData) => {
    await salvarAsync({
      Fornecedor: {
        _OID: String(dadosIniciais?.Fornecedor._OID ?? ""),
        _Nome: data._Nome,
        _CNPJ: data._CNPJ,
        TipoFornecedor: { _OID: data._TipoFornecedor }
      },
      msgSucesso: "Fornecedor salvo com sucesso!"
    });
    form.reset();
    incluirFornecedor(); // novo objeto pro proximo cadastro
```

```
filtroForm.handleSubmit(handleBuscar)();
  };
  const isAnyLoading = isIniciando || isBuscando || isSalvando;
  return (
    <Box sx={containerStyle}>
      <Box sx={headerBarStyle}>
        <Button
          variant="contained"
          startIcon={<Save />}
          onClick={form.handleSubmit(handleSalvar)}
          disabled={isAnyLoading}>
          Salvar
        </Button>
      </Box>
      <Typography variant="h3">Cadastro de fornecedor</Typography>
      <Box sx={formSectionStyle}>        <FormField name="_Nome" control={form.control}
label="Nome" size="small" fullWidth />        <FormField name="_CNPJ" control={form.control}
label="CNPJ" size="small" fullWidth />
        <SelectInput
          name="_TipoFornecedor"
          control={form.control}
          label="Tipo"
          options={tiposFornecedor}
          size="small"
          fullWidth
        />
      </Box>
      <Divider />
      <Typography variant="h4">Fornecedores cadastrados</Typography>
      <Box sx={filtroStyle}>        <FormField name="_Nome" control={filtroForm.control}
label="Filtrar por nome" size="small" />
        <Button
          variant="outlined"
          startIcon={<Search />}
onClick={filtroForm.handleSubmit(handleBuscar)}
          disabled={isBuscando}>
```

```
          Buscar
        </Button>
      </Box>
      <DataTable
        columns={colunas}
        data={fornecedores}
        loading={isBuscando}
        emptyMessage="Nenhum fornecedor encontrado."
        stickyHeader
        maxHeight={400}
      />
    </Box>
  );
};
const FornecedorPage: React.FC = () => {
  const { session } = useAuth();
  return (    <UseCaseManager session={session} useCaseId={FORNECEDOR_RMS.USE_CASE}
autoClose={false} openOnMount={false}>
      <FornecedorContent />
    </UseCaseManager>
  );
};
export default FornecedorPage;
```

7. Barrel da página

```
// src/pages/Cadastro/Fornecedor/index.ts
export { default } from "./FornecedorPage";
```

Obrigatório. Sem ele, o `lazy(() => import("@pages/Cadastro/Fornecedor"))` falha em runtime **sem erro de compilação** — o TypeScript não verifica o alvo do import dinâmico.

8. Caminho

```
// src/routes/paths.ts
export const PATHS = {
  LOGIN: "login",
  DASHBOARD: "dashboard",
  CADASTRO: {
    FORNECEDOR: "cadastro/fornecedor" // novo
  }
} as const;
```

9. Rota

```
// src/routes/routes.ts
export const routes: AppRoute[] = [
  // ...
  {
    path: PATHS.CADASTRO.FORNECEDOR,    element: lazy(() =>
import("@pages/Cadastro/Fornecedor")),
    guard: "protected"
  }
];
```

10. Menu

```
// src/routes/menuTree.ts
export const menuTree: MenuNode[] = [  { label: "Dashboard", path: PATHS.DASHBOARD, mode:
"route" },
```

```
  {
    label: "Cadastro",    children: [{ label: "Fornecedor", path: PATHS.CADASTRO.FORNECEDOR }]
// modo default: "tab"
  }
];
```

Árvore final

```
src/pages/Cadastro/Fornecedor/
├── FornecedorPage.tsx
├── FornecedorPage.styles.ts
├── schemas.ts
├── index.ts
└── service/
    ├── constants.ts
    ├── hooks.ts
    └── interfaces.ts
```

Mais três arquivos tocados: `routes/paths.ts`, `routes/routes.ts`, `routes/menuTree.ts`.

Verificação

Sem suíte de testes, esta é a evidência. Rode tudo.

Estático

```
npm run lint && npx tsc --noEmit
```

Funcional — `npm run dev`, então:

- [ ] O item "Fornecedor" aparece sob "Cadastro" no menu lateral
- [ ] Clicar abre a aba (não 404, não "Rota não encontrada")
- [ ] O select "Tipo" vem preenchido — prova que o caso de uso abriu e o request de abertura respondeu
- [ ] Salvar com campos vazios mostra as mensagens de validação e **não** envia request

- [ ] Salvar preenchido mostra a notificação verde
- [ ] Buscar preenche a tabela
- [ ] Buscar sem resultado mostra "Nenhum fornecedor encontrado."
- [ ] Nenhum erro no console
- [ ] Aba Network: os requests saem para a URL do `config.json` do ambiente ativo
- [ ] F5 na tela mantém a sessão (não cai no login)
- [ ] Fechar a aba encerra o caso de uso (se a tela usar `useTabCloseCallback` — ver 09)

Falha proposital

- [ ] Com o backend inacessível, a busca mostra notificação vermelha e a aplicação não trava

Só depois de tudo isso registre a feature como `passing` no `harness/state/feature_list.json` — ver 19.

Erros comuns

Sintoma	Causa
Tela em branco, console: hook fora de contexto	`useCurioMutation` no mesmo componente que renderiza o `UseCaseManager`
Select "Tipo" vazio	Caso de uso não abriu; verifique `status` e o id
Import dinâmico falha em runtime	Faltou `index.ts` na pasta da página
Item no menu leva a 404 ou "Rota não encontrada"	Registrado em `menuTree` mas não em `routes.ts`
Rota duplica barra (`//cadastro`)	`PATHS` com barra inicial
Requests repetidos em loop	Faltou o `useRef` de guarda no `useEffect`
Backend recebe id/RM diferente do esperado	Constante duplicada fora de `service/constants.ts`, divergindo do `_RMS`

Revision #2
Created Thu, Aug 20, 2026 5:08 PM by Geraldo Barbosa
Updated Tue, Aug 25, 2026 4:55 PM by Geraldo Barbosa