Instituto do Emprego e Formação Profissional, IP
Centro de Emprego e Formação Profissional do Médio Tejo
Serviço de Formação Profissional de Tomar

PROJETO LARAVEL

FCT – FORMAÇÃO EM CONTEXTO DE TRABALHO

FÁBIO RIBEIRO E RICARDO BRITO

Nº 8086915 – Nº 5677370

CET REDES - 2024/2025

1 DE JULHO DE 2025

PESSOAS 2030 PORTUGAL 2030 Cofinanciado pela União Europeia

Índice

Índice de Imagens

Introdução

A ideia para este projeto surgiu por parte de entidade empregadora. Uma certa lista de contactos que estava organizada em um ficheiro de Excel, embora fosse usável era muito mais difícil de organizar em comparação com um programa deste estilo.

Então nasceu a ideia e a sugestão de fazer um programa/aplicação para podermos armazenar e gerir uma base de dados com mais facilidade e eficiência. Considerando que eramos de longe entendidos da matéria, houve bastantes horas de pesquisa e aprendizagem.

Após o aprender veio o fazer, finalmente chegamos a um ponto onde nos sentíamos confortáveis para começar a construção deste projeto. Uma das várias decisões que tivemos de tomar foi acerca de quais seriam os programas que iriamos utilizar para o desenvolvimento. Depois de várias respostas e questões chegamos a uma conclusão final.

Programas Utilizados

PhpStorm – Decidimos que este seria o melhor IDE para este tipo de programação;

Php – Para utilizarmos a linguagem de programação php;

Node.js – Para a linguagem de programação JavaScript;

Git – Para fazermos a partilha do projeto no Github, para podermos programar independentemente um do outro;

SQLite – Para usar uma base de dados local;

Composer – Para a instalação (via terminal) dos vários recursos usados:

- **Laravel** – Framework;

- **Tailwind CSS** e **DaisyUi** – Para os estilos e css. O Tailwind já vem com o Laravel e o DaisyUi é um plugin;

- **Spatie** – Um pacote para usar permissões no Laravel;

Estrutura do Programa

Temos então aqui a estrutura de pastas do nosso programa. Mas o que é que fazem estas pastas e o que é que contêm lá dentro? Vamos então à descoberta.

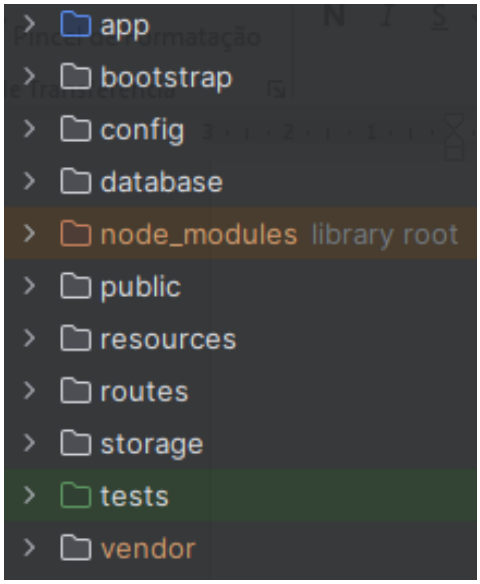


Figura 1 - Estrutura do Programa

App

Este nosso diretório contém os ficheiros que vão controlar as funcionalidades e lógica necessárias para o funcionamento da aplicação, o **backend**.

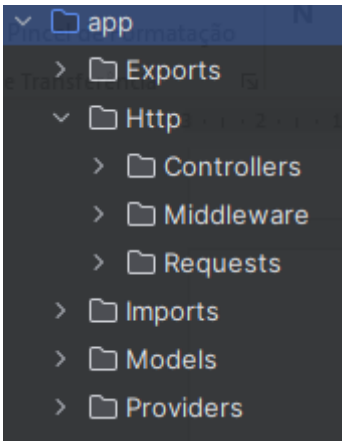


Figura 2 – App

Começando pelas mais importantes, dentro da subpasta **HTTP** que contém o processamento de pedidos HTTP:

- **Controllers** – Os Controladores são responsáveis por receber o pedido das routes e devolver uma reposta. **Controlam a lógica da aplicação**. Ex: O ContactController vai controlar a BD[1] dos contactos, neste caso vai ditar quais são os campos que podem ser preenchidos, que tipo de campos são, como vamos fazer a atualização dos mesmos, etc.
- **Middleware** – Contém filtros aplicados aos pedidos. Ex: Um dos ficheiros dentro desta pasta vai detetar a linguagem do browser que está a ser usado.
- **Requests** – Contém ficheiros personalizados que servem para **validar e autorizar dados** enviados por formulários ou APIs. Validam os dados antes de chegarem aos controladores e muitas vezes verificam se o utilizador tem a permissão ou permissões necessárias.

[1] Base de Dados

Também extremamente importantes:

- **Models** – Contém os modelos **Eloquent**, que representam as tabelas da base de dados. Cada um define relações, atributos e regras para uma entidade (Ex: contacto, user).
- **Providers** – Composto por apenas 1 ficheiro que existe por defeito, **AppServiceProvider**. Ele é carregado automaticamente sempre que a aplicação é executada. Serve para registrar **configurações globais e ligações** no container de serviços do Laravel.

Finalmente, no caso da nossa aplicação, também existem estas duas subpastas acessórias para a importação/exportação da tabela de contactos:

- **Exports** – Ficheiro para fazer a exportação da nossa base de dados. Vamos buscar os dados da nossa BD, neste caso a “contactos”, para passar para um ficheiro xlsx (Excel) com o mesmo nome.
- **Imports** – O oposto do Exports. Importa uma base de dados de acordo com os campos que inserirmos neste ficheiro, isto irá sempre adicionar à BD, nunca irá substituir dados que já lá estejam. Neste caso só temos 1 ficheiro em cada uma das pastas, mas poderíamos adicionar mais, caso fosse necessário fazer o mesmo para BDs adicionais.

Database

A pasta database contém tudo o que está relacionado com a estrutura e o conteúdo da base de dados da aplicação. Também contém a própria BD (database.sqlite), ter em atenção que esta BD é sempre local.

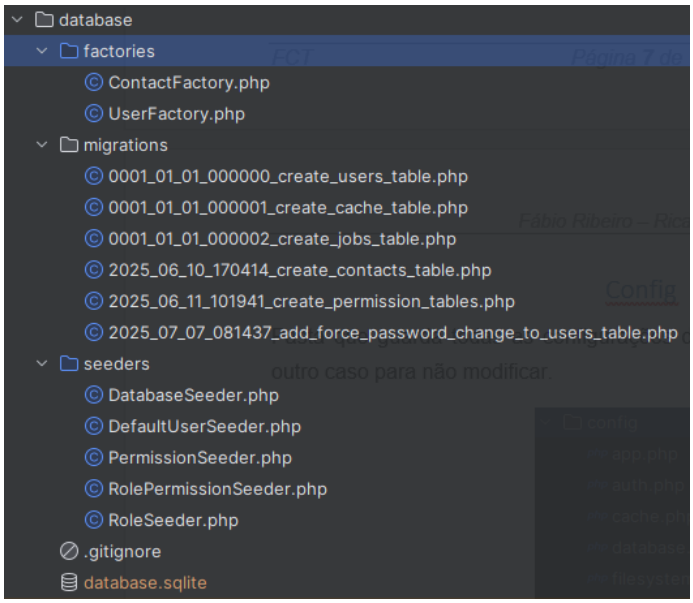


Figura 3 - Database

Ela está dividida nas seguintes subpastas:

- **Factories** - Contém os ficheiros usados para **gerar entradas falsas** utilizadas para testes ou algo que possa requerer rapidamente popular uma base de dados. São usados também nos seeders abaixo.
- **Migrations** - Define a **estrutura das tabelas** na base de dados, cada ficheiro contém o código que vai criar uma tabela, permitindo criar, alterar e apagar colunas e em alguns casos adicionar colunas a uma tabela já feita.
- **Seeders** - Estes são os ficheiros responsáveis por inserir **dados pré-criados** nas tabelas criadas pelas migrações, como por exemplo criar os utilizadores, cargos e permissões necessárias para o funcionamento básico do programa. Podem usar as factories acima para adquirir os dados.

Resources

A seguinte pasta contém o **front-end** da aplicação, é aquilo que é apresentado ao utilizador.

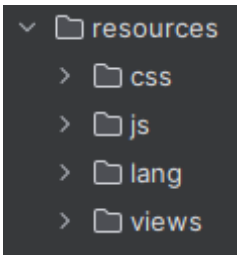


Figura 4 – Resources

As suas subpastas são:

- **Css** - Contém o ficheiro **app.css**, o qual controla o estilo visual geral das páginas.
- **Js** - Onde estão os ficheiros **JavaScript** personalizados da aplicação.
- **Lang** - As traduções usadas no programa, neste caso são dois ficheiros json, "en" para inglês e "pt" para português.
- **Views** - Contém os **ficheiros de visualização** (blade.php) usados para gerar o HTML da aplicação. Contém os layouts, páginas e componentes. Geralmente está dividida em várias subpastas, de acordo com as páginas em que estão a ser usadas.

Routes

A pasta das routes contem os ficheiros onde se definem todas as **rotas** da aplicação, as ligações entre URLs e o que a aplicação deve fazer quando essas URLs são acedidas.

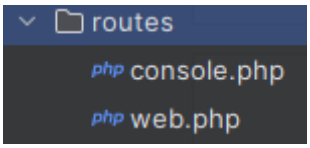


Figura 5 - Routes

Contém apenas dois ficheiros:

- **Console.php** - Define **comandos personalizados de Artisan**, ou seja, é utilizado apenas quando se escreve comandos no terminal, "php artisan".
- **Web.php** - Contém as **rotas da web**, ou seja, as que devolvem páginas HTML (ex.: /contactos para a página dos contactos). É aqui que normalmente é usado o middleware para adicionar outras funcionalidades a estas rotas e garantir a segurança das mesmas, garantir que utilizadores sem certas permissões possam aceder a determinadas páginas. Aqui está um exemplo de um utilizador normal a tentar aceder a uma página em que são precisos privilégios de admin:

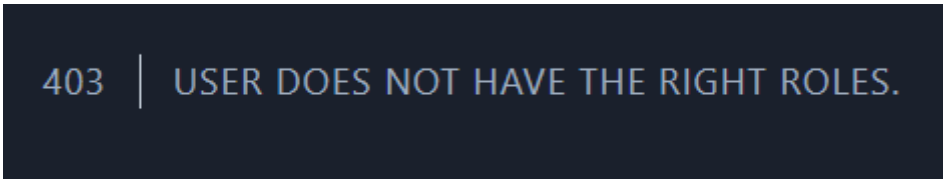


Figura 6 - Mensagem de Aviso 1

Outras Pastas

Também importantes, mas requerem pouca ou nenhuma modificação:

Bootstrap - Inicia a aplicação. Contém o ficheiro app.php, que carrega o núcleo do Laravel, e a subpasta cache, que guarda ficheiros de arranque das rotas.

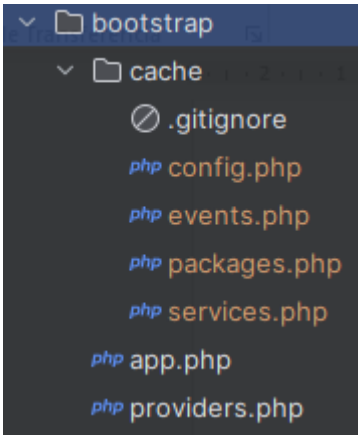


Figura 7 – Bootstrap

Config - Basicamente, os ficheiros de configuração da aplicação.

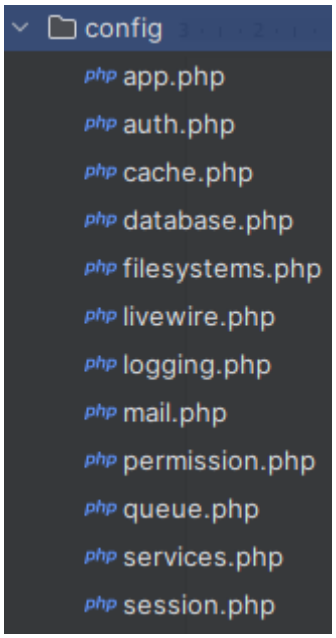


Figura 8 - Config

Node_modules - Gerada pelo Node.js, contém os pacotes do **front-end** (vite, tailwind, etc.). **Não mexer!**

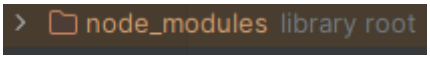


Figura 9 - Node_Modules

Public - Ponto de entrada web da aplicação, contém vários ficheiros públicos (para o programa) como os ícones da app e os ficheiros CSS/JS[2] compilados.

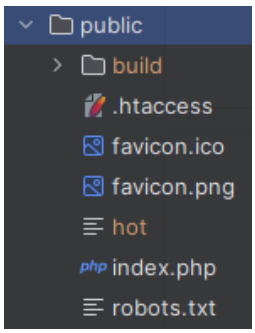


Figura 10 - Public

Storage - **Guarda ficheiros** gerados pela aplicação (logs, cache, uploads, sessões). Contém as subpastas app, onde estão os uploads e outros ficheiros.

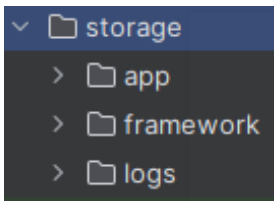


Figura 11 – Storage

Tests - Contém os testes automatizados da aplicação.

Figura 12 - Tests

Vendor - Gerada pelo composer, contém todas as dependências do PHP. **Absolutamente não mexer!**

Figura 13 – Vendor

[2] JavaScript

Outros Ficheiros

Na base do projeto estão ficheiros essenciais ao funcionamento do Laravel.

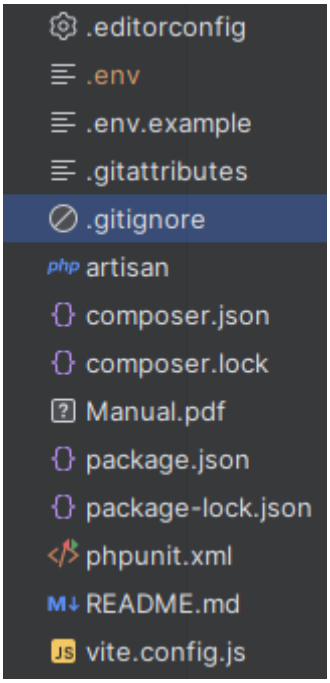


Figura 14 – Outros Ficheiros

Mas o que faz estes ficheiros tão importantes? É exatamente isso que vamos ver, pela mesma ordem em que estão no programa:

- **.Editorconfig** - Este ficheiro define as **regras de formatação do código** (como espaços, indentação, fim de linha). Serve para garantir que todos os programadores usam o mesmo estilo, independentemente do editor usado.
- **.Env** - Contém as configurações específicas do ambiente, como por exemplo o nome da base de dados, as chaves da aplicação, e as credenciais. **Não deve ser partilhado** por razões de segurança.
- **.Env.example** - É um modelo do **.env**. Serve como referência para outros programadores saberem quais as variáveis necessárias. Normalmente, é copiado para formar o **.env**.

- **Artisan** - Ficheiro php usado na **linha de comandos**. É o comando principal do Laravel para executar tarefas.
- **Composer.json** - Contém as **dependências** php do projeto e outras definições como pacotes obrigatórios, scripts automáticos e o nome e descrição do projeto.
- **Composer.lock** - Ficheiro gerado automaticamente após instalar as dependências com o Composer. Garante que todos usam exatamente as mesmas versões dos pacotes. ***Não deve ser editado manualmente*.**
- **Manual.pdf** - Este é o manual que estás a ler, em formato PDF, incluído dentro do programa.
- **Package.json** - Ficheiro que define as **dependências de front-end** da aplicação (JavaScript). Usado com o Node.js (via **npm**), normalmente para Vite, Tailwind CSS, etc.
- **Package-lock.json** - Gerado automaticamente após instalar pacotes via **npm**. Guarda todas as versões exatas dos pacotes instalados. Tal como o **composer.lock**, ***não deve ser alterado manualmente*.**
- **Phpunit.xml** - Ficheiro de configuração para os **testes automáticos** com PHPUnit. Define o ambiente de testes.
- **Vite.config.js** - Configuração do **Vite**, a ferramenta que compila e gere os recursos da aplicação (CSS, JS, etc.). Este ficheiro liga o Laravel ao **front-end**.

Utilização Geral

Já vimos as pastas, o que está dentro delas e como funciona a estrutura do nosso programa, mas agora vamos ver como é que tudo funciona na prática.

Vamos então por as mãos na massa.

Login

Começando no início, para aceder à aplicação deverá ser feito o **login de forma obrigatória,** tentar aceder a qualquer página sem o login feito irá dar uma mensagem de erro.

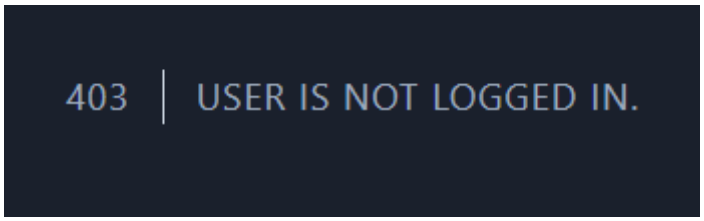


Figura 15 - Mensagem de Aviso 2

Ao tentar aceder pela primeira vez, considerando que as tabelas dos utilizadores contem apenas os que foram criados usando o seeder, temos que fazer o login com uma dessas contas.

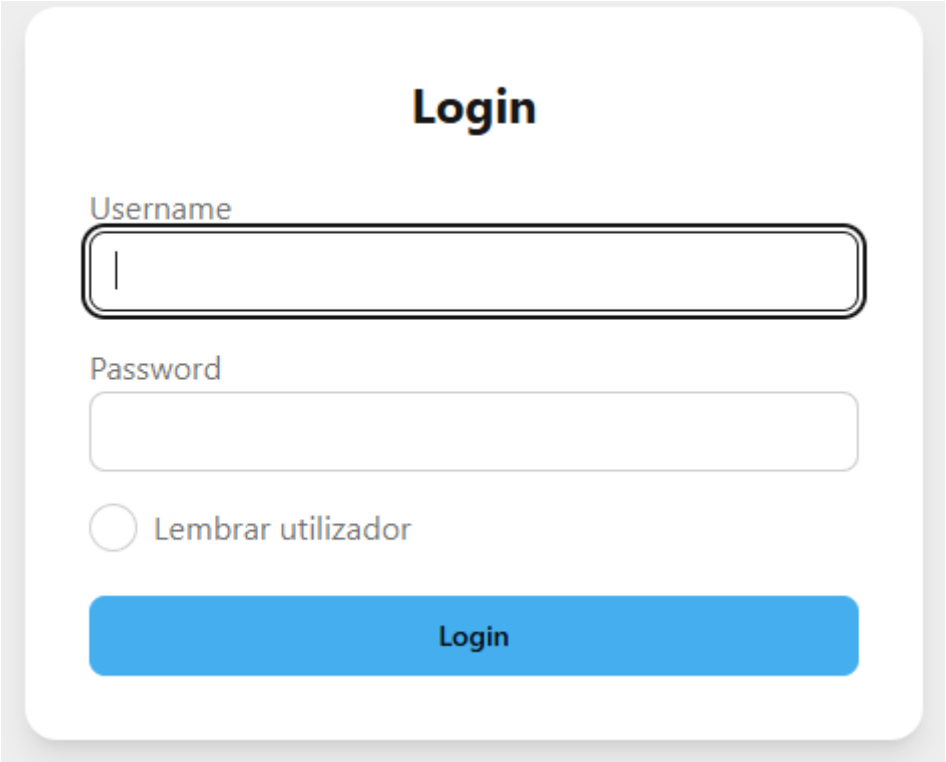


Figura 16 – Login

A

Mudança de password

No caso de o utilizador estar a usar uma conta criada pelo administrador, por defeito, será enviado automaticamente para uma página em que terá de mudar de password e introduzir uma nova no seu primeiro login. **O sistema está configurado para não aceitar a mesma password**.

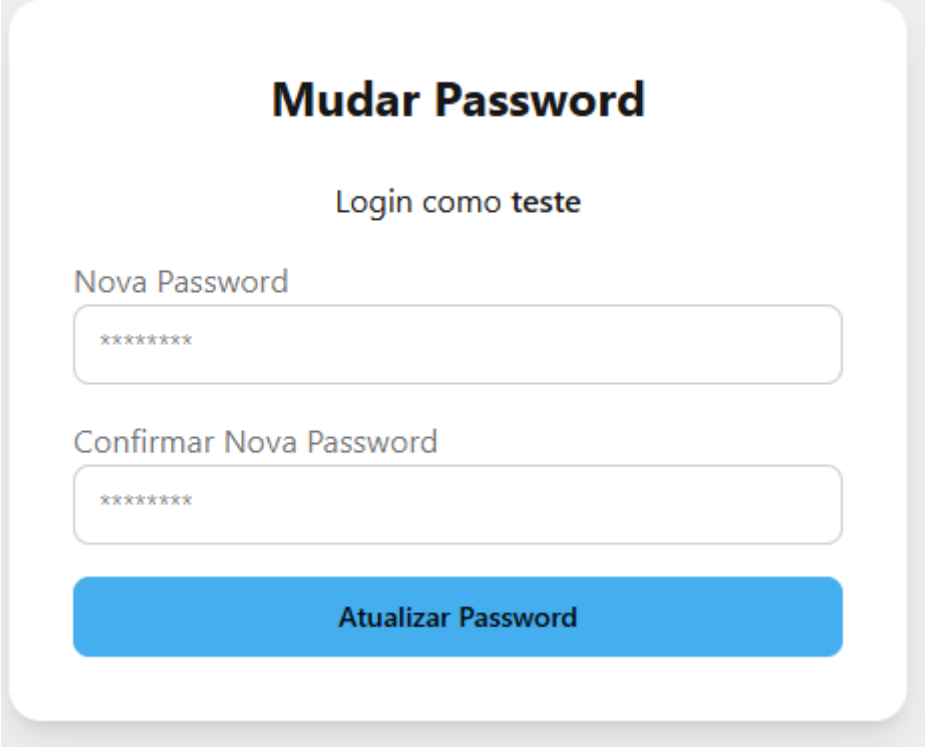


Figura 17 - Mudar Password

Home/Dashboard

Logo a seguir ao login, somos enviados automaticamente para a página "Home/Dashboard", esta página, no caso de admins[3] terá duas tabs[4], a inicial que será a de **administrador**, com os utilizadores mais recentes, e a dos **contactos**, com os contactos mais recentes.

No caso de um utilizador normal, este apenas verá a tab de contactos.

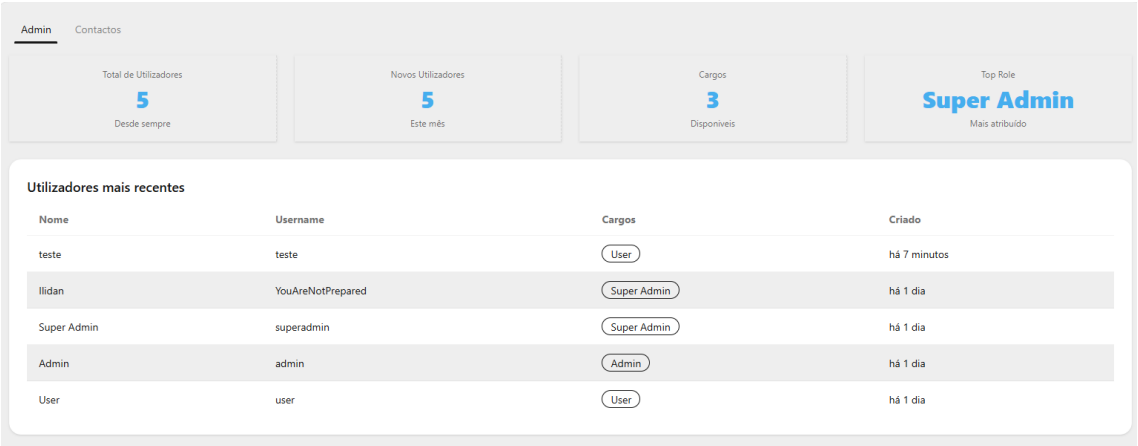


Figura 18 - Dashboard Admin

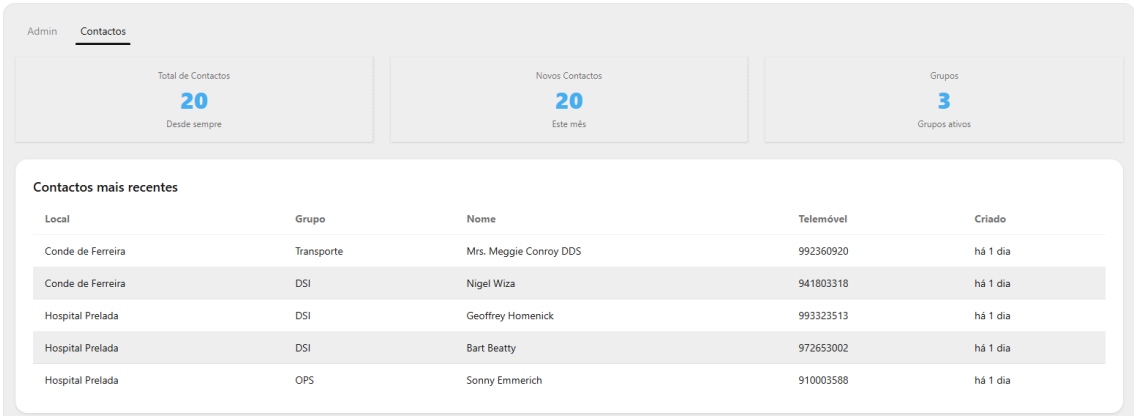


Figura 19 - Dashboard Contactos

[3] Administradores
[4] Separadores

Sidebar[5]

Em todas as páginas exceto as dedicadas a login e mudança forçada de password estará à esquerda um **menu vertical** contendo a navegação da aplicação. Este menu, em ecrãs pequenos fica escondido e volta a aparecer quando é passado o rato perto da beira esquerda da página.

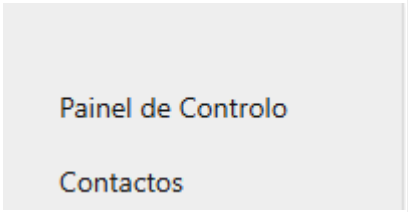


Figura 20 - Sidebar 1

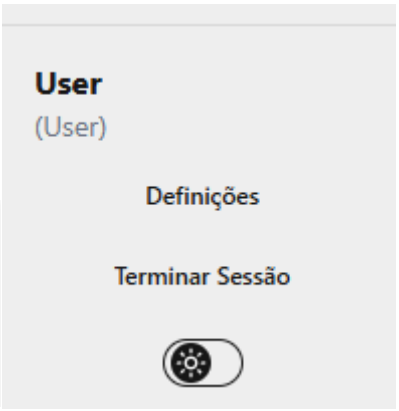


Figura 21 - Sidebar 2

Para um utilizador normal, este menu terá dois elementos na parte de cima, a página **home/dashboard** para a qual é direcionado no início da sessão, e a página **Contactos**. Em baixo, está o nome do utilizador, assim como o seu cargo (user, admin), e também 2 “links”. Um link para as definições do utilizador e outro para fazer o logout da aplicação.

[5] Barra Lateral

Definições

Esta página permite-nos mudar o nosso nome (mas não o username) e a password sem intervenção do administrador.

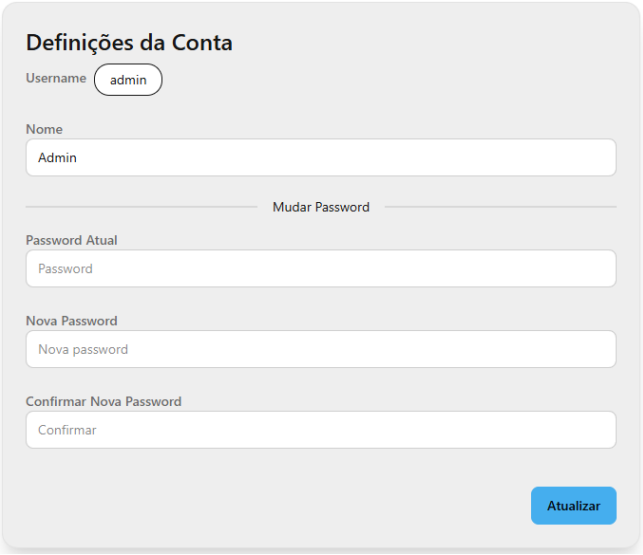


Figura 22 - Definições

No caso de o utilizador ser um admin, na sidebar, a seguir aos **Contactos**, terá também um componente dedicado a administradores, com as páginas **Gerir Utilizadores**, **Gerir Cargos** e **Gerir Permissões**.

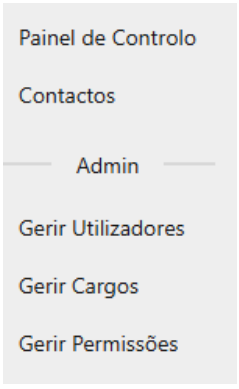


Figura 23 - Admin Sidebar

Contactos

Esta página é a central deste programa. É aqui que vamos mostrar os campos principais da nossa BD (Local, Grupo, Nome e Telemóvel) , mas de uma forma muito mais estética.

A mesma é composta por dois elementos chave, o cabeçalho da tabela, e a tabela em si.

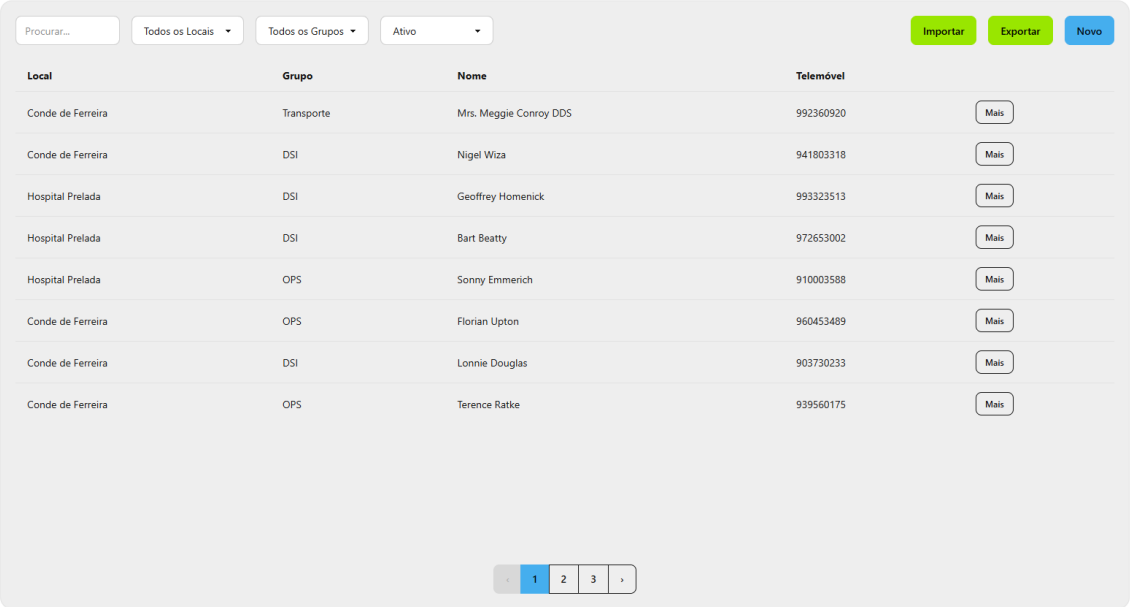


Figura 24 - Contactos

O **Cabeçalho** contém a procura e filtros da tabela, assim como os botões de importar e exportar a tabela, tudo feito através de ficheiros de excel. Ao seu lado temos também outro botão para criar um novo contacto através de um modal[6].

[6] Janela do estilo pop-up

O modal que aparece quando se carrega no botão **Novo** contém os quatro campos principais. É necessário preencher todos antes de poder ser criado este novo contacto.

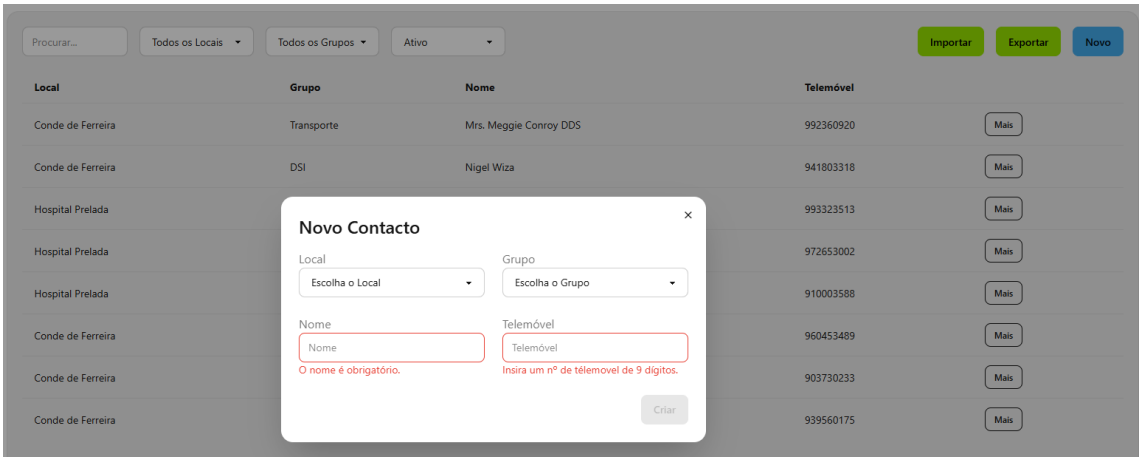


Figura 25 - Novo Contacto

Os primeiros dois podem ser escolhidos utilizando as localizações e grupos que nós definimos no programa e os dois últimos podem ser ambos pesquisados “manualmente” com a barra de procura. Procura e filtros podem ser combinados em qualquer ordem.

Apenas para administradores, a tabela tem também um filtro que permite ver os contactos que foram "apagados" (**soft delete**[7]).

Contém também um botão que vai levar a uma página que vai mostrar todos os dados do contacto correspondente.

[7] Apagados na tabela, mas não na base de dados. Esconder em vez de apagar. Apenas visível para admins, utilizando o filtro.

Esta página está dividia em duas tabs, **Contacto**, para informações pessoais e o **Equipamento**, para as informações mais técnicas.

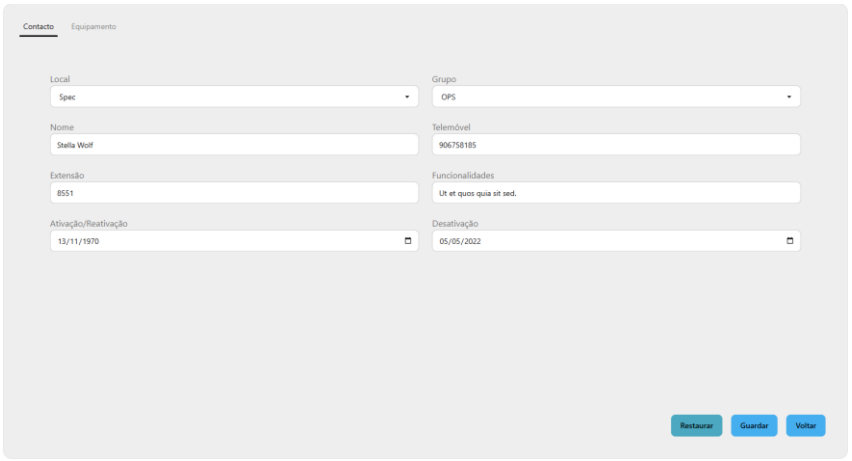


Figura 26 - Contacto do Registo (desativado)

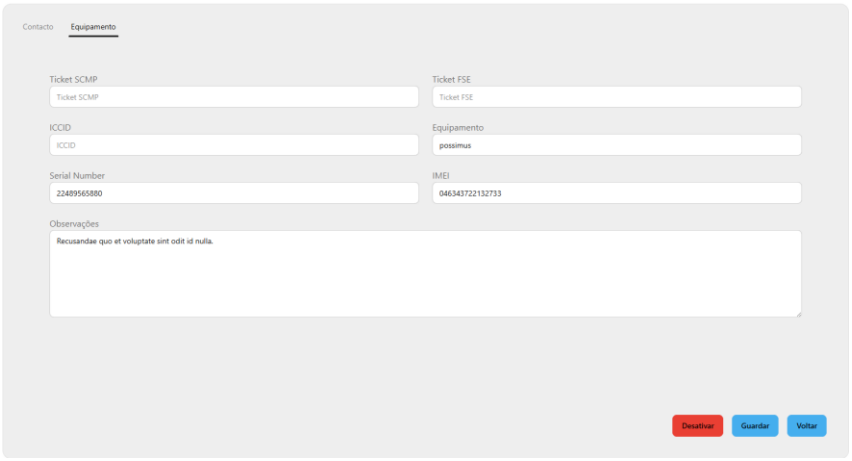


Figura 27 - Equipamento do Registo (ativado)

Esta página tem também no canto inferior direito 4 botões:

- **Restaurar** - Faz o restauro do contacto. O utilizador tem que ser admin para poder aceder a este tipo de registo.
- **Desativar** - Este botão faz um **soft delete** do contacto.
- **Apenas 1 destes 2 aparece, dependendo do estado do registo.**
- **Guardar** - Este botão guarda alterações feitas ao contacto;
- **Voltar** - Volta para trás, ou seja, para a tabela dos Contactos exatamente como estava anteriormente, na mesma página, com os mesmos filtros.

Gerir Utilizadores

Uma das páginas para uso exclusivo de administradores, esta página permite criar, editar, desativar e restaurar utilizadores e os cargos de cada um.

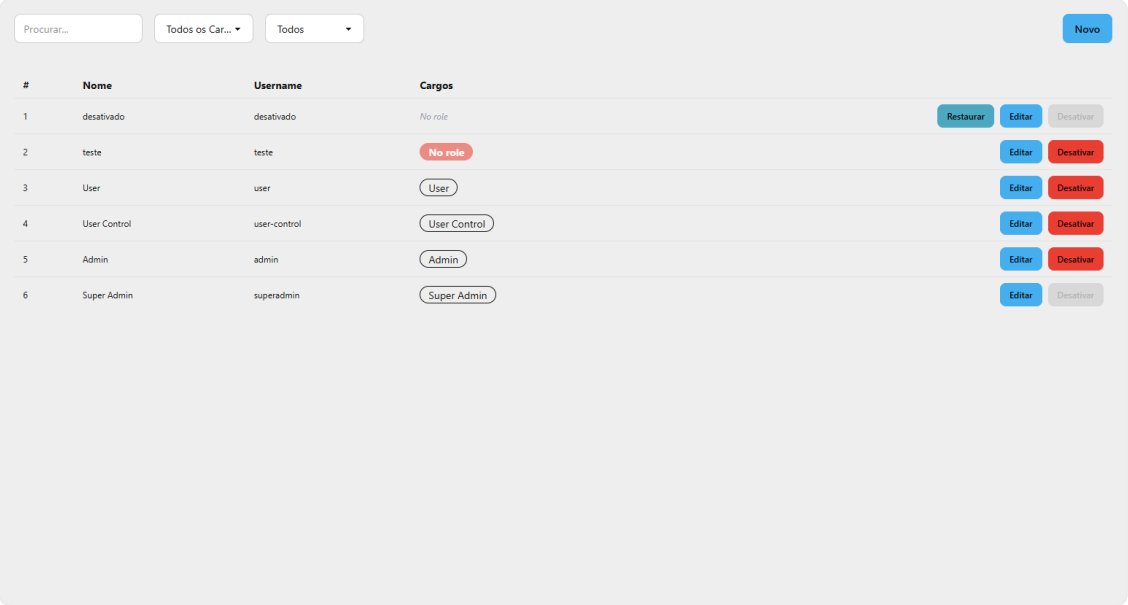


Figura 28 - Utilizadores

O **Cabeçalho** é constituído por um campo de pesquisa que permite procurar os utilizadores tanto pelo Nome como pelo Username, tem também incluído 2 filtros, um que permite discriminar os utilizadores por Cargo, e outro para verificar os utilizadores desativados. Tem mais um botão para criar um novo utilizador.

A Tabela contém o **Id**, **Nome**, **Username** e **Cargo** dos utilizadores, à direta de tudo tem também os botões usados para editar, desativar e restaurar cada utilizador.

Quando um utilizador é desativado ele perde o seu cargo e o botão de desativar fica desligado. Há esquerda do botão de editar irá aparecer um botão para restaurar o utilizador. Nesta restauração, o cargo antigo não volta automaticamente, ou seja, temos de voltar a adicionar. Por este motivo também tem uma pequena animação no campo do cargo após ser feito o restauro, para chamar a atenção ao admin.

Tal como nos contactos, tanto a criação como a edição de utilizador são feitas através de modais, no modal aparecem quatro campos, **Nome**, **Username**, **Password** e **Cargos**. Temos também um campo para reintroduzir a password e confirmar que está correta.

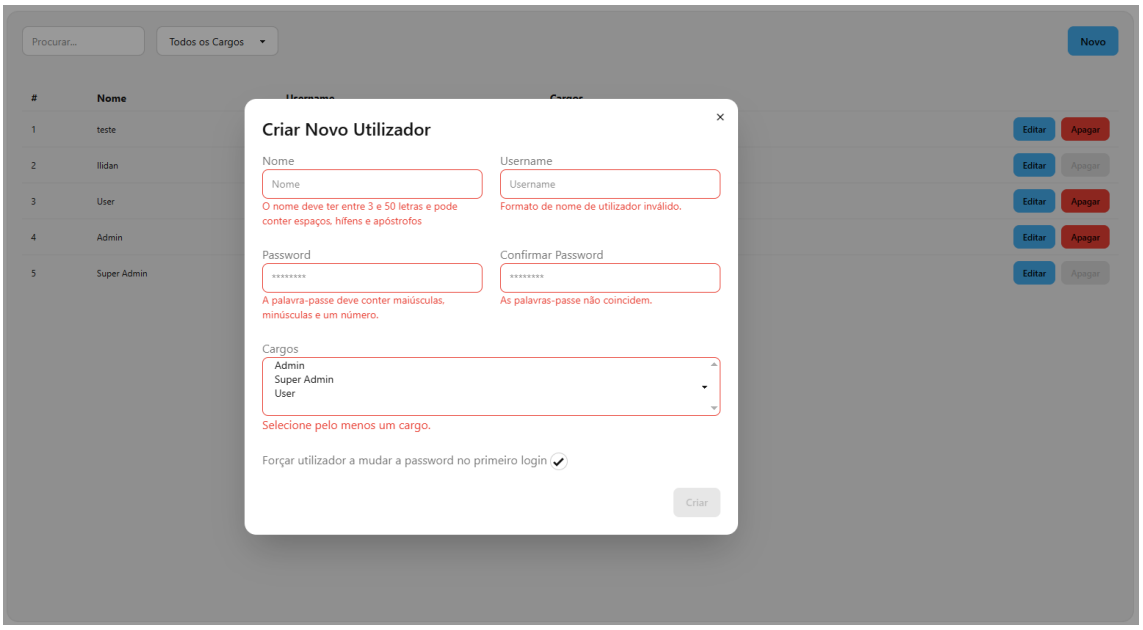


Figura 29 - Criar Utilizador

No caso de criar um novo utilizador, todos os campos são obrigatórios. No caso de editar um utilizador, não é permitido remover os campos e deixar em branco, mas caso não seja necessário mudar a password, é possível mudar os restantes campos sem a introduzir.

Os modais têm também uma checkbox[8] para obrigar o utilizador a mudar de password no próximo login. Esta opção está ativada por defeito na criação de um novo utilizador, mas é também possível reativá-la para um utilizador existente, de modo a obrigar a introdução de uma nova password no próximo login.

[8] Caixa de Seleção

Gerir Cargos

Esta página permite criar e apagar cargos bem como gerir as permissões de cada um.

Figura 30 - Cargos

O **Cabeçalho**, visto a quantidade de Cargos ser por sua natureza pequena, contém apenas um botão para criar um novo cargo. Não tem qualquer forma de procura.

Este botão abre um modal onde se pode indicar o nome do cargo que queremos criar, assim como escolher as permissões que o mesmo irá ter. É apenas obrigatório e pelo menos uma permissão.

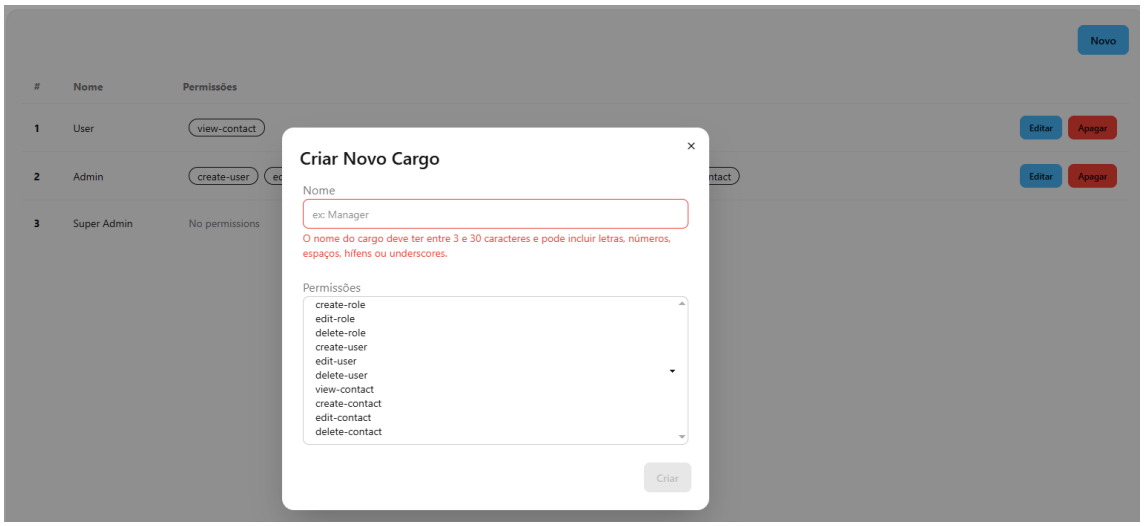


Figura 31 - Criar Cargo

A **Tabela** contém o **Id**, **Nome** e **Permissões**, assim como botões para editar e apagar cada cargo.

O botão editar também abre um modal com as mesmas características do modal anterior.

Gerir Permissões

Esta página permite criar e apagar permissões.

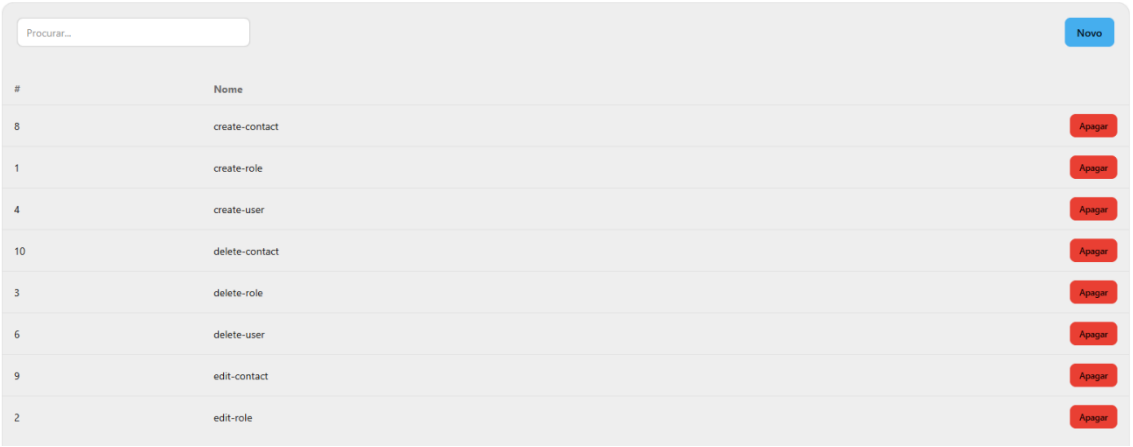


Figura 32 - Permissões

O **Cabeçalho** contém um campo de busca e um botão para criar uma nova permissão.

A **Tabela** contém o **Id** e **Nome**, assim como um botão para apagar cada permissão.

Novas permissões são criadas através de um modal cujo único campo é o nome.

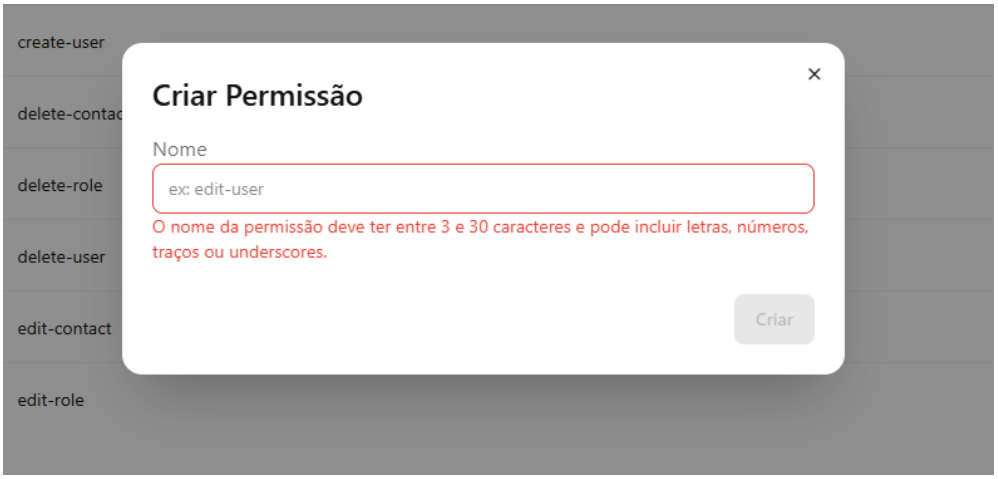


Figura 33 - Criar Permissão

Atenção: As novas permissões criadas através desta forma terão de ser programadas no código da aplicação para terem efeito. Esta página serve apenas para não ser necessário interagir diretamente com a base de dados.

SQLite Trek

No computador onde se encontra a base de dados podemos instalar a aplicação SQLite Trek diretamente da Microsoft Store. Com esta aplicação temos a opção de aceder diretamente à BD.

Aqui dentro estamos com poder total e podemos editar a base de dados sem restrições nenhumas. Mas a razão principal para a utilização desta ferramenta, neste caso, será para dar a possibilidade de apagar um registo permanentemente.

Visto que a nossa BD se encontra em SQLite, para podermos manusear os registos teremos de usar comandos SQL.

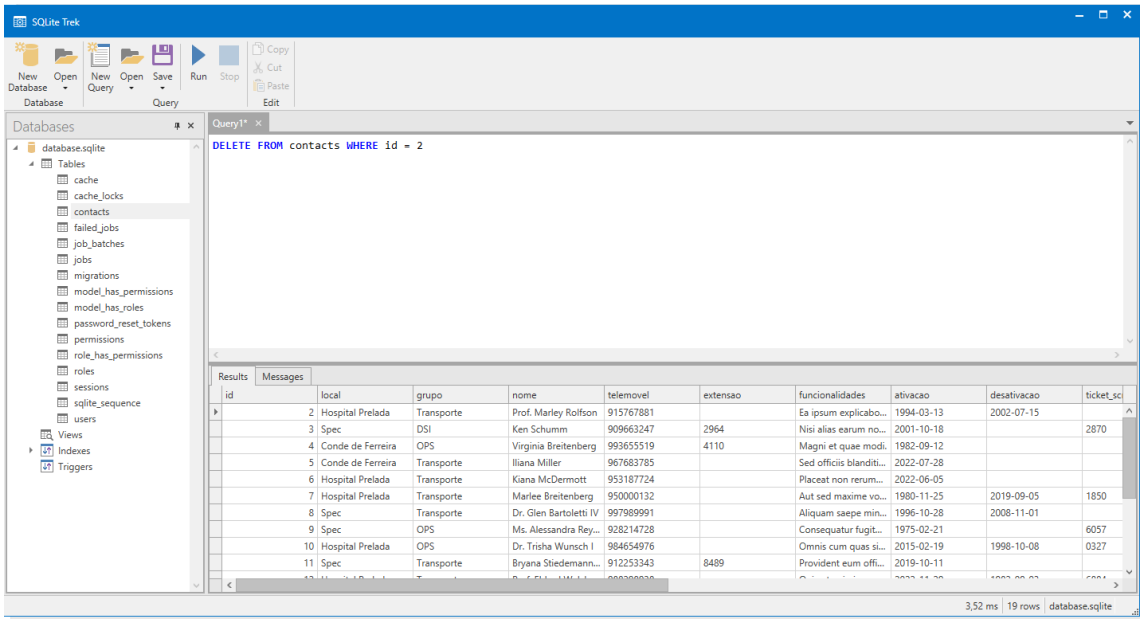


Figura 34 - SQLite Trek

Neste exemplo, podemos visualizar a tabela “contacts” e fazer uma query para apagar o contacto com o id 2. Existem várias maneiras de chegar ao mesmo resultado, mas esta (usando o id) é a mais segura porque o id de cada contacto é único.

Conclusão

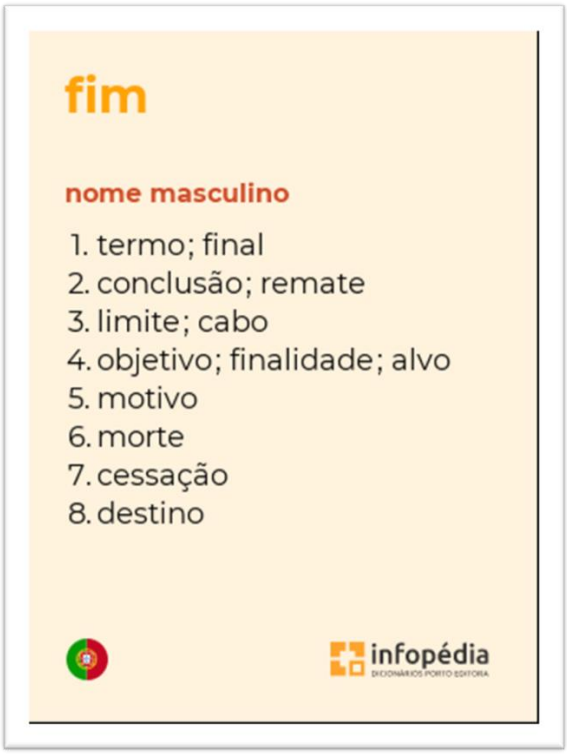
E por aqui ficamos.

Esperamos nós que após a leitura completa deste manual, qualquer mero mortal estará muito melhor familiarizado com a estrutura, não só deste programa, mas também com a estrutura do próprio Laravel.

Agradecemos toda a vossa atenção e interesse neste projeto.

Obrigado pela leitura;

Fábio Ribeiro, Ricardo Brito



Documentos

Ainda não te sentes à vontade ou queres explorar mais sobre o assunto?

Não temas, aqui estão os links que podes visitar para aceder à documentação e ás páginas oficiais dos programas que foram utilizados para a construção e desenvolvimento deste projeto:

https://www.jetbrains.com/phpstorm/

https://www.php.net/docs.php

https://nodejs.org/docs/latest/api/

https://git-scm.com/doc

https://www.sqlite.org/docs.html

https://getcomposer.org/doc/

https://laravel.com/docs/12.x

https://tailwindcss.com/docs/installation/using-vite

https://daisyui.com/docs/intro/

https://spatie.be/docs/laravel-permission/v6/introduction