3. desenvolvimento do aplicativo

O aplicativo foi desenvolvido utilizando-se do framework Ionic em sua versão 4.0. O funcionamento do aplicativo inicia-se com a identificação do usuário. Para a autenticação, o usuário deverá fornecer suas credenciais compostas por um e-mail e uma senha previamente cadastrados, caso o usuário ainda não seja cadastrado no sistema, será possível realizar o seu cadastro através da opção “Registrar” que se encontra na tela de “login”.

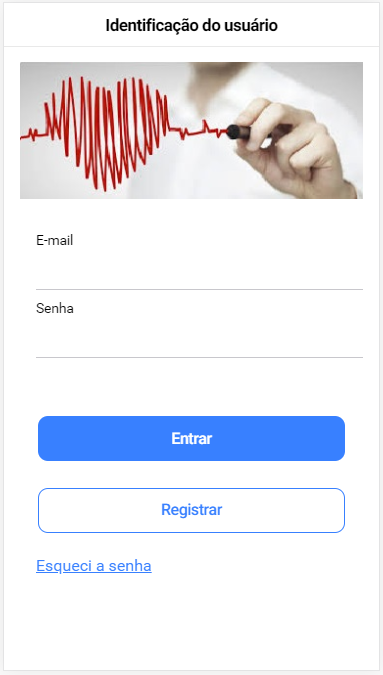
3.1 O registro do usuário

Para o registro do usuário, serão solicitadas algumas informações obrigatórias (Nome, e-mail, senha, telefone, número do cpf e número do rg). Apenas após o preenchimento e validação destas informações será habilitado o botão “Registrar”, que possibilitará a efetivação do cadastro (registro) do usuário. O procedimento “Registrar usuário” irá consumir uma API (*https://api-qlife.herokuapp.com/api/v1/usuario*) através do seu método POST que fará novamente as validações necessárias e retornará um status indicando o sucesso ou não do registro do usuário.

3.2 A recuperação de senhas esquecidas

Ainda relacionada ao login do usuário, outra funcionalidade oferecida pelo aplicativo é a recuperação de senha esquecida pelo usuário. Este acesso está disponível na tela de “Identificação do usuário” através do link “Esqueci minha senha” que acionará uma nova página, a partir da qual um usuário já cadastrado informará o seu e-mail. Ao “Solicitar nova senha” o aplicativo irá consumir o método POST da API “*https://api-qlife.herokuapp.com/auth/forgot*” que irá verificar a validade do usuário e se for válido enviará uma nova senha para este e-mail informado que possibilitará o acesso deste usuário ao aplicativo.

3.3 Identificação do usuário (Login)

O acesso ao aplicativo se dará através de uma autenticação do usuário informando suas credenciais (e-mail e senha) a partir da tela de “Identificação do usuário”. Após inserir suas credenciais o aplicativo faz uma requisição ao método POST da API [*https://api-qlife.herokuapp.com/login*](https://api-qlife.herokuapp.com/login) *(envia as credenciais do usuário no cabeçalho da requisição)* e aguarda o retorno da requisição (response). Em caso de insucesso o aplicativo informa ao usuário a falha no acesso e aguarda por novas tentativas. Em caso de sucesso na autenticação do usuário, a API retorna o status de sucesso (200) e também retorna um **token** do tipo **JWT** que ficará armazenado no **storage** da aplicação enquanto durar a seção. Este token é de suma importância, porque a partir deste ponto da aplicação, todas as requisições à API, obrigatoriamente deverão receber uma identificação do usuário, uma assinatura e esta identificação será este token. Esta assinatura será incluída no cabeçalho de todas as requisições http. Devido a grande quantidade de requisições que podem ser realizadas, a inclusão desta assinatura ocorrerá automaticamente através de um **interceptor** que será abordado a seguir neste documento.

3.4 Interceptos

Conforme o próprio nome sugere, o interceptor possibilita a interrupção de uma requisição HTTP e faça alguma modificação na mesma caso seja necessário. Neste projeto, por motivo de organização, temos uma pasta chamada “Interceptors”, a qual irá conter todos os *interceptors* que criarmos.

Teremos dois interceptos neste projeto: um interceptor de autenticação (auth-interceptor.ts), que irá interromper todas as requisições HTTP feitas pelo aplicativo e inserir a assinatura (token) recebida no login e (que se encontra gavada na storage) em todas as requisições. Teremos também um interceptor de erro (error-interceptor.ts) que terá uma atuação semelhante ao anterior, porém quando a requisição for recebida na volta. Este analisará e possíveis erros retornados e fará o devido tratamento. Desta forma evita a necessidade do tratamento do mesmo erro em todas as requisições individualmente.

Os grandes benefícios dos interceptors são a centralização e a redução de códigos de programas escritos que facilitam possíveis alterações e aprimoramento do aplicativo.

3.5 Local Storage

Um recurso que será utilizado durante o andamento da aplicação é a “local storage”, através deste recurso, informações poderão ser gravadas localmente na aplicação e poderão ser recuperadas quando necessário. É importante salientar que tais informações não serão perdidas com o encerramento da sessão atual, de forma que o aplicativo poderá ser encerrado e da próxima vez que for iniciado estas informações poderão ser obtidas novamente. Obviamente, estas informações poderão ser eliminadas quando julgar necessário.

Um exemplo do uso deste recurso é o token que é utilizado no cabeçalho da maioria das APIs utilizadas por este aplicativo. Quanto feita a autenticação do usuário na tela inicial (identificação do usuário), um token é gravado no “local storage” e a partir deste momento, em todas as requisições HTTP em que for necessário informar o token no cabeçalho da requisição, este token é recuperado do “local storage” onde foi gravado.

3.6 Iniciando a aplicação

Após sucesso na validação do usuário na tela de login (identificação do usuário) um token é retornado (Response / headers) e gravado no local storage, conforme citado anteriormente. Na sequência é invocada o método GET da API “https://api-qlife.herokuapp.com*/api/v1/usuario*” que retornará algumas informações do usuário que fez login em uma arquivo no formato JSON (JavaScript Object Notation). O conteúdo retornado (Response) desta API é analisado e se o usuário possuir um “Perfil Pessoal”, a aplicação será direcionada para a tela de “Perfil Pessoal” do usuário, caso ele ainda não possua um Perfil Pessoal, a aplicação será direcionada para a tela de ”Dados do usuário”.

3.7 Dados do usuário

A tela de “Dados do usuário” traz uma representação do funcionamento geral de toda a solução que faz uso de APIs para criar, buscar, alterar, excluir dados.

- Para obter dados de um objeto (especificamente neste caso, um usuário) busca as informações através do método GET de alguma API, passando algum parâmetro quando necessário.

3.6 Serviços

Algumas funcionalidades da aplicação s

3.7 Uso de APIs

Sadfasd diagramas da