Utilizando Observable

\*\*EventEmmiter tamém é um observable, por isso podemos utilizar o subscribe nele.

Utilizando o timer do observable para esconder a snackbar:

import { Observable } from 'rxjs/Observable';

import 'rxjs/add/observable/timer'

  this.notificationSrvc.notifier.subscribe(message=>{

      this.message = message;

      this.snackVisibility = 'visible';

      Observable.timer(3000).subscribe(timer=>{

         this.snackVisibility ='hidden'

      });

Quando a mudança ocorrer o evento será avisado e atualizado aos inscritos nele.

Criando um observable para atualizar o menu do app quando trocar de usuário:

Serviço:

import { HttpClient } from '@angular/common/http';

import { Injectable, EventEmitter } from '@angular/core';

import { EmitterVisitorContext } from '@angular/compiler';

/\*

  Generated class for the UpdateUserProvider provider.

  See https://angular.io/guide/dependency-injection for more info on providers

  and Angular DI.

\*/

export class UpdateUserProvider {

 newUser = new EventEmitter()

 updateUser(){

   //console.log('USUARIO NO SERVICO UPDATE', user)

   this.newUser.emit();

 }

}

No login chamamos o o evento:

 async login(): Promise<void> {

    this.navCtrl.setRoot(this.navCtrl.getActive().component);

    let result = await this.usuarioSrvc.autenticate(this.form.email, this.form.senha);

    console.log('Teste',result);

    if(result.success){

       UsuarioProvider.RegisterLogin(result.data);

       console.log('Usuario logado no login', configHelper.storageKeys.user, JSON.stringify(result.data.usuario))

       UsuarioProvider.RegisterLogin(result.data)

      //  this.updateUserSrvc.updateUser( JSON.stringify(result.data.usuario))

      this.updateUserSrvc.updateUser();

      //this.events.publish(configHelper.Events.atualizacaoUserMenu)

      this.navCtrl.setRoot('TabsPage')

    }

  }

E no componente fazemos o subscribe dele:

  this.updateUserSrvc.newUser.subscribe(user=>{

      this.atualizarUser();

    console.log('USuario \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*', user)

    // this.atualizarUser(user)

  })