```
import { Injectable } from '@angular/core';
import { v1 as uuidv1 } from 'uuid';

@Injectable({
  providedIn: 'root'
})
export class LocalStoreService {

  private ls = window.localStorage
  constructor() { }

  public setItem(key, value) {
    value = JSON.stringify(value)
    this.ls.setItem(key, value)
    return true
  }

  public getItem(key) {
    let value = this.ls.getItem(key)
    try {
      return JSON.parse(value)
    } catch (e) {
      // console.log(e)
      return null
    }
  }

  public clear() {
    this.ls.clear();
  }

  public getDeviceUuid(){
    var uuid = this.getItem('device-id');

    if(!uuid) {
      uuid = uuidv1();
      this.setItem('device-id', uuid);
    }

    return uuid;
  }
}
```

________________________xxx__——______

```
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { ApiRequestOptions } from '../models/request-
options.model';
import { LocalStoreService } from './local-store.service';
import { Router } from '@angular/router';
import * as moment from 'moment';
import * as _ from "lodash";
import { Utils } from '../utils';
import { ToastrService } from 'ngx-toastr';

@Injectable({
  providedIn: 'root'
})
export class ApiClientService {

  user: any;

  constructor(
    private http: HttpClient,
    private storage: LocalStoreService,
    private router: Router,
    private toastr: ToastrService,
  ) {
    this.user = this.storage.getItem('Procity:user');
  }

  private defaultErrorHandler(error, options:
ApiRequestOptions) {
    console.error(error);

    if (error.status === 401) {
      this.router.navigateByUrl('/sessions/signin');
      return;
    }

    if (error.status === 403) {

      this.router.navigateByUrl('others/erro', {
        state: {
          status: error.status,
          message: 'Desculpe, você não possui permissão para
acessar esse recurso',
          ticket: '8ksrf83'
        }
      });

      return;
    }

    if (error.status === 422 || error.status === 400) {

      console.log(error);

      if (options.formGroup) {
        //this.toastr.success('Algo deu errado durante essa
opera', 'Oops ... ', { timeOut: 3000 });
        Utils.dispatchErrors(options.formGroup,
error.error.errorMessages);
        return;
      }
    }

    this.router.navigateByUrl('others/erro', {
      state: {
        status: error.status,
        message: 'Um condição inesperada pela aplicação
causou um erro durante sua requisição.',
        ticket: '8ksrf83'
      }
    });
```

```
  }

  private defaultSuccessHandler(result) {

  }

  request() {

  }

  get(baseUrl: string, action: string, options?:
ApiRequestOptions) {

    let defaultHeaders: any = {};
    options = options || new ApiRequestOptions();

    if (this.user) {
      defaultHeaders.Authorization = `Bearer $
{this.user.authToken}`;
      defaultHeaders.iid = this.user.iid || '-';
    }

    defaultHeaders =
{ ...defaultHeaders, ...options.headers };

    this.http.get(baseUrl + action, {
      headers: defaultHeaders,
      params: options.params
    })
      .toPromise()
      .then(result => {
        options.successCallback ?
options.successCallback(result) :
this.defaultSuccessHandler(result);
      })
      .catch(error => {
        options.errorCallback ?
options.errorCallback(error) :
this.defaultErrorHandler(error, options);
```

```
      });
  }

  post(baseUrl: string, action: string, options?:
ApiRequestOptions) {

    let defaultHeaders: any = {};
    options = options || new ApiRequestOptions();

    if (this.user) {
      defaultHeaders.Authorization = `Bearer $
{this.user.authToken}`;
      defaultHeaders.iid = this.user.iid || "-";
    }

    defaultHeaders =
{ ...defaultHeaders, ...options.headers };

    console.log(baseUrl + action, options.params,
defaultHeaders);

    this.http.post(baseUrl + action, options.params, {
      headers: defaultHeaders
    })
      .toPromise()
      .then(result => {
        options.successCallback ?
options.successCallback(result) :
this.defaultSuccessHandler(result);
      })
      .catch(error => {
        options.errorCallback ?
options.errorCallback(error) :
this.defaultErrorHandler(error, options);
      });
  }
  put(baseUrl: string, action: string, options?:
ApiRequestOptions) {
    let defaultHeaders: any = {};
    options = options || new ApiRequestOptions();

    if (this.user) {
      defaultHeaders.Authorization = `Bearer $
{this.user.authToken}`;
      defaultHeaders.iid = this.user.iid || "-";
    }

    defaultHeaders =
{ ...defaultHeaders, ...options.headers };

    console.log(baseUrl + action, options.params,
defaultHeaders);

    this.http.put(baseUrl + action, options.params, {
      headers: defaultHeaders
    })
      .toPromise()
      .then(result => {
        options.successCallback ?
options.successCallback(result) :
this.defaultSuccessHandler(result);
      })
      .catch(error => {
        options.errorCallback ?
options.errorCallback(error) :
this.defaultErrorHandler(error, options);
      });
  }

  datesToPtBr(result: any) {

    console.log(result);

    if (!result || !result.value)
      return;

    //Array
    if (result.value.length) {
      console.log('iei.. array  ');
      result.value.forEach(element => {
        this.datesToPtBr({ value: element });
      });
      return;
    }

    //Object
    _.map(result.value, (v, key) => {
      var d = moment(v, 'YYYY-MM-DD');
      if (d.isValid()) {
        console.log(v, key, 'valid... ');
        result.value[key] = d.format('DD/MM/YYYY')
      }
    });
  }

  datesFromPtBr(params: any) {

    if (typeof (params) !== 'object')
      return;

    //Object
    var newObj = {};
    _.map(params, (v, key) => {

      var d = moment(v, 'DDMMYYYY');
      if (d.isValid() && v.length === 8) {
        newObj[key] = d.format('YYYY-MM-DD')
      } else {
        newObj[key] = v;
      }
    });

    return newObj;
  }
}
```

____________XXX_________

```
import { Injectable } from '@angular/core';
import { LocalStoreService } from './local-store.service';
import { Router } from '@angular/router';
import { ApiClientService } from './api-client.service';
import { environment } from 'src/environments/environment';
import { ApiRequestOptions } from '../models/request-
options.model';
import { HttpClient } from '@angular/common/http';

@Injectable({
  providedIn: 'root'
})
export class AuthService {
  authenticated: boolean = false;
  sindicato: boolean = false;
  user: any;

  constructor(
    private store: LocalStoreService,
    private router: Router,
    private apiClient: ApiClientService,
    private http: HttpClient
  ) {
    this.checkAuth();
  }

  checkAuth() {
    this.authenticated =
this.store.getItem('Procity:authenticated');
    this.user = this.store.getItem('Procity:user');
    this.sindicato = this.store.getItem('Procity:sindicato');
  }
  getuser() {
    return this.user;
  }

  createSession(user: any) {
    this.authenticated = true;
    this.sindicato = user.sindicato;
    this.user = user;
    this.store.setItem('Procity:authenticated', true);
    this.store.setItem('Procity:sindicato', user.sindicato);
    this.store.setItem('Procity:user', user);
  }

  basicSignin(login: string, senha: string, successCallback: 
Function, errorCallback: Function) {

      this.http.post(environment.apiAuth + 'api/
AutenticacaoTask/signin', {
        login: login,
        senha: senha,
        uUid: 'Web (N/A)',
        userAgent: 'Web (N/A)',
        so: 'Web (N/A)',
        versao: 'Web (N/A)',
        marca: 'Web (N/A)',
        modelo: 'Web (N/A)',
      })
        .toPromise()
        .then(result => {
          successCallback(result);
        })
        .catch(error => {
          errorCallback(error);
        });
  }

  signout() {
    this.authenticated = false;
    this.store.setItem('Procity:authenticated', false);
    this.store.setItem('Procity:sindicato', false);
    this.store.setItem('Procity:user', null);
    this.router.navigateByUrl('/sessions/signin',
{skipLocationChange: true});
  }
}
```

_________ xxx ______

```
import { Injectable } from '@angular/core';
import { CanActivate, Router } from '@angular/router';
import { AuthService } from './auth.service';

@Injectable({
  providedIn: 'root'
})
export class AuthGaurd implements CanActivate {

  constructor(
    private router: Router,
    private auth: AuthService
  ) { }

  canActivate() {
    if(this.auth.authenticated) {
      return true;
    } else {
      this.router.navigateByUrl('/sessions/signin');
    }
  }
}
```

______ xxx________

```
import { Injectable } from '@angular/core';
import { Observable, of } from 'rxjs';
import { ApiClientService } from './api-client.service';
import { HttpClient } from '@angular/common/http';
import { FormGroup } from '@angular/forms';
import { ApiRequestOptions } from '../models/request-
options.model';
import { environment } from 'src/environments/environment';

interface Motorista {
  data: any[];
}

@Injectable({
  providedIn: 'root'
})
export class MotoristaService {

  constructor(
    private apiClient: ApiClientService,
    private http: HttpClient
  ) { }

  inserir(motorista: any, success: Function, formGroup:
FormGroup) {

    var options = new ApiRequestOptions(motorista, success);
    options.formGroup = formGroup;
    this.apiClient.post(environment.apiMain, 'api/
motoristatask/registrar-motorista-web', options);
  }

  getById(id: any, success: Function) {

  }

  excluir(idmotorista: any, success: Function) {

  }
  getAll (success: Function){
    var options = new ApiRequestOptions(null, success);
    this.apiClient.get(environment.apiMain, 'api/
MotoristaQuery', options);
  }
  getAllPositions (success: Function, idEmpresa){
    var options = new ApiRequestOptions({idEmpresa},
success);
    this.apiClient.get(environment.apiMain, 'api/
GeolocalizacaoQuery', options);
  }
}
```

______XXX ______

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { DashboardAnalyticsComponent } from './dashboard-
analytics/dashboard-analytics.component';
import { DashboardSalesComponent } from './dashboard-sales/
dashboard-sales.component';
import { DashboardCampaignComponent } from './dashboard-
campaign/dashboard-campaign.component';
import { DashboardFinanceComponent } from './dashboard-
finance/dashboard-finance.component';
import { DashboardStockComponent } from './dashboard-stock/
dashboard-stock.component';
import { DashboardProductsComponent } from './dashboard-
products/dashboard-products.component';
import { DashboardEventComponent } from './dashboard-event/
dashboard-event.component';
import { DashboardSindicatoComponent } from './dashboard-
sindicato/dashboard-sindicato.component';
import { DashboardBonusComponent } from './dashboard-bonus/
dashboard-bonus.component';
import { DashboardSindicatoMapaComponent } from './dashboard-
sindicato-mapa/dashboard-sindicato-mapa.component';
import { AuthSindicato } from 'src/app/shared/services/
auth.sindicato';
```

```
import { AuthSuperUser } from 'src/app/shared/services/
auth.su';

const routes: Routes = [
  {
    path: 'v1',
    component: DashboardAnalyticsComponent
  },
  {
    path: 'v2',
    component: DashboardSalesComponent
  },
  {
    path: 'v3',
    component: DashboardCampaignComponent
  },
  {
    path: 'v4',
    component: DashboardFinanceComponent
  },
  {
    path: 'v5',
    component: DashboardStockComponent
  },
  {
    path: 'v6',
    component: DashboardProductsComponent
  },
  {
    path: 'v7',
    component: DashboardEventComponent
  },
  {
    path: 'sindicato',
    canActivate:[AuthSindicato],
    component: DashboardSindicatoMapaComponent
  },
  {
    path: 'mapa',
    canActivate:[AuthSindicato],
    component: DashboardSindicatoMapaComponent
  },
  {
    path: 'v8',
    component: DashboardBonusComponent
  }
];

@NgModule({
  imports: [RouterModule.forChild(routes)],
  exports: [RouterModule]
})
export class DashboardRoutingModule { }
```

__________xxx_______

```
import { NgModule } from '@angular/core';
import { Routes, RouterModule } from '@angular/router';
import { AdminLayoutComponent } from './shared/components/
layouts/admin-layout/admin-layout.component';
import { AuthLayoutComponent } from './shared/components/
layouts/auth-layout/auth-layout.component';
import { AuthGaurd } from './shared/services/auth.gaurd';
import { BlankLayoutComponent } from './shared/components/
layouts/blank-layout/blank-layout.component';

const routes: Routes = [
  {
    path: '',
    redirectTo: 'dashboard/v7',
    pathMatch: 'full'
  },
  {
    path: '',
    component: AuthLayoutComponent,
    children: [
      {
        path: 'sessions',
        loadChildren: './views/sessions/
sessions.module#SessionsModule'
      }
    ]
  },
  {
    path: '',
    component: BlankLayoutComponent,
    children: [
      {
        path: 'others',
        loadChildren: './views/others/
others.module#OthersModule'
      }
    ]
  },
  {
    path: '',
    component: AdminLayoutComponent,
    canActivate: [AuthGaurd],
    children: [
      {
        path: 'dashboard',
        loadChildren: './views/dashboard/
dashboard.module#DashboardModule'
      },
      {
        path: 'uikits',
        loadChildren: './views/ui-kits/ui-
kits.module#UiKitsModule'
      },
      {
        path: 'ecommerce',
        loadChildren: './views/ecommerce/
ecommerce.module#EcommerceModule'
      },
      {
        path: 'forms',
        loadChildren: './views/forms/
forms.module#AppFormsModule'
      },
      {
        path: 'charts',
        loadChildren: './views/charts/
charts.module#ChartsModule'
      },
      {
        path: 'inbox',
        loadChildren: './views/inbox/
inbox.module#InboxModule'
      },
      {
        path: 'calendar',
        loadChildren: './views/calendar/
calendar.module#CalendarAppModule'
      },
      {
        path: 'chat',
        loadChildren: './views/chat/chat.module#ChatModule'
      },
      {
        path: 'tables',
        loadChildren: './views/data-tables/data-
tables.module#DataTablesModule'
      }
    ]
  },
  {
    path: '**',
    redirectTo: 'others/404'
  }
```

```
];

@NgModule({
  imports: [RouterModule.forRoot(routes, {useHash: true})],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```