import { ComponentCanDeactivate } from '../../../commons/nri-deactivate.guard';

import { FinanceService } from './../../../commons/nri-auth/services/finance.service';

import { YearOfAccountDTO } from './../../../DTO/YearOfAccountDTO';

import { ClaimUtils } from './../claimUtils';

*// region IMPORTS*

import { ClaimDocument } from './../../../commons/nri-auth/services/documentsInfo.DTO';

import { ClaimStatusDTO } from './../../../DTO/ClaimStatusDTO';

import { ClaimStatusService } from './../../../commons/nri-auth/services/claimStatus.service';

import { CurrencyExchangeService } from './../../../commons/nri-auth/services/currencyExchange.service';

import { DeductibleDTO } from './../../../DTO/DeductibleDTO';

import { DocumentDTO } from './../../../DTO/DocumentDTO';

import { PolicyShareDTO } from './../../../DTO/PolicyShareDTO';

import { NriInputSelectOptions } from './../../../commons/nri-inputs/nri-input-select/nri-input-select.component';

import { ClaimRecoveryDTO } from './../../../DTO/ClaimRecoveryDTO';

import { PaymentEstimateTypeDTO } from './../../../DTO/PaymentEstimateTypeDTO';

import { ClaimPaymentDTO } from './../../../DTO/ClaimPaymentDTO';

import { ClaimsPaymentService } from './../../../commons/nri-auth/services/claimsPayment.service';

import { LocationService } from './../../../commons/nri-auth/services/location.service';

import { InsuranceStructureDTO } from './../../../DTO/InsuranceStructureDTO';

import { CountryDTO } from './../../../DTO/CountryDTO';

import { OrganisationService } from './../../../commons/nri-auth/services/organisation.service';

import { LocationDTO } from './../../../DTO/LocationDTO';

import { OrganisationDTO } from './../../../DTO/OrganisationDTO';

import { CurrencyService } from './../../../commons/nri-auth/services/currency.service';

import { CurrencyDTO } from './../../../DTO/CurrencyDTO';

import { ClaimTypeDTO } from './../../../DTO/ClaimTypeDTO';

import { ClaimTypeService } from './../../../commons/nri-auth/services/claimType.service';

import { PolicyService } from './../../../commons/nri-auth/services/policy.service';

import { ClaimsService } from './../../../commons/nri-auth/services/claims.service';

import { PolicyDTO } from './../../../DTO/PolicyDTO';

import { ClaimDTO } from './../../../DTO/ClaimDTO';

import { Component, OnInit, ViewChild, HostListener, QueryList, ViewChildren } from '@angular/core';

import { NriUser } from './../../../commons/nri-header/nri-header-component';

import { NriGlobalStateService } from './../../../commons/nri-global-state/nri-global-state.service';

import { NriTask, NriComment } from './../../../commons/nri-search/nri-search-task/nri-search-task.component';

import { NriAlert } from './../../../commons/nri-search/nri-search-alert/nri-search-alert.component';

import { \_if } from 'rxjs/observable/if';

import { element } from 'protractor';

import { Router } from '@angular/router';

import { Http } from '@angular/http';

import { Observable } from 'rxjs/Rx';

import { NriAuthService, AdalService, AuthorizationService, AuthHttp, DocumentService, UserService } from '../../../commons/nri-auth';

import { ServerTestService } from '../../../commons/serverTest.service';

import { NriCheckboxItems } from '../../../commons/nri-inputs/nri-input-checkbox/nri-input-checkbox.component';

import { NriRadioOptions } from '../../../commons/nri-inputs/nri-input-radio/nri-input-radio.component';

import { NriTimelineComponent } from './../../../commons/nri-timeline/nri-timeline.component';

import data from './data';

import { isOptionsSelectedGreatherThan, required, maxLenght } from '../../../commons/nri-form/index';

import { SurveyDTO } from '../../../DTO/SurveyDTO';

import { UserDTO } from '../../../DTO/UserDTO';

import { DocumentTypeDTO } from '../../../DTO/DocumentTypeDTO';

import { NriUploadFilesComponent } from '../../../commons/nri-upload-files/nri-upload-files.component';

import { DateUtils } from '../../../Utils/DateUtils';

import { NriInputDatepickerComponent, NriInputMultiSelectOptions, NriInputSelectComponent } from '../../../commons/nri-inputs/index';

import { minDate, maxDate } from '../../../commons/nri-form/validations/index';

import \* as moment from 'moment';

**moment**.**locale**('en');

import { AppInsightsService } from '@markpieszak/ng-application-insights';

import { NriPaginationComponent } from '../../../commons/nri-pagination';

import { NriAccordionComponent } from '../../../commons/nri-accordion';

*// endregion IMPORTS*

interface **DocRow** {

    name: string;

    type: string;

    version: number;

    addedBy: string;

    lastModified: any;

}

@**Component**({

    templateUrl: './create-claims.component.html',

    styleUrls: ['./create-claims.component.scss']

})

export class **CreateClaimsComponent** implements **OnInit**, **ComponentCanDeactivate** {

*// region SETTINGS*

    @**ViewChild**(**NriTimelineComponent**) public timelineRender;

*/\* @ViewChild('currencySelect') public currencySelect; \*/*

    @**ViewChild**('dateOfLossDP') public dateOfLossDP: **NriInputDatepickerComponent**;

    @**ViewChild**('documentSelect') public documentSelect: **NriInputSelectComponent**;

    @**ViewChildren**('paymentEstimate') public paymentEstimate: **QueryList**<**NriInputSelectComponent**>;

    @**ViewChild**(**NriUploadFilesComponent**) public uploadFiles;

    @**ViewChildren**('payments') public payments: **QueryList**<**NriAccordionComponent**>;

    @**ViewChild**('paymentsPagination') public paymentsPagination: **NriPaginationComponent**;

    @**ViewChildren**('recoveries') public acordionRec: **QueryList**<**NriAccordionComponent**>;

    @**ViewChild**('recoveriesPagination') public recoveriesPagination: **NriPaginationComponent**;

    public recoveryIndex: number;

    public recoveriesPageSelected = 1;

    public recoveriesPerPage: number = 5;

    public paymentsPageSelected = 1;

    public paymentsPerPage: number = 5;

    public paymentIndex: number;

    public validDateOfLoss = [**minDate**, **maxDate**];

    public maxDateOfLoss: **Date**;

    public dark: boolean = false;

    public defaultSelectorText: string = 'See all';

    public stepTitles: **Array**<{ check: boolean, name: string, locked: boolean }> = [

        { check: false, name: 'Details', locked: false },

        { check: false, name: 'Payments & Estimates', locked: false },

        { check: true, name: 'Recoveries', locked: false },

        { check: true, name: 'Summary', locked: false },

        { check: true, name: 'Documents', locked: false }

    ];

    public calculationValues: **NriCheckboxItems**[] = [

        { text: 'Co-Insured', value: 'coi' },

        { text: 'RI (Out)', value: 'out' },

        { text: 'RI (in)', value: 'in' },

        { text: '100% NRI', value: 'nri' }

    ];

    public historicRate = {

        from: null,

        to: null,

        date: null

    };

    public ClaimPaymentEstimateTypeCategory = {

        payment: 'PAYMENT',

        estimate: 'ESTIMATE',

        other: 'OTHER'

    };

    public searchPaymentEstimateTypesFilters: {

        category: {}

    };

    public approvedCheckBoxItem: **NriCheckboxItems**[] = [

        { text: 'Approved', value: 'approved' }

    ];

    public isFormValid: boolean = false;

*// Claim Status*

    public readonly INCOMPLETE = 'INCOMPLETE';

    public readonly OPEN = 'OPEN';

    public readonly SETTLED = 'SETTLED';

    public readonly CLOSED = 'CLOSED';

    public readonly AUTHORISATION = 'AUTHORISATION PENDING';

    public readonly SUMMARYINPUTWIDTH = 170;

    public optionSelected: number[] = [];

    public coiSelected: boolean = false;

    public outSelected: boolean = false;

    public inSelected: boolean = false;

    public nriSelected: boolean = false;

    public outValue: string = '';

    public inValue: string = '';

    public showClaimantOnClaimType: string = 'Liability';

    public totalPaymentThreshold = 1 \* Math.**pow**(10, 6); *// 1 million*

*// endregion SETTINGS*

*// region DATA - DETAILS*

    public typesOfClaim: string[] = [];

    public claim: **ClaimDTO** = null;

    public policy: **PolicyDTO** = null;

    public organisation: **OrganisationDTO** = null;

    public allPoliciesNumbers: string[] = [];

*// public allPolicies: PolicyDTO[] = [];*

    public allClaimTypes: **ClaimTypeDTO**[] = [];

    public allClaimTypesSelect: **NriInputSelectOptions**[] = [];

    public allCurrencies: **CurrencyDTO**[] = [];

    public allCurrenciesSelect: **NriInputMultiSelectOptions**[] = [];

    public allDeductibles: **DeductibleDTO**[] = [];

    public allDeductiblesSelect: **NriInputSelectOptions**[] = [];

    public allLossAdjusterOrganisations: **OrganisationDTO**[] = [];

    public allLossAdjusterOrganisationsSelect: **NriInputMultiSelectOptions**[] = [];

    public allLocations: **LocationDTO**[] = [];

    public allLocationsSelect: **NriInputMultiSelectOptions**[] = [];

    public country: string = '';

    public insuredName: string = '';

    public cedingPoolRef: string = '';

    public totalAmountPaid: number = 0;

    public totalOSEstimated: number = 0;

    public showClaimant: boolean = false;

    public allClaimStatus: **ClaimStatusDTO**[];

    public allClaimStatusSelect: **NriInputSelectOptions**[];

    public readonly NOTAPPLICABLE = 'N/A';

    public validation = [ maxLenght ];

    public isCatastropheCodeValid = true;

*// endregion DATA - DETAILS*

*// region DATA - PAYMENTS & ESTIMATES*

    public totalCurrencyEstimate: number = 0;

    public totalCurrencyEstimateFromGBP: number = 0;

    public totalPayments: number = 0.00;

    public totalPaymentsFromGBP: number = 0;

    public totalPaymentAmount: number = 0;

    public paymentEstimateTypesOptions: **NriInputSelectOptions**[] = [];

    public firstPaymentEstimateTypesOptions: **NriInputSelectOptions**[] = [];

*// endregion DATA - PAYMENTS & ESTIMATES*

*// region DATA - RECOVERIES*

    public claimRecoveries: **ClaimRecoveryDTO**[];

    public claimRecoveriesPool: **NriInputSelectOptions**[] = [];

    public claimRecoveriesTotal: number = 0;

    public claimRecoveriesTotalGBP: number = 0;

*// endregion DATA - RECOVERIES*

*// region DATA - SUMMARY*

    public summaryCurrencyOptions = [] as **NriInputSelectOptions**[];

    public summaryCurrentCurrency: **CurrencyDTO**;

    public summaryCurrentCurrencyIndex: number = 0;

*// endregion DATA - SUMMARY*

*// region DATA - DOCUMENTS*

    public claimDocuments: **DocumentDTO**[] = [{} as **DocumentDTO**];

    public uploadComponentUrl: string = process.env.CONTEXT.rest.addClaimFile;

    public documentInfo = [] as **ClaimDocument**[];

    public documentTypesOpt: **NriInputSelectOptions**[] = [];

    public fileAdded = false;

    public docName: string = '';

    public docType: string = '';

    public docTypeId: number;

    public docDescription = '';

    public currentUser: string;

    public userDetails: any;

    public documentRows: **DocRow**[] = [];

    public SortableColumns= [true, true, true, true, true];

    public documentColumns = [

        { name: 'Name' },

        { name: 'Type' },

        { name: 'Version' },

        { name: 'Added by' },

        { name: 'Last modified' }

    ];

    public coinsurance: number = 0;

    public reinsurance: number = 0;

    public netOfDeductible: number = 0;

    public netGrossShare: number = 0;

    public nriNetShare: number = 0;

    public claimsColor: string;

    public currencySelect: **NriInputMultiSelectOptions**[] = [];

    public sizeValidation = [ maxLenght ];

*// endregion DATA - DOCUMENTS*

    public isOpenningClaim = false;

    public fileUploaded: boolean = true;

    public errorMessage: string;

    private workComplete: boolean = false;

    constructor(private adalService: **AdalService**,

                private nriAuthService: **NriAuthService**,

                private serverTestService: **ServerTestService**,

                private authorizationService: **AuthorizationService**,

                private http: **AuthHttp**,

                private router: **Router**,

                private stateService: **NriGlobalStateService**,

                private claimService: **ClaimsService**,

                private policyService: **PolicyService**,

                private claimTypeService: **ClaimTypeService**,

                private claimStatusService: **ClaimStatusService**,

                private currencyService: **CurrencyService**,

                private organisationService: **OrganisationService**,

                private locationService: **LocationService**,

                private claimsPaymentService: **ClaimsPaymentService**,

                private currencyExchangeService: **CurrencyExchangeService**,

                private claimUtils: **ClaimUtils**,

                private documentsService: **DocumentService**,

                private userService: **UserService**,

                private financeService: **FinanceService**,

                private dateUtils: **DateUtils**,

                private appInsightsService: **AppInsightsService**) {

        this.appInsightsService.**startTrackPage**('Claim Create');

        stateService

        .**isLoading**()

        .**subscribe**((promise) => {

            promise.**then**((res) => {

                if (!res) {

                    this.appInsightsService.**stopTrackPage**('Claim Create');

                }

            });

        });

        this.dark = stateService.**getDarkTheme**();

        stateService

            .**isDarkTheme**()

            .**subscribe**((dark) => {

                this.dark = dark;

            });

        this.userService.**getUserDetails**()

            .**then**((details) => {

                this.userDetails = details;

        });

        this.claimsColor = stateService.**getColor**('claims');

*/\* Details Loaders \*/*

        this.**loadPolicies**();

*// this.loadClaimTypes();*

        this.**loadClaimStatus**();

        this.**loadClaim**();

        this.**loadPolicy**();

        this.**loadOrganisation**();

        this.**loadOriginalCurrencies**();

        this.**loadLossAdjusters**();

*/\* Payments & Estiamtes Loaders \*/*

        this.**loadPaymentEstimateTypes**();

*/\* readyOnly Values \*/*

        this.**getLatestEstimate**();

        this.**updatePaymentAmount**();

        this.**updateRecoveryTotal**();

*/\* date of loss max limit\*/*

        const today =  new **Date**();

        this.maxDateOfLoss = today;

    }

    public **ngOnInit**() {

        console.**log**('Create Claims page');

    }

    @**HostListener**('window:beforeunload')

    public **canDeactivate**() {

        if (this.stateService.userTimedOut) {

            return true;

        } else {

            return this.workComplete;

        }

    }

*// region ACTIONS*

    public **onBack**() {

        this.router.**navigate**(['claims']);

    }

    public **onSuccessBack**() {

        this.workComplete = true;

        this.router.**navigate**(['claims']);

    }

    public **onOpen**() {

        this.isFormValid = this.**updateValidations**();

        if (this.isFormValid) {

*/\* if (this.claim.totalPayments > this.totalPaymentThreshold ) {*

*this.claim.statusID = {*

*claimStatusID: this.getClaimStatus(this.AUTHORISATION).claimStatusID*

*} as ClaimStatusDTO;*

*} else {*

*this.claim.statusID = {*

*claimStatusID: this.getClaimStatus(this.OPEN).claimStatusID*

*} as ClaimStatusDTO;*

*} \*/*

            this.**onSave**('open');

        } else {

            this.**openFailureModal**();

        }

    }

*// region VALIDATORS*

    public **updateValidations**() {

        const isDetailsValid = this.claimUtils.**isClaimDetailsValid**(this.claim);

        this.stepTitles[0].check = isDetailsValid;

        this.stepTitles[1].check = !!this.claim.claimPaymentID && this.claim.claimPaymentID.length > 0;

*// let isPaymentValid = this.stepTitles[1].check;*

*// if (!!this.claim.claimPaymentID && this.claim.claimPaymentID.length) {*

*//     this.claim.claimPaymentID.forEach((cp) => {*

*//         isPaymentValid = this.claimUtils.isClaimPaymentsValid(cp);*

*//     });*

*//     this.stepTitles[1].check = isPaymentValid;*

*// }*

*// let isRevoceryValid = this.stepTitles[2].check;*

*// if (!!this.claim.claimRecoveryID && this.claim.claimRecoveryID.length) {*

*//     this.claim.claimRecoveryID.forEach((cr) => {*

*//         isRevoceryValid = this.claimUtils.isClaimRecoveriesValid(cr);*

*//     });*

*//     this.stepTitles[2].check = isRevoceryValid;*

*// }*

*// isDocumentsValid*

        return isDetailsValid; *// && isPaymentValid && isRevoceryValid;*

    }

*// endregion VALIDATORS*

    public **onSave**(type) {

        const thisScope = this;

        thisScope.isOpenningClaim = true;

        switch (type) {

            case 'open':

                if (this.claim.totalPayments > this.totalPaymentThreshold ) {

                    this.claim.statusID = {

                        claimStatusID: this.**getClaimStatus**(this.AUTHORISATION).claimStatusID,

                        claimStatusName: this.**getClaimStatus**(this.AUTHORISATION).claimStatusName

                    } as **ClaimStatusDTO**;

                } else {

                    this.claim.statusID = {

                        claimStatusID: this.**getClaimStatus**(this.OPEN).claimStatusID,

                        claimStatusName: this.**getClaimStatus**(this.OPEN).claimStatusName

                    } as **ClaimStatusDTO**;

                }

                break;

            case 'save':

                this.claim.statusID = {

                    claimStatusID: this.**getClaimStatus**(this.INCOMPLETE).claimStatusID,

                    claimStatusName: this.**getClaimStatus**(this.INCOMPLETE).claimStatusName

                } as **ClaimStatusDTO**;

                break;

        }

        if (this.claim.policyID !== undefined) {

            const policy = {

                id: this.claim.policyID.id,

                reinsuranceOut: this.claim.policyID.reinsuranceOut

            } as **PolicyDTO**;

            this.claim.policyID = policy;

        } else {

            this.**openFailureModal**();

            return;

        }

        if (!this.claim.selfInsuredRetention) {

            this.claim.selfInsuredRetention = 0;

        }

*// this.claim.claimPaymentID = this.claim.claimPaymentID;*

*// this.claim.claimRecoveryID = this.claimRecoveries;*

        this.claimService.**createClaim**(this.claim, false).**subscribe**(

            (res: **ClaimDTO**) => {

                thisScope.claim.claimID = res.claimID;

                thisScope.claim.claimNumber = res.claimNumber;

                thisScope.isOpenningClaim = false;

                if (thisScope.claim.documentID && thisScope.claim.documentID.length > 0) {

                    thisScope.claim.documentID.**forEach**((doc) => {

                        thisScope.documentInfo.**unshift**({

                            claimNumber: res.claimID.**toString**(),

                            name: doc.name,

                            type: doc.documentType.id.**toString**()

                        });

                    });

                    thisScope.uploadFiles.**uploadFiles**();

                } else {

                    this.**openSuccessModal**();

                }

                if (thisScope.claim.statusID.claimStatusName === 'OPEN' && this.claim.policyID.reinsuranceOut) {

                    thisScope.claimUtils.**downloadExcelDocument**(thisScope.claim.claimID);

                }

                thisScope.isOpenningClaim = false;

            },

            (error) => {

                console.**log**(error);

                thisScope.**openFailureModal**();

                thisScope.isOpenningClaim = false;

            }

        );

    }

    public **fileSubmited**() {

        this.**openSuccessModal**();

    }

    public **getDocumentTypeByDomain**() {

        if (this.documentTypesOpt.length === 0) {

            this.documentsService.**getDocumentTypeByDomain**('CLAIM')

            .**then**((response) => {

                    (response as **DocumentTypeDTO**[]).**forEach**((element) => {

                        this.documentTypesOpt.**push**(

                            {

                                value: element.id.**toString**(),

                                text: element.documentType

                            });

                    });

                });

        }

    }

    public **fileSubmitionError**(message, claim) {

        this.fileUploaded = false;

        this.errorMessage = message;

        this.**openSuccessModal**();

    }

    public **activateStep**(step: number) {

        this.timelineRender.**activateStep**(step);

        window.**scrollTo**(0, 0);

    }

    public **onQuickStep**(evt: number) {

        this.isFormValid = this.**updateValidations**();

        window.**scrollTo**(0, 0);

    }

*// endregion ACTIONS*

*// region START DETAILS*

    public **loadClaim**() {

        if (this.claim === null) {

            this.claim = {} as **ClaimDTO**;

*/\* this.claim.statusID = {*

*claimStatusID: this.getClaimStatus(this.INCOMPLETE).claimStatusID*

*} as ClaimStatusDTO; \*/*

        }

        this.**setTotalAmountPaid**();

        this.**setTotalOSEstimated**();

    }

    public **loadPolicy**() {

        if (this.policy === undefined) {

            this.policy = {} as **PolicyDTO**;

        }

    }

    public **loadOrganisation**() {

        if (this.organisation === undefined) {

            this.organisation = {} as **OrganisationDTO**;

        }

    }

    public **loadPolicies**() {

*// Logic to bring only values on dropdown then bring data per each selection*

        this.policyService.**getAllPolicyNumbersWithOpenYear**()

        .**then**((response) => {

            this.allPoliciesNumbers = response as string[];

        });

    }

    public **loadClaimTypes**() {

        if (this.allClaimTypesSelect.length === 0) {

            this.claimTypeService.**getAllClaimTypes**()

                .**then**((response) => {

                    this.allClaimTypes = (response as **ClaimTypeDTO**[]);

                    const claimTypes: **NriInputSelectOptions**[] = this.allClaimTypes.**map**((ct) => {

                        if (!!ct) {

                            return {

                                text: !!ct.claimTypeName ? ct.claimTypeName : ct.claimTypeID.**toString**(),

                                value: ct.claimTypeID.**toString**()

                            } as **NriInputSelectOptions**;

                        }

                    });

                    this.allClaimTypesSelect = claimTypes;

                });

        }

    }

    public **loadClaimStatus**() {

        this.claimStatusService.**getAllClaimStatus**()

            .**then**((response) => {

                this.allClaimStatus = (response as **ClaimStatusDTO**[]);

                const claimStatus: **NriInputSelectOptions**[] = this.allClaimStatus.**map**((cs) => {

                    if (!!cs) {

                        return {

                            text: !!cs.claimStatusName ? cs.claimStatusName : cs.claimStatusID.**toString**(),

                            value: cs.claimStatusID.**toString**()

                        } as **NriInputSelectOptions**;

                    }

                });

                this.allClaimStatusSelect = claimStatus;

            });

    }

    public **getClaimStatus**(name: string): **ClaimStatusDTO** {

        if (this.allClaimStatus) {

            return this.allClaimStatus.**find**((cs) => {

                return cs.claimStatusName === name;

            });

        }

        return undefined;

    }

    public **loadOriginalCurrencies**() {

        if (this.allCurrenciesSelect.length === 0) {

            this.currencyService.**getAllActiveCurrencys**()

                .**then**((response) => {

                    this.allCurrencies = (response as **CurrencyDTO**[]);

                    const currencies: **NriInputMultiSelectOptions**[] = this.allCurrencies.**map**((c) => {

                        if (!!c) {

                            return {

                                name: !!c.name ? c.name : c.id.**toString**(),

                                id: c.id.**toString**()

                            };

                        }

                    });

                    this.allCurrenciesSelect = currencies;

                });

        }

    }

public **loadDeductibles**() {

        if (this.policy.deductibleID) {

            this.allDeductiblesSelect = this.policy.deductibleID.**filter**((dd) =>

                !!dd.id  ).**map**((d) => {

                return {

                   text: (d.deductibleComment ? d.deductibleComment : d.deductibleTypeID.name)

                    + ': ' + (d.amount ? d.amount : '')

                    + ' ' + (d.currencyID ? d.currencyID.code : ''),

                    value: d.id.**toString**()

                } as **NriInputSelectOptions**;

            });

        }else {

            this.allDeductiblesSelect = [];

        }

    }

    public **loadLossAdjusters**() {

        const types: string[] = ['Loss Adjuster'];

*// OLD: getByTypes*

        this.organisationService.**getByTypesMainInfo**(types)

            .**then**((response) => {

                this.allLossAdjusterOrganisations = (response as **OrganisationDTO**[]);

                const lossAdjusterOrganisations: **NriInputMultiSelectOptions**[] = this.allLossAdjusterOrganisations.**map**((o) => {

                    if (o.active === true) {

                        return {

                            name: o.fullName,

                            id: o.id.**toString**()

                        };

                    }

                });

                this.allLossAdjusterOrganisationsSelect = lossAdjusterOrganisations;

                this.allLossAdjusterOrganisationsSelect.**unshift**({

                    name: this.NOTAPPLICABLE,

                    id: this.NOTAPPLICABLE

                });

            });

    }

    public **setPolicy**(policyNumber: any) {

*// OLD: this.policy = this.getPolicy(policyNumber);*

        if (!!policyNumber) {

            this.policyService.**getPolicyByPolicyNumberManualMapped**(policyNumber)

            .**then**((response) => {

                this.policy = (response as **PolicyDTO**);

                this.**setYearOfAccount**();

                this.**setLocations**();

                this.**setCountry**();

                this.**setInsuredName**();

*// this.setCedingPoolRef(); TODO delete after tests*

                this.**setRecoveriesPool**();

                this.**loadDeductibles**();

                this.currencySelect = [];

                if (this.policy.currencyID) {

                    const tempCurrency = [];

                    tempCurrency.**push**({

                        id: this.policy.currencyID.id.**toString**(),

                        name: this.policy.currencyID.name

                    });

                    this.currencySelect = tempCurrency;

                    this.**onCurrencySelected**(this.policy.currencyID.id);

                }

                this.claim.policyID = this.policy;

                this.**setPolicyValues**();

                if (this.claim.dateOfLoss && this.claim.dateOfLoss < this.policy.startDate) {

                    this.claim.dateOfLoss = this.policy.startDate;

                    this.dateOfLossDP.**clearSimple**();

                }

            });

        }

    }

    public **setCedingPoolRef**(cedingPoolRef: string) {

        this.claim.ceedingPoolRef = cedingPoolRef;

    }

*/\**

*private setCedingPoolRef() {*

*this.cedingPoolRef = this.policy.otherPolicyRefs;*

*}*

*\*/*

    public **setCatastropheCode**(catastropheCode: string) {

        this.claim.catastropheCode = catastropheCode;

    }

    public **setCatastropheCodeValidation**(valid: boolean) {

        this.isCatastropheCodeValid = valid;

    }

    public **setClaimType**(claimTypeId: any) {

        this.claim.claimTypeID = this.**getClaimType**(claimTypeId);

        this.showClaimant = this.claim.claimTypeID.claimTypeName.**indexOf**(this.showClaimantOnClaimType) !== -1;

        this.**updateValidations**();

    }

    public **setLossAdjuster**(organisationId: any) {

        if (organisationId !== this.NOTAPPLICABLE) {

            this.organisation = this.**getOrganisation**(organisationId[0]);

            this.claim.lossAdjuster = this.organisation;

        } else {

            this.claim.lossAdjuster = undefined;

        }

    }

    public **setTotalAmountPaid**() {

        if (!!this.claim.claimPaymentID && this.claim.claimPaymentID.length > 0) {

            this.totalAmountPaid = 0;

            this.claim.claimPaymentID.**forEach**((element: **ClaimPaymentDTO**) => {

                if (!!element.paymentEstimateType && element.paymentEstimateType.name.**toLowerCase**() === 'payment') {

                    this.totalAmountPaid += element.paymentAmount;

                }

            });

        }

    }

    public **setTotalOSEstimated**() {

        if (this.claim.claimNumber) {

            const searchFilter = {

                claimNumber: this.claim.claimNumber

            };

            this.claimsPaymentService.**getFilteredClaimPayments**(searchFilter)

                .**subscribe**((response) => {

                    const claimPayment = (response as **ClaimPaymentDTO**[]).**map**((cp) => {

*// if (cp.paymentEstimateType && cp.paymentEstimateType.paymentEstimateTypeID !== "3") {*

*//     this.totalCurrencyEstimate += cp.o\_estimate;*

*//     this.totalPaymentAmount += cp.paymentAmount;*

*// }*

                    });

                    this.totalOSEstimated = this.totalCurrencyEstimate - this.totalPaymentAmount;

                });

        }

    }

    public **onCurrencySelected**(evt: any) {

        if (evt !== undefined) {

            if (evt.length === 0) {

                this.claim.currency = undefined;

                this.summaryCurrentCurrency = undefined;

            } else {

                this.allCurrencies.**some**((c) => {

                    if (**String**(c.id) === **String**(evt)) {

                        this.claim.currency = c;

                        return true;

                    }

                });

                this.summaryCurrentCurrency = this.claim.currency;

                this.**updateSummaryCurrencyOptions**();

                this.**updateSummary**();

            }

        }

    }

    public **currencySelected**(evt: any) {

        if (evt === 'GBP') {

            this.summaryCurrentCurrency = {

                name: 'Sterling Pound',

                code: 'GBP'

            } as **CurrencyDTO**;

        } else {

            this.summaryCurrentCurrency = this.claim.currency;

        }

        this.**updateSummary**();

    }

    public **getLatestEstimate**() {

        this.claim.currentEstimate = this.claimUtils.**getLatestEstimate**(this.claim)[0];

        this.claim.currentEstimateGBP = this.claimUtils.**getLatestEstimate**(this.claim)[1];

        this.**updateSummary**();

        this.**updateValidations**();

    }

    public **getAdditionalFees**() {

        return !!this.claim.additionalFees ? this.claim.additionalFees : 0;

    }

    public **setDateOfLoss**(evt: string) {

        const datePicked = new **Date**(evt).**getTime**();

        if (datePicked) {

            if (datePicked < new **Date**(this.policy.startDate).**getTime**() || datePicked > this.maxDateOfLoss.**getTime**()) {

                this.dateOfLossDP.**clearSimple**();

            } else {

                this.claim.dateOfLoss = datePicked;

            }

        }else {

            this.claim.dateOfLoss = undefined;

        }

        this.**updateValidations**();

    }

    public **convertToDate**(date: string) {

        if (date) {

            return new **Date**(date);

        }

        return undefined;

    }

    public **setDescription**(evt: any) {

        if (evt !== undefined) {

            this.claim.description = evt;

        }

    }

    public **setAdditionalFees**(evt: any) {

        this.claim.additionalFees = evt as number;

    }

    public **setSelfInsuredRetention**(evt: any) {

        this.claim.selfInsuredRetention = evt as number;

    }

    public **setDeductible**(evt: any) {

        const deductible = this.policy.deductibleID.**find**((x) => x.id === +evt);

        this.claim.deductibleID = deductible;

        if (this.policy.rOE && this.policy.currencyID.code && (this.claim.deductibleID.currencyID.code === this.policy.currencyID.code)) {

            this.claim.deductibleID.rate = this.policy.rOE;

        } else {

            const roe = {

                from: this.claim.deductibleID.currencyID.code,

                to: 'GBP',

                date: **moment**(new **Date**(this.policy.rOEDate ? this.policy.rOEDate : this.policy.startDate)).**format**('YYYY-MM-DD')

            };

            this.currencyExchangeService.**historicRate**(roe).**then**((response) => {

                if (!!response) {

                    this.claim.deductibleID.rate = +response;

                }

            });

        }

        this.netOfDeductible = this.claimUtils.**setNetOfDeductibleValue**(this.claim);

        this.**updateSummary**();

    }

    public **setClaimant**(evt: any) {

        this.claim.claimant = evt;

    }

    public **setNotes**(evt: any) {

        if (evt !== undefined) {

            this.claim.notes = evt;

        }

    }

    public **setCountry**() {

        if (!!this.policy.insured && !!this.policy.insured.countryId) {

            this.claim.country = this.policy.insured.countryId.name;

        }

    }

    public **setInsuredName**() {

        if (!!this.policy && !!this.policy.insured) {

            this.claim.insuredName = this.policy.insured.fullName;

        }

    }

    public **setLocation**(evt: any) {

        this.allLocations.**find**((l) => {

            if (**String**(l.id) === **String**(evt)) {

                this.claim.locationID = {

                    id: **Number**(evt),

                    name: l.name

                } as **LocationDTO**;

                this.claim.country = l.country.name;

            }

            return l.id === evt;

        });

    }

    public **getNRINetSharePercentage**() {

        return this.claimUtils.**getNRINetSharePercentage**(this.claim);

    }

*// endregion DETAILS*

*// region START PAYMENTS & ESTIMATES*

    public **loadPaymentEstimateTypes**() {

        this.claimsPaymentService.**getPaymentEstimateTypes**(this.searchPaymentEstimateTypesFilters).**subscribe**((response) => {

            const types = (response as **PaymentEstimateTypeDTO**[]);

            types.**forEach**((type: **PaymentEstimateTypeDTO**) => {

                this.paymentEstimateTypesOptions.**push**({

                    text: type.name,

                    value: type.paymentEstimateTypeID.**toString**()

                } as **NriInputSelectOptions**);

            });

        });

        this.claimsPaymentService.**getPaymentEstimateTypes**(null).**subscribe**((response) => {

            const types = (response as **PaymentEstimateTypeDTO**[]);

            types.**forEach**((type: **PaymentEstimateTypeDTO**) => {

                if (this.**isFirstPaymentEstimateType**(type.name)) {

                    this.firstPaymentEstimateTypesOptions.**push**({

                        text: type.name,

                        value: type.paymentEstimateTypeID.**toString**()

                    } as **NriInputSelectOptions**);

                }

            });

        });

    }

    public **isFirstPaymentEstimateType**(type: string): boolean {

        return this.claimUtils.**isFirstPaymentEstimateType**(type);

    }

    public **setTotalCurrencyEstimate**() {

        this.totalCurrencyEstimate = this.claim.currentEstimate;

    }

    public **setTotalCurrencyEstimateFromGBP**() {

        this.totalCurrencyEstimateFromGBP = this.claim.currentEstimateGBP;

    }

    public **addPayment**() {

        if (!this.claim.claimPaymentID) {

            this.claim.claimPaymentID = [];

        }

        const claimPaym = {} as **ClaimPaymentDTO**;

        if (this.claim.currency) {

            claimPaym.currency = this.claim.currency.name;

        }

        claimPaym.approved = false;

        claimPaym.roe = 1.0;

        this.claim.claimPaymentID.**push**(claimPaym);

        this.paymentIndex = this.claim.claimPaymentID.**indexOf**(claimPaym);

    }

    public **setPaymentEstimateType**(evt: any, index: number) {

        this.claim.claimPaymentID[index].paymentEstimateType = {} as **PaymentEstimateTypeDTO**;

        this.claim.claimPaymentID[index].paymentEstimateType.paymentEstimateTypeID = evt;

        const paymentType = this.paymentEstimateTypesOptions.**find**((x) => x.value === evt);

        this.claim.claimPaymentID[index].paymentEstimateType.name = paymentType ? paymentType.text : '';

        this.claim.claimPaymentID[index].isEstimate =

            this.claim.claimPaymentID[index].paymentEstimateType.name.**toLowerCase**().**includes**('estimate');

        this.claim.claimPaymentID[index].paymentAmount = undefined;

        this.claim.claimPaymentID[index].paymentAmountGBP = undefined;

        this.claim.claimPaymentID[index].o\_estimate = undefined;

        this.claim.claimPaymentID[index].o\_estimateGBP = undefined;

        this.**getLatestEstimate**();

        this.**updatePaymentAmount**();

    }

    public **setPaymentAmount**(value: number, currentPayment: **ClaimPaymentDTO**) {

        if (!currentPayment.isEstimate) {

            currentPayment.paymentAmount = **Number**(value);

            currentPayment.paymentAmountGBP = **Number**((**Number**(value) / currentPayment.roe).**toFixed**(2));

        }

        this.**getLatestEstimate**();

        this.**updatePaymentAmount**();

        this.**setTotalAmountPaid**();

    }

    public **setRateOfExchange**(value: any, index: number) {

        this.claim.claimPaymentID[index].roe = **Number**(value);

        if (this.claim.claimPaymentID[index].o\_estimate && this.claim.claimPaymentID[index].roe) {

            this.claim.claimPaymentID[index].o\_estimateGBP =

**Number**((this.claim.claimPaymentID[index].o\_estimate / **Number**(value)).**toFixed**(2));

        } else {

            this.claim.claimPaymentID[index].o\_estimateGBP = 0;

        }

        if (this.claim.claimPaymentID[index].paymentAmount && this.claim.claimPaymentID[index].roe) {

            this.claim.claimPaymentID[index].paymentAmountGBP =

**Number**((this.claim.claimPaymentID[index].paymentAmount / **Number**(value)).**toFixed**(2));

        } else {

            this.claim.claimPaymentID[index].paymentAmountGBP = 0;

        }

        this.**updatePaymentAmount**();

        this.**getLatestEstimate**();

    }

    public async **getRateOfExchange**(dateOfPayment: string, payment: **ClaimPaymentDTO**) {

        if (!!dateOfPayment) {

            const today = new **Date**();

            let roe = 1;

            const isTomorow = new **Date**(dateOfPayment).**getTime**() > today.**getTime**();

            if (!isTomorow) {

                roe = await this.**setRateOfExchangeByDate**(new **Date**(dateOfPayment));

            } else {

                roe = await this.**setRateOfExchangeByDate**(today);

            }

            payment.roe = roe;

        } else {

            payment.roe = 1;

        }

        if (payment.paymentAmount) {

            payment.paymentAmountGBP = **Number**((payment.paymentAmount / payment.roe).**toFixed**(2));

        } else if (payment.o\_estimate) {

            payment.o\_estimateGBP = **Number**((payment.o\_estimate / payment.roe).**toFixed**(2));

        }

    }

    public async **setDateOfPayment**(dateOfPayment: string, payment: **ClaimPaymentDTO**) {

        let newDateOfPayment;

        if (!!dateOfPayment) {

            newDateOfPayment = new **Date**(dateOfPayment).**getTime**();

        }

        if (newDateOfPayment !== payment.dateOfPayment) {

            payment.dateOfPayment = newDateOfPayment;

            this.claim = this.claimUtils.**orderPaymentsByDate**(this.claim);

            this.**getRateOfExchange**(dateOfPayment, payment);

            this.paymentIndex = this.claim.claimPaymentID.**indexOf**(payment);

            this.**updatePaymentAmount**();

            this.**getLatestEstimate**();

        }

    }

    public **setOSEstimate**(evt: any, index: number) {

        const currentPayment = this.claim.claimPaymentID[index];

        currentPayment.o\_estimate =  **Number**(evt);

        currentPayment.o\_estimateGBP = **Number**((**Number**(evt) / currentPayment.roe).**toFixed**(2));

        this.claimUtils.**refreshPaymentsEstimate**(this.claim);

        this.**getLatestEstimate**();

    }

    public **setApproved**(evt: any, index: number) {

        if (evt.length > 0) {

            this.claim.claimPaymentID[index].approved = true;

            this.claim.claimPaymentID[index].approvalDate = new **Date**();

            const user: **UserDTO** = {} as **UserDTO**;

            user.id = this.userDetails.id;

            user.name = this.userDetails.fullName;

            this.claim.claimPaymentID[index].approvalUser = user;

        } else {

            this.claim.claimPaymentID[index].approved = false;

            this.claim.claimPaymentID[index].approvalDate = null;

            this.claim.claimPaymentID[index].approvalUser = null;

        }

    }

*// endregion END PAYMENTS & ESTIMATES*

*// region START RECOVERIES*

    public **setRecoveriesPool**() {

        this.claimRecoveriesPool = [];

        if (this.policy.policyShareID) {

            this.policy.policyShareID.**forEach**((element: **PolicyShareDTO**) => {

                if (element.organisationID !== undefined && element.insurerRole === 'reout') {

                    this.claimRecoveriesPool.**push**({

                        text: element.organisationID.nRIalias,

                        value: **String**(element.organisationID.id)

                    } as **NriInputSelectOptions**);

                }

            });

        }

    }

    public **addNewRecovery**() {

        if (this.claim.claimRecoveryID === undefined) {

            this.claim.claimRecoveryID = [];

        }

        const recovery = {} as **ClaimRecoveryDTO**;

        recovery.paymenRoE = 1.0;

        this.claim.claimRecoveryID.**push**(recovery);

        this.recoveryIndex = this.claim.claimRecoveryID.**indexOf**(recovery);

    }

    public **removeRecovery**(index: number) {

        this.claim.claimRecoveryID.**splice**(index, 1);

        if (this.recoveriesPageSelected > 1 && (this.recoveriesPageSelected - 1) \* this.recoveriesPerPage ===  this.claim.claimRecoveryID.length) {

            this.recoveriesPagination.**setActive**(this.recoveriesPageSelected - 1);

        }

        if (this.claim.claimRecoveryID.length === 0) {

            this.claim.claimRecoveryID = undefined;

        }

        this.**updateRecoveryTotal**();

    }

    public **setRecoveryPool**(value, index) {

        this.claim.claimRecoveryID[index].pool = value;

        if (this.policy.policyShareID) {

            this.policy.policyShareID.**forEach**((element) => {

                if (element.organisationID !== undefined && element.organisationID.id === **Number**(value)) {

                    this.claim.claimRecoveryID[index].porportion = element.reinsurerPct;

                    this.**setOSBalance**(value, index);

                }

            });

        }

    }

    public **setOSBalance**(value: number, index: number) {

        const currentRecovery = this.claim.claimRecoveryID[index];

        currentRecovery.os\_Balance = value;

        currentRecovery.os\_Balance\_GBP = **Number**((value / currentRecovery.paymenRoE).**toFixed**(2));

    }

    public **setPaid**(value: number, index: number) {

        const currentRecovery = this.claim.claimRecoveryID[index];

        currentRecovery.paymentAmount = value;

        currentRecovery.paymentAmountGBP = **Number**((value / currentRecovery.paymenRoE).**toFixed**(2));

        this.**updateRecoveryTotal**();

    }

    public async **setDueDate**(value: **Date**, index: number) {

        if (this.claim.claimRecoveryID) {

            this.claim.claimRecoveryID[index].dueDate = new **Date**(value).**getTime**();

            const today = new **Date**();

            let roe = 1;

            const isTomorow = new **Date**(value).**getTime**() > today.**getTime**();

            if (!isTomorow) {

                roe = await this.**setRateOfExchangeByDate**(value);

            } else {

                roe = await this.**setRateOfExchangeByDate**(today);

            }

            this.claim.claimRecoveryID[index].paymenRoE = roe;

            if (this.claim.claimRecoveryID[index].paymentAmount) {

                this.claim.claimRecoveryID[index].paymentAmountGBP = **Number**((this.claim.claimPaymentID[index].paymentAmount / roe).**toFixed**(2));

            }

            this.**updateRecoveryTotal**();

        }

    }

    public **setRecoveriesRoe**(value: number, index: number) {

        this.claim.claimRecoveryID[index].paymenRoE = **Number**(value);

        this.claim.claimRecoveryID[index].os\_Balance\_GBP = **Number**((this.claim.claimRecoveryID[index].os\_Balance / **Number**(value)).**toFixed**(2));

        this.claim.claimRecoveryID[index].paymentAmountGBP = **Number**((this.claim.claimRecoveryID[index].paymentAmount / **Number**(value)).**toFixed**(2));

        this.**updateRecoveryTotal**();

    }

*// endregion END RECOVERIES*

*// region START SUMMARIES*

    public **updateSummary**() {

        if (this.summaryCurrentCurrency && this.summaryCurrentCurrency.code === 'GBP') {

            this.summaryCurrentCurrencyIndex = 1;

        } else {

            this.summaryCurrentCurrencyIndex = 0;

        }

        this.**getGrossFGUPaid**();

        this.**getNRIGrossPaid**();

        this.**getNRINetPaid**();

        this.**getGrossFGUOutstanding**();

        this.**getNRIGrossOutstanding**();

        this.**getNRINetOutstanding**();

        this.**getGrossFGUIncurred**();

        this.**getNRIGrossIncurred**();

        this.**getNRINetIncurred**();

        this.**getAdjusterFees**();

        this.**getEngFees**();

        this.**getSolicitorFees**();

        this.**getRecoveriesExpectedRI**();

        this.**getRecoveriesOutstandingRI**();

        this.**getRecoveriesRIRecovered**();

        this.**getTotalGrossFGU**();

        this.**getTotalNRIGross**();

        this.**getTotalNRINet**();

    }

    public **getGrossFGUPaid**() {

        return this.claimUtils.**getGrossFGUPaid**(this.claim)[this.summaryCurrentCurrencyIndex];

    }

    public **getNRIGrossPaid**() {

        return this.claimUtils.**getNRIGrossPaid**(this.claim)[this.summaryCurrentCurrencyIndex];

    }

    public **getNRINetPaid**() {

        return this.claimUtils.**getNRINetPaid**(this.claim)[this.summaryCurrentCurrencyIndex];

    }

    public **getGrossFGUOutstanding**() {

        return this.claimUtils.**getGrossFGUOutstanding**(this.claim)[this.summaryCurrentCurrencyIndex];

    }

    public **getNRIGrossOutstanding**() {

        return this.claimUtils.**getNRIGrossOutstanding**(this.claim)[this.summaryCurrentCurrencyIndex];

    }

    public **getNRINetOutstanding**() {

        return this.claimUtils.**getNRINetOutstanding**(this.claim)[this.summaryCurrentCurrencyIndex];

    }

    public **getGrossFGUIncurred**() {

        return this.claimUtils.**getGrossFGUIncurred**(this.claim)[this.summaryCurrentCurrencyIndex];

    }

    public **getNRIGrossIncurred**() {

        return this.claimUtils.**getNRIGrossIncurred**(this.claim)[this.summaryCurrentCurrencyIndex];

    }

    public **getNRINetIncurred**() {

        return this.claimUtils.**getNRINetIncurred**(this.claim)[this.summaryCurrentCurrencyIndex];

    }

    public **getAdjusterFees**() {

        return this.claimUtils.**getAdjusterFees**(this.claim)[this.summaryCurrentCurrencyIndex];

    }

    public **getEngFees**() {

        return this.claimUtils.**getEngFees**(this.claim)[this.summaryCurrentCurrencyIndex];

    }

    public **getSolicitorFees**() {

        return this.claimUtils.**getSolicitorFees**(this.claim)[this.summaryCurrentCurrencyIndex];

    }

    public **getRecoveriesExpectedRI**() {

        return this.claimUtils.**getRecoveriesExpectedRI**(this.claim)[this.summaryCurrentCurrencyIndex];

    }

    public **getRecoveriesOutstandingRI**() {

        return this.claimUtils.**getRecoveriesOutstandingRI**(this.claim)[this.summaryCurrentCurrencyIndex];

    }

    public **getRecoveriesRIRecovered**() {

        return this.claimUtils.**getRecoveriesRIRecovered**(this.claim)[this.summaryCurrentCurrencyIndex];

    }

    public **getTotalGrossFGU**() {

        return this.claimUtils.**getTotalGrossFGU**(this.claim)[this.summaryCurrentCurrencyIndex];

    }

    public **getTotalNRIGross**() {

        return this.claimUtils.**getTotalNRIGross**(this.claim)[this.summaryCurrentCurrencyIndex];

    }

    public **getTotalNRINet**() {

        return this.claimUtils.**getTotalNRINet**(this.claim)[this.summaryCurrentCurrencyIndex];

    }

    public **onRemovePayment**(index: number) {

        if (!this.claim.claimPaymentID[index].approved) {

            this.claim.claimPaymentID.**splice**(index, 1);

            if (this.paymentsPageSelected > 1 && (this.paymentsPageSelected - 1) \* this.paymentsPerPage ===  this.claim.claimPaymentID.length) {

                this.paymentsPagination.**setActive**(this.paymentsPageSelected - 1);

            }

            if (this.claim.claimPaymentID.length === 0) {

                this.claim.claimPaymentID = undefined;

            }

            if (this.claim.claimPaymentID !== undefined) {

                if (this.claim.claimPaymentID[0] && this.claim.claimPaymentID[0].paymentEstimateType

                    && !this.**isFirstPaymentEstimateType**(this.claim.claimPaymentID[0].paymentEstimateType.name)) {

                        this.claim.claimPaymentID[0].paymentEstimateType.code = '';

                        this.claim.claimPaymentID[0].paymentEstimateType.name = '';

                        this.claim.claimPaymentID[0].paymentEstimateType.paymentEstimateTypeID = -1;

                    }

            }

            this.**updatePaymentAmount**();

            this.**updateRecoveryTotal**();

            this.**getLatestEstimate**();

        }

    }

    public **updatePaymentAmount**() {

        let value: number = 0;

        let valueGBP: number = 0;

        if (this.claim.claimPaymentID) {

            this.claim.claimPaymentID.**forEach**((element: **ClaimPaymentDTO**) => {

                if (element.paymentEstimateType &&

                    element.paymentAmount &&

                    element.paymentEstimateType.name &&

                    !element.paymentEstimateType.name.**toLowerCase**().**includes**('estimate')) {

                    valueGBP += **Number**(element.paymentAmountGBP);

                    value += **Number**(element.paymentAmount);

                }

            });

        }

        this.claimUtils.**refreshPaymentsEstimate**(this.claim);

        this.claim.totalPayments = **Number**(value.**toFixed**(2));

        this.claim.totalPaymentsGBP = **Number**(valueGBP.**toFixed**(2));

*// this.getLatestEstimate();*

        this.**updateSummary**();

    }

*// endregion SUMMARIES*

    public **isOpenDisabled**() {

        return !(this.isFormValid && !this.isOpenningClaim);

    }

*// region DOCUMENTS*

    public **setDocType**(event) {

        if (!!event) {

            this.docTypeId = +event;

        }

    }

    public **activeDocBtn**() {

        if (this.fileAdded && !!this.docTypeId && this.docName !== '') {

            return false;

        } else {

            return true;

        }

    }

    public **setDocName**(event) {

**setTimeout**(() => {

        if (!!event) {

            this.docName = event[0].name;

        }}, 500);

    }

    public **pushDoc**() {

        let document = {} as **DocumentDTO**;

        let documentRow: **DocRow** = {

            name: '',

            type: '',

            version: null,

            addedBy: '',

            lastModified: '',

        };

        this.documentTypesOpt.**forEach**((element) => {

            if (+element.value === this.docTypeId) {

                this.docType = element.text;

            }

        });

        if (!this.claim.documentID) {

            this.claim.documentID = [];

        }

        if (this.fileAdded) {

            documentRow.name = this.docName;

            documentRow.type = this.docType;

            documentRow.version = 0;

            documentRow.addedBy = this.userDetails.givenName;

            documentRow.lastModified = new **Date**();

            document.name = this.docName;

            document.documentType = {} as **DocumentTypeDTO**;

            document.documentType.id = this.docTypeId;

            document.userID = {} as **UserDTO**;

            document.userID.id = this.userDetails.id;

            document.description = this.docDescription;

            document.claim = {

                claimID: this.claim.claimID

            } as **ClaimDTO**;

*/\* document.survey = {*

*surveyID: 1*

*} as SurveyDTO; \*/*

            this.documentRows.**push**(documentRow);

            this.claim.documentID.**push**(document);

        }

        document = {} as **DocumentDTO**;

        documentRow = {} as **DocRow**;

        this.docName = '';

        this.docType = '';

        this.docTypeId = undefined;

        this.fileAdded = false;

        this.docDescription = '';

        this.**hideDocumentMessage**();

        this.documentSelect.**clearField**();

    }

    public **setLatestEstimate**() {

        this.claim.latestEstimate = this.claim.currentEstimate || 0 ;

    }

    public **setCalculatedFieldBasedOnLatestEstimate**() {

        this.coinsurance = this.claimUtils.**setCoinsuranceValue**(this.claim);

        this.reinsurance = this.claimUtils.**setReinsuranceValue**(this.claim);

        this.netOfDeductible = this.claimUtils.**setNetOfDeductibleValue**(this.claim);

        this.netGrossShare = this.claimUtils.**setNetGrossShareValue**(this.claim);

        this.nriNetShare = this.claimUtils.**setNriNetShareValue**(this.claim, this.netOfDeductible);

    }

    public **loadTitle**(payment: **ClaimPaymentDTO**, i: any) {

        return ((!!payment.paymentEstimateType && !!payment.paymentEstimateType.name)

                ? (payment.paymentEstimateType.name ) : '')

            + (!!payment.dateOfPayment && !!payment.paymentEstimateType

                ? (' on ' + this.dateUtils.**convertDateToString**(payment.dateOfPayment)) : '');

    }

*// endregion DOCUMENTS*

    public **checkPaymentClass**(payment: **ClaimPaymentDTO**): string {

        if (!!payment && !!payment.paymentEstimateType && !!payment.paymentEstimateType.paymentEstimateTypeID

            && payment.paymentEstimateType.paymentEstimateTypeID !== -1 && !!payment.dateOfPayment && !!payment.approved) {

            return 'icon-check';

        }

        return 'icon-close';

    }

    public **checkPaymentColor**(payment: **ClaimPaymentDTO**): string {

        if (!!payment && !!payment.paymentEstimateType && !!payment.paymentEstimateType.paymentEstimateTypeID

            && payment.paymentEstimateType.paymentEstimateTypeID !== -1 && !!payment.dateOfPayment && !!payment.approved) {

            return '#4ED285';

        }

        return '#D88B79';

    }

    public **checkRecoveryClass**(recovery: **ClaimRecoveryDTO**): string {

        if (!!recovery && !!recovery.pool && !!recovery.dueDate) {

            return 'icon-check';

        }

        recovery.checked = false;

        return 'icon-close';

    }

    public **checkRecoveryColor**(recovery: **ClaimRecoveryDTO**): string {

        if (!!recovery && !!recovery.pool && !!recovery.dueDate) {

            return '#4ED285';

        }

        return '#D88B79';

    }

    public **isPaymentAproved**(payment): boolean {

        return (payment.approved) === true ? true : false;

     }

     public **paymentApprovalDate**(payment: **ClaimPaymentDTO**): string {

        return (payment && payment.approvalDate) ? this.dateUtils.**convertDateToString**( new **Date**(payment.approvalDate).**getTime**()) : '';

     }

     public **paymentApprovalBy**(payment): string {

        return (payment && payment.approvalUser) ? payment.approvalUser.name : '';

     }

*// region PRIVATES*

    private **hideDocumentMessage**() {

        if (document.**getElementById**('fileReadyNotification') !== null) {

            document.**getElementById**('fileReadyNotification').style.display = 'none';

        }

    }

    private **verifiedVariable**(variable: any) {

        return !!variable ? variable : '';

    }

    private **generateClaimNumber**(policy: **PolicyDTO**, claim: **ClaimDTO**) { *// NRIIMPLEM-102*

        const policyDataToClaim = {

            policyNumber: policy.policyNumber,

            yearOfAccount: policy.yearOfAccount

        };

        this.claimService.**generateClaimNumber**(policyDataToClaim).**then**((response) => {

            this.claim.claimNumber = (response as string);

        });

    }

    private **updateSummaryCurrencyOptions**() {

        this.summaryCurrencyOptions = [];

        if (!!this.claim.currency && this.claim.currency.code !== 'GBP') {

            this.summaryCurrencyOptions.**push**({

                text: this.claim.currency.name,

                value: this.claim.currency.code

            } as **NriInputSelectOptions**);

        }

        const GBP = this.allCurrencies.**find**((c) => {

            return c.code === 'GBP';

        });

        this.summaryCurrencyOptions.**push**({

            text: GBP.name,

            value: GBP.code

        } as **NriInputSelectOptions**);

    }

    private **setYearOfAccount**() {

        this.claim.yearOfAccount = this.policy.yearOfAccount;

    }

    private **setPolicyValues**() {

        if (this.claim.policyID) {

            this.optionSelected = []; *// Clean Array*

            if (this.claimUtils.**isCoInsured**(this.claim)) {

                this.optionSelected.**push**(0); *// Co-insured*

            }

            if (this.claim.policyID.reinsuranceOut) {

                this.optionSelected.**push**(1); *// RI (Out)*

            }

            if (!!this.claim.policyID.insuranceStructureID &&

                this.claim.policyID.insuranceStructureID.insuranceStructureCode === 'REINSURANCE\_INWARDS') {

                this.optionSelected.**push**(2); *// RI (in)*

            }

            if (this.claim.policyID.nriGrossShare && +this.claim.policyID.nriGrossShare === 100) {

                this.optionSelected.**push**(3); *// 100% NRI*

            }

        }

        this.**updateValidations**();

    }

    private async **setRateOfExchangeByDate**(date: **Date**): **Promise**<number> {

        let roe = 1;

        if (this.claim.currency) {

            this.historicRate.from = 'GBP';

            this.historicRate.to = this.claim.currency.code;

            this.historicRate.date = **moment**(new **Date**(date)).**format**('YYYY-MM-DD');

            return this.currencyExchangeService.**historicRate**(this.historicRate).**then**((response) => {

                roe = **Number**(response);

                console.**log**(roe);

                return roe;

            }).**catch**((err) => {

                console.**log**(err);

                return roe;

            });

        } else {

            return roe;

        }

    }

    private **setLocations**() {

        this.allLocations = this.policy.locationId;

        if (this.allLocations) {

            const locations: **NriInputMultiSelectOptions**[] = this.allLocations.**map**((l) => {

                if (!!l && !!l.id) {

                    return {

                        name: !!l.name ? l.name : l.id.**toString**(),

                        id: l.id.**toString**()

                    };

                }

            });

            this.allLocationsSelect = locations;

        }

    }

    private **updateRecoveryTotal**() {

        let value: number = 0;

        let valueGBP: number = 0;

        if (this.claim.claimRecoveryID) {

            this.claim.claimRecoveryID.**forEach**((element) => {

                value += element.paymentAmount ? **Number**(element.paymentAmount) : 0;

                valueGBP += element.paymentAmount ? **Number**(element.paymentAmount) / element.paymenRoE : 0;

            });

        }

        this.claim.totalPaid = **Number**(value.**toFixed**(2));

        this.claim.totalPaidGBP = **Number**(valueGBP.**toFixed**(2));

        this.**updateSummary**();

    }

*/\* Helpers \*/*

    private **getPolicy**(policyNumber: string) {

        if (!!policyNumber) {

            this.policyService.**getPolicyByPolicyNumberManualMapped**(policyNumber)

            .**then**((response) => {

                this.policy = (response as **PolicyDTO**);

            });

        } else {

            this.policy = ({} as **PolicyDTO**);

        }

        return null;

    }

    private **getOrganisation**(organisationId: string) {

        return this.allLossAdjusterOrganisations.**filter**((o) => {

            return o.id.**toString**() === organisationId;

        })[0];

    }

    private **getClaimType**(claimTypeId: string) {

        return this.allClaimTypes.**filter**((ct) => {

            return ct.claimTypeID.**toString**() === claimTypeId;

        })[0];

    }

    private **getClaimPayment**(type: string) {

        const searchFilter = {

            claimNumber: this.claim.claimNumber,

            paymentEstimateType: type

        };

        this.claimsPaymentService.**getFilteredClaimPayments**(searchFilter)

            .**subscribe**((response) => {

                return (response as **ClaimPaymentDTO**[]);

            });

    }

*// endregion PRIVATES*

*// region START MODALS - nri-modal methods*

    private **openSuccessModal**() {

        const modal = document.**getElementById**('successModal');

        const link = modal.**getElementsByClassName**('href-style')[0] as **HTMLElement**;

        link.**click**();

    }

    private **closeSuccessModal**() {

        const modal = document.**getElementById**('successModal');

        const button = modal.**getElementsByClassName**('close')[0] as **HTMLElement**;

        button.**click**();

    }

    private **openWarningModal**() {

        const modal = document.**getElementById**('warningModal');

        const link = modal.**getElementsByClassName**('href-style')[0] as **HTMLElement**;

        link.**click**();

    }

    private **closeWarningModal**() {

        const modal = document.**getElementById**('warningModal');

        const button = modal.**getElementsByClassName**('close')[0] as **HTMLElement**;

        button.**click**();

    }

    private **openFailureModal**() {

        const modal = document.**getElementById**('failureModal');

        const link = modal.**getElementsByClassName**('href-style')[0] as **HTMLElement**;

        link.**click**();

    }

    private **closeFailureModal**() {

        const modal = document.**getElementById**('failureModal');

        const button = modal.**getElementsByClassName**('close')[0] as **HTMLElement**;

        button.**click**();

    }

*// endregion START MODALS - nri-modal methods*

*// region START DOCUMENTS - nri-updload-document methods*

    private **uploadDocEvent**(event) {

        console.**log**(event);

    }

*// endregion START DOCUMENTS - nri-updload-document methods*

*// region HELPERS*

    private **isYearOfAccountOpen**(year: number): boolean {

        let isOpen: boolean = false;

        this.financeService.**getYear**(year).**then**((response) => {

            isOpen = ((response as **YearOfAccountDTO**).status.**toLowerCase**() === 'open');

        });

        return isOpen;

    }

*// endregion HELPERS*

    private **setPaymentOpen**() {

        const nrPayments = this.payments ? this.payments.**toArray**().length : -1;

        if (this.paymentIndex != null && nrPayments > 0) {

            const page = Math.**ceil**(((**Number**(this.paymentIndex) + 1) / this.paymentsPerPage));

            this.paymentsPagination.**setActive**(page);

            if (this.paymentsPageSelected === page) {

**setTimeout**(() => {

                    if (this.paymentIndex != null) {

                        const index = **Number**(this.paymentIndex) - ((page - 1) \* this.paymentsPerPage);

                        const payment = this.payments.**toArray**()[index];

                        if (payment) {

                            payment.**open**();

                            this.paymentIndex = null;

                            this.claimUtils.**scrollToPayment**(index);

                        }

                    }

                }, 500);

            }

        }

    }

    private **setRecoveryOpen**() {

        const nrRecoveries = this.acordionRec ? this.acordionRec.**toArray**().length : -1;

        if (this.recoveryIndex != null && nrRecoveries > 0) {

            const page = Math.**ceil**(((**Number**(this.recoveryIndex) + 1) / this.recoveriesPerPage));

            this.recoveriesPagination.**setActive**(page);

            if (this.recoveriesPageSelected === page) {

**setTimeout**(() => {

                    if (this.recoveryIndex != null) {

                        const index = **Number**(this.recoveryIndex) - ((page - 1) \* this.recoveriesPerPage);

                        const recovery = this.acordionRec.**toArray**()[index];

                        if (recovery) {

                            recovery.**open**();

                            this.recoveryIndex = null;

                            this.claimUtils.**scrollToRecovery**(index);

                        }

                    }

                }, 500);

            }

        }

    }

}