Lab-report-detail

onInit

Costruzione della form dove per la selezione dell’ordine viene passato un oggetto

this.labReport$.subscribe(res => {

this.labReport = res;

this.labReportForm = this.\_fb.group({

dateReport : [this.labReport.dateReport ? new Date(this.labReport.dateReport) : '', Validators.compose([Validators.required])],

description : [this.labReport.description],

kmTransfert : [{value:this.labReport.kmTransfert, disabled:true}],

kmTransfertCost : [{value: this.labReport.kmTransfertCost, disabled:true}],

kmTransfertTotCost : [{value: this.labReport.kmTransfertTotCost, disabled:true}],

//TODO: Servono?

order\_Id : [this.labReport.order\_Id],

sapCodeOrder : [this.labReport.sapCodeOrder],

orderDescription: [this.labReport.sapCodeOrderDescription],

//------------------------------------------------------------

order: [{ value:this.labReport.order , disabled:(!this.labReport.order)}],

Nella part HTML inserisco il component che mi gestisce la selezione dell’ordine :

<div>

<smart-lab-report-order #ordine

formControlName="order"

[logicAreaId]="logicArea"></smart-lab-report-order>

</div>

Lab-report-order

All’interno di @Component inserire l provider :

providers: [

{

provide: NG\_VALUE\_ACCESSOR,

useExisting: forwardRef( () => LabReportOrderComponent ),

multi: true

}

]

La definizione della classe (utilizza ControlValueAccessor) :

export class LabReportOrderComponent implements ControlValueAccessor, OnInit {

@Input() \_order : Order = new Order();

get order() {

return this.\_order;

}

set order(val : Order) {

this.\_order = val;

this.propagateChange(this.\_order);

this.propagateTouched(this.\_order);

}

Aggiungere i seguenti metodi:

//#region CONTROL VALUE ACCESSOR INTERFACE

propagateChange = (\_: any) => { };

propagateTouched = (\_: any) => { };

isDisabled : boolean = false;

setDisabledState(isDisabled: boolean): void{

this.isDisabled = isDisabled;

}

/\*\*

\* Tutti i cambiamenti di valore della form passano per WriteValue

\* @param value

\*/

writeValue(value: Order) {

if (value) {

this.order = {...value};

}else{

this.order = null;

}

}

/\*\*

\* L'evento di Change viene propagato tramite registerOnChange

\* @param fn

\*/

registerOnChange(fn) {

this.propagateChange = fn;

}

/\*\*

\* L'evento di focus viene propagato tramite registerOnTouched

\*/

registerOnTouched(fn) {

this.propagateTouched = fn;

}

//#endregion