**Exemple de formulaire complexe Angular**

*Implémenté avec Material*

*Template :*

<mat-card [formGroup]="mainForm">

<mat-card-title>Inscription</mat-card-title>

<mat-card class="form-card" formGroupName="personalInfo">

<mat-card-subtitle>Informations personnelles</mat-card-subtitle>

<div class="form-row">

<mat-form-field appearance="fill">

<mat-label>Prénom</mat-label>

<input type="text" matInput formControlName="firstName">

<mat-error>Ce champ est requis</mat-error>

</mat-form-field>

<mat-form-field appearance="fill">

<mat-label>Nom</mat-label>

<input type="text" matInput formControlName="lastName">

<mat-error>Ce champ est requis</mat-error>

</mat-form-field>

</div>

</mat-card>

<mat-card class="form-card">

<mat-card-subtitle>Comment préférez-vous être contacté(e) ?</mat-card-subtitle>

<mat-radio-group [formControl]="contactPreferenceCtrl">

<mat-radio-button value="email">Mail</mat-radio-button>

<mat-radio-button value="phone">Telephone</mat-radio-button>

</mat-radio-group>

</mat-card>

<mat-card class="form-card" formGroupName="email" \*ngIf="showEmailCtrl$ | async">

<mat-card-subtitle>Email</mat-card-subtitle>

<div class="form-row">

<mat-form-field appearance="fill">

<mat-label>Email</mat-label>

<input type="text" matInput formControlName="email">

<mat-error>{{ getFormControlErrorText(emailCtrl) }}</mat-error>

</mat-form-field>

<mat-form-field appearance="fill">

<mat-label>Confirmer votre adresse email</mat-label>

<input type="text" matInput formControlName="confirm">

<mat-error>{{ **getFormControlErrorText(confirmCtrl)** }}</mat-error>

</mat-form-field>

</div>

<small class="error-text" \*ngIf="showEmailError$ | async">Les deux adresses ne sont pas identiques</small>

</mat-card>

<mat-card class="form-card" \*ngIf="showPhoneCtrl$ | async">

<mat-card-subtitle>Téléphone</mat-card-subtitle>

<mat-form-field appearance="fill">

<mat-label>Numéro de téléphone</mat-label>

<input type="text" matInput formControlName="phone">

<mat-error>{{ getFormControlErrorText(phoneCtrl) }}</mat-error>

</mat-form-field>

</mat-card>

<mat-card class="form-card" formGroupName="loginInfo">

<mat-card-subtitle>Information de connexion</mat-card-subtitle>

<mat-form-field appearance="fill">

<mat-label>Nom d'utilisateur</mat-label>

<input type="text" matInput formControlName="username">

<mat-error>Ce champ est requis</mat-error>

</mat-form-field>

<div class="form-row">

<mat-form-field appearance="fill">

<mat-label>Mot de passe</mat-label>

<input type="password" matInput formControlName="password">

<mat-error>{{ getFormControlErrorText(passwordCtrl) }}</mat-error>

</mat-form-field>

<mat-form-field appearance="fill">

<mat-label>Confirmer le mot de passe</mat-label>

<input type="password" matInput formControlName="confirmpassword">

<mat-error>{{ getFormControlErrorText(confirmPasswordCtrl) }}</mat-error>

</mat-form-field>

</div>

<small class="error-text" \*ngIf="showPasswordError$ | async">Les mots de passe ne sont pas identiques</small>

</mat-card>

<mat-card-actions \*ngIf="mainForm.valid">

<button mat-flat-button color="primary" (click)="onSubmitForm()" [disabled]="loading">ENREGISTRER</button>

</mat-card-actions>

<mat-spinner \*ngIf="loading" color="primary" mode="indeterminate"></mat-spinner>

</mat-card>

*TypeScript :*

Affichage d'une sous-partie du formulaire selon la sélection dans un autre champ avec changement de validateurs pour les champs impactés

Validator personnalisé comparant la valeur de 2 champs distincts et ajouté sur le form-group contenant les 2 champs concernés

Affichage du message d'erreur personnalisé portant sur la validation des 2 champs du même from-group

Affichage de messages d'erreur personnalisé en fonction du type d'erreur retourné par le controle (form-control ou form-group)

Emission d'une nouvelle valeur de l'observable changeValues uniquement à la sorti d'un champ et non plus à chaque nouvelle frappe

@Component({

selector: 'app-complex-form',

templateUrl: './complex-form.component.html',

styleUrls: ['./complex-form.component.scss']

})

export class ComplexFormComponent implements OnInit {

loading: boolean = false;

mainForm!: FormGroup;

personalInfoForm!: FormGroup;

emailForm!: FormGroup;

loginInfoForm!: FormGroup;

contactPreferenceCtrl!: FormControl;

phoneCtrl!: FormControl;

emailCtrl!: FormControl;

confirmCtrl!: FormControl;

passwordCtrl!: FormControl;

confirmPasswordCtrl!: FormControl;

*// gère l'affichage d'une partie du formulaire selon la valeur séléctionnée dans le champ contactPreferenceCtrl*

showEmailCtrl$!: Observable<boolean>;

showPhoneCtrl$!: Observable<boolean>;

showEmailError$!: Observable<boolean>;

showPasswordError$!: Observable<boolean>;

constructor(private fb: FormBuilder, private formService: ComplexFormService) {}

ngOnInit(): void {

this.initFormControls();

this.initMainForm();

this.initFormObservables();

}

initFormObservables() {

this.showEmailCtrl$ = this.contactPreferenceCtrl.valueChanges.pipe(

startWith(this.contactPreferenceCtrl.value),

map(preference => preference === 'email'),

tap(showEmail => this.setEmailValidators(showEmail))

);

this.showPhoneCtrl$ = this.contactPreferenceCtrl.valueChanges.pipe(

startWith(this.contactPreferenceCtrl.value), *// emmet la 1re valeur par défaut*

map(preference => preference === 'phone'),

tap(showPhone => this.setPhoneValidators(showPhone))

);

*// affiche ou non le message d'erreur lié à la validation des 2 champs du même form-group*

this.showEmailError$ = this.emailForm.statusChanges.pipe(

map(status => status === 'INVALID' && this.emailCtrl.value && this.confirmCtrl.value)

);

this.showPasswordError$ = this.loginInfoForm.statusChanges.pipe(

map(status => this.loginInfoForm.hasError('confirmEqual') && this.passwordCtrl.value && this.confirmPasswordCtrl.value)

)

}

private setEmailValidators(showEmail: boolean) {

if(showEmail) {

this.emailCtrl.addValidators([Validators.required, Validators.email]);

this.confirmCtrl.addValidators([Validators.required, Validators.email]);

} else {

this.emailCtrl.clearValidators();

this.confirmCtrl.clearValidators();

}

this.emailCtrl.updateValueAndValidity();

this.confirmPasswordCtrl.updateValueAndValidity();

}

private setPhoneValidators(showPhone: boolean) {

if(showPhone) {

this.phoneCtrl.addValidators([Validators.required,Validators.minLength(10), Validators.maxLength(10)])

} else {

this.phoneCtrl.clearValidators();

}

this.phoneCtrl.updateValueAndValidity();

}

initMainForm(): void {

this.mainForm = this.fb.group({

personalInfo: this.personalInfoForm,

contactPreference: this.contactPreferenceCtrl,

email: this.emailForm,

phone: this.phoneCtrl,

loginInfo: this.loginInfoForm

});

}

initFormControls(): void {

this.personalInfoForm = this.fb.group({

firstName: ['', Validators.required],

lastName: ['', Validators.required]

});

this.contactPreferenceCtrl = this.fb.control('email', []);

this.phoneCtrl = this.fb.control('', []);

this.emailCtrl = this.fb.control('', []);

this.confirmCtrl = this.fb.control('', []);

this.emailForm = this.fb.group({

**email**: this.emailCtrl,

**confirm**: this.confirmCtrl

}, {

validators: [confirmEqualValidator('**email**', '**confirm**')], *// voir validateur personnalisé*

updateOn: 'blur' *// l'obs valueChanges n'émettra qu'à la sortie d'un champ*

});

this.passwordCtrl = this.fb.control('', [Validators.required]);

this.confirmPasswordCtrl = this.fb.control('', [Validators.required]);

this.loginInfoForm = this.fb.group({

username: ['', Validators.required],

password: this.passwordCtrl,

confirmpassword: this.confirmPasswordCtrl

}, {

validators: [confirmEqualValidator('password', 'confirmpassword')],

updateOn: 'blur'

})

}

onSubmitForm() {

this.loading = true;

this.formService.saveUserInfo(this.mainForm.value).pipe(

tap(saved => {

this.loading = false;

if(saved) {

this.resetForm();

} else {

console.log('Echec de l\'enregistrement des données du formulaire');

}

})

).subscribe();

}

private resetForm() {

this.mainForm.reset();

this.contactPreferenceCtrl.patchValue('email');

}

getFormControlErrorText(ctrl: AbstractControl) {

if(ctrl.hasError('required')) {

return 'Ce champ est requis';

} else if(ctrl.hasError('email')) {

return 'Merci d\'entrer une adresse mail valide';

} else if(ctrl.hasError('minlength')) {

return 'Ce numéro de téléphone ne contient pas assez de chiffre';

} else if(ctrl.hasError('maxlength')) {

return 'Ce numéro de téléphone contient trop de chiffres';

} else {

return 'Ce champ contient une erreur';

}

}

}

***Validator personnalisé comparant les valeurs de 2 champs formulaire***

*Validator à placer sur le form-group contenant les 2 champs (et non pas sur les form-control)*

import { AbstractControl, ValidationErrors, ValidatorFn } from "@angular/forms";

export function confirmEqualValidator(main: string, confirm: string): ValidatorFn {

return (ctrl: AbstractControl): null | ValidationErrors => {

if(!ctrl.get(main) || !ctrl.get(confirm)) {

return {

confirmEqual: 'Invalid control names'

};

}

const mainValue = ctrl.get(main)!.value;

const confirmValue = ctrl.get(confirm)!.value;

return mainValue === confirmValue ? null : {

confirmEqual: {

main: mainValue,

confirm: confirmValue

}

};

};

}