javais commencé le decoupage de ce composant , et jaimerais avois le type scritp et le html du sous composant centre de composition
voici larchitecture que je souhaite avoir
voici le grand composant a decouper

```
# Architecture de Refactoring - Détails Candidature

## Structure Proposée

```
details-candidature/
├── details.component.ts (Parent simplifié)
├── details.component.html
├── components/
│   ├── candidature-sidebar/
│   │   ├── candidature-sidebar.component.ts
│   │   ├── candidature-sidebar.component.html
│   │   └── candidature-sidebar.component.css
│   │
│   ├── candidature-alerts/
│   │   ├── candidature-alerts.component.ts
│   │   ├── candidature-alerts.component.html
│   │   └── candidature-alerts.component.css
│   │
│   ├── onglets/
│   │   ├── informations-personnelles/
│   │   │   ├── informations-personnelles.component.ts
│   │   │   ├── informations-personnelles.component.html
│   │   │   └── informations-personnelles.component.css
│   │   │
│   │   ├── pieces-fournir/
│   │   │   ├── pieces-fournir.component.ts
│   │   │   ├── pieces-fournir.component.html
│   │   │   ├── pieces-fournir.component.css
│   │   │   └── components/
│   │   │   ├── piece-card/
│   │   │   │   ├── piece-card.component.ts
│   │   │   │   └── piece-card.component.html
│   │   │   └── avis-piece/
│   │   │   ├── avis-piece.component.ts
│   │   │   └── avis-piece.component.html
│   │   │
│   │   ├── lieu-depot/
│   │   │   ├── lieu-depot.component.ts
│   │   │   ├── lieu-depot.component.html
│   │   │   └── lieu-depot.component.css
│   │   │
│   │   ├── centre-composition/
│   │   │   ├── centre-composition.component.ts
│   │   │   ├── centre-composition.component.html
│   │   │   └── centre-composition.component.css
│   │   │
│   │   └── epreuves-composition/
│   │   ├── epreuves-composition.component.ts
│   │   ├── epreuves-composition.component.html
│   │   └── epreuves-composition.component.css
```
```

```
|   |
|   └── shared/
|       ├── loading-spinner/
|       |   ├── loading-spinner.component.ts
|       |   └── loading-spinner.component.html
|       └── form-actions/
|           ├── form-actions.component.ts
|           └── form-actions.component.html
```

Responsabilités des Composants

1. **details.component.ts** (Parent)
Responsabilités :
- Orchestration globale
- Gestion de l'état principal (candidature, concours)
- Appels API principaux
- Communication avec le service de paiement TresorPay
- Navigation entre onglets

Données partagées via @Input:
- `candidature`
- `concours`
- `currentCorps`
- `isFormLocked`
- `loadingSubmission`

2. **candidature-sidebar.component**
Responsabilités :
- Affichage des informations du candidat
- Menu de navigation (onglets)
- Barre de progression
- Boutons d'action (Soumettre, Payer)

@Input:
- `candidature: FullCandidature`
- `tabs: Tab[]`
- `activeTab: number`
- `progression: number`
- `btnSubmitIsVisible: boolean`

@Output:
- `tabChange: EventEmitter<number>`
- `submitCandidature: EventEmitter<void>`
- `payerCandidature: EventEmitter<void>`
- `downloadReceipt: EventEmitter<void>`

3. **candidature-alerts.component**
Responsabilités :
- Affichage des alertes contextuelles selon l'étape
- Gestion des messages de validation/rejet

@Input:
- `etapeCode: string`
- `observation: string`

4. **informations-personnelles.component**
Responsabilités :
- Formulaire d'informations personnelles

- Gestion du handicap et type de handicap
- Validation des champs

@Input:
- `candidature: FullCandidature`
- `selectData: any`
- `isFormLocked: boolean`
- `loadingSubmission: boolean`

@Output:
- `submitForm: EventEmitter<FormData>`

5. **pieces-fournir.component**
Responsabilités :
- Liste des pièces à fournir
- Gestion du FormArray de documents
- Ouverture des modales de pièces

@Input:
- `documentTypes: DocumentType[]`
- `candidature: FullCandidature`
- `isFormLocked: boolean`
- `loadingSubmission: boolean`

@Output:
- `submitDocuments: EventEmitter<FormData>`
- `openPieceModal: EventEmitter<number>`
- `previewFile: EventEmitter<string | File>`

Sous-composants :
- **piece-card.component** : Carte individuelle pour chaque pièce
- **avis-piece.component** : Affichage des avis/observations

6. **lieu-depot.component**
Responsabilités :
- Formulaire de sélection du lieu de dépôt
- Liste des directions régionales

@Input:
- `lieuxDepot: any[]`
- `selectedLieuDepot: string`
- `isFormLocked: boolean`
- `loadingSubmission: boolean`

@Output:
- `submitLieuDepot: EventEmitter<string>`

7. **centre-composition.component**
Responsabilités :
- Formulaire de sélection de la commune
- Liste des centres de composition

@Input:
- `centreComposition: any[]`
- `selectedCommune: string`
- `isFormLocked: boolean`
- `loadingSubmission: boolean`

@Output:

- `submitCommune: EventEmitter<string>`

8. **epreuves-composition.component**
Responsabilités :
- Affichage des épreuves obligatoires et facultatives
- Gestion de la sélection exclusive (toggle)
- Validation de la sélection

@Input:
- `epreuves: any[]`
- `epreuvesFacultatif: any[]`
- `selectedEpreuveFacultative: number`
- `isFormLocked: boolean`
- `loadingSubmission: boolean`

@Output:
- `submitEpreuves: EventEmitter<any>`

9. **loading-spinner.component** (Shared)
Responsabilités :
- Écran de chargement réutilisable

@Input:
- `message: string`

10. **form-actions.component** (Shared)
Responsabilités :
- Bouton de soumission réutilisable
- État de chargement

@Input:
- `isValid: boolean`
- `isLoading: boolean`
- `isDisabled: boolean`
- `label: string`

@Output:
- `submit: EventEmitter<void>`

Services à Créer/Refactoriser

1. **CandidatureStateService**
```typescript
@Injectable()
export class CandidatureStateService {
  private candidatureSubject = new BehaviorSubject<FullCandidature>(null);
  candidature$ = this.candidatureSubject.asObservable();

  updateCandidature(candidature: FullCandidature) {
    this.candidatureSubject.next(candidature);
  }
}
```

2. **CandidatureFormService**
```typescript
@Injectable()
export class CandidatureFormService {
  buildInformationsPersonnellesForm(candidature: FullCandidature): FormGroup
```

```
  buildDocumentsForm(documentTypes: DocumentType[]): FormGroup
  buildLieuDepotForm(selectedLieu: string): FormGroup
  buildCommuneForm(selectedCommune: string): FormGroup
  buildEpreuvesForm(selectedEpreuve: number): FormGroup
}
```

Avantages de cette Architecture

✅ **Séparation des responsabilités** : Chaque composant a un rôle clair
✅ **Réutilisabilité** : Composants partagés (loading-spinner, form-actions)
✅ **Testabilité** : Composants plus petits, plus faciles à tester
✅ **Maintenabilité** : Modifications isolées dans les sous-composants
✅ **Lisibilité** : Code plus court et plus compréhensible
✅ **Performance** : Change detection optimisée avec OnPush
✅ **Évolutivité** : Ajout de nouvelles fonctionnalités facilité

Migration Progressive

Phase 1 : Extraction des Composants Simples
1. Loading Spinner
2. Form Actions
3. Candidature Alerts

Phase 2 : Extraction des Onglets
1. Informations Personnelles
2. Lieu de Dépôt
3. Centre Composition

Phase 3 : Composants Complexes
1. Pièces à Fournir (avec sous-composants)
2. Épreuves Composition
3. Sidebar avec navigation

Phase 4 : Services et État
1. CandidatureStateService
2. CandidatureFormService
3. Optimisation des appels API

Exemple d'Utilisation

```typescript
// details.component.ts (simplifié)
@Component({
  selector: 'app-details-candidature',
  template: `
    <app-loading-spinner *ngIf="loading" />

    <div *ngIf="!loading" class="layout">
      <app-candidature-sidebar
        [candidature]="candidature"
        [tabs]="tabs"
        [activeTab]="activeTab"
        [progression]="progression"
        (tabChange)="activeTab = $event"
        (submitCandidature)="envoyerCandidature()"
        (payerCandidature)="payerCandidature()"
      />
```

```
      <main>
        <app-candidature-alerts [etapeCode]="candidature?.etape?.code" />

        <ng-container [ngSwitch]="activeTab">
          <app-informations-personnelles
            *ngSwitchCase="0"
            [candidature]="candidature"
            (submitForm)="onSubmitInfos($event)"
          />

          <app-pieces-fournir
            *ngSwitchCase="1"
            [documentTypes]="documentTypes"
            (submitDocuments)="onSubmitDocs($event)"
          />

          <!-- ... autres onglets -->
        </ng-container>
      </main>
    </div>
  `
})
export class DetailsCandidatureComponent {
  // Propriétés et logique simplifiées
}
```

Cette architecture permettra une meilleure organisation du code et facilitera grandement la maintenance future de votre application.

ceci est le grand composant

```
import { Component, inject } from '@angular/core';
import {
    FormArray,
    FormBuilder,
    FormControl,
    FormGroup,
    FormsModule,
    ReactiveFormsModule,
    Validators
```

```
} from '@angular/forms';
import { ActivatedRoute, Router } from '@angular/router';
import { CorpsARecruter } from
'../../../../core/models/entities/recrutement/corps-a-recruter.model';
import { EspaceCandidatService } from
'../../../../core/services/recrutements/espace-candidat.service';
import { CommonModule } from '@angular/common';
import { RecrutementVueCandidat } from
'../../../../core/models/entities/recrutement/recrutement-vue-candidat.
model';
import {
    successToaster,
    errorToaster,
    errorMessage,
    getExtension,
    generateUUIDv4
} from '../../../../core/helpers/helpers';
import { Recrutement } from
'../../../../core/models/entities/recrutement/recrutement.model';
import { FullCandidature } from
'../../../../core/models/entities/recrutement/candidature.model';
import { Utilisateur } from
'../../../../core/models/entities/recrutement/avis-differe.model';
import { FilePreviewService } from
'../../../../components/commons/file-preview/file-preview.service';
import { StorageService } from
'../../../../core/services/storage.service';
import { DomSanitizer } from '@angular/platform-browser';
import Swal from 'sweetalert2';
import { ErrorType } from
'../../../../core/models/entities/commons.models';
import { PlanGeographiqueRecrutementService } from
'../../../../core/services/recrutements/plan-geo-recrutement.service';
import { Piece } from
'../../../../core/models/entities/referentiel/piece.model';
import { PieceCandidatureModalComponent } from
'../components/piece-candidature-modal/piece-candidature-modal.componen
t';
import { environment } from '../../../../../environments/environment';
import {
    PaymentReportModalComponent,
    paymentStatus
```

```
} from
'../../../../components/commons/payment-report-modal/payment-report-mod
al.component';
import { Document, DocumentType, PieceSoumise, UploadedFile } from
'../..';
import { Select2Data, Select2UpdateEvent, Select2UpdateValue, Select2 }
from 'ng-select2-component';
import { CentreCompositionService } from
'../../../../core/services/recrutements/centre-composition.service';
import { LucideAngularModule } from 'lucide-angular';
import { ActeHandicapComponent } from
"../components/acte-handicap/acte-handicap.component";
import { ButtonLoaderComponent } from
"../../../../components/commons/button-loader.component";
import { TypeHandicapService } from
'../../../../core/services/referentiels/type-handicap.service';
import { StubPagination } from
'../../../../core/models/builders/pagination.builder';
import { PaginationOptions } from
'../../../../core/models/others/http.models';
import { RecrutementService } from
'../../../../core/services/recrutements/recrutement.service';
import { PieceGlobalRecrutementService } from
'../../../../core/services/recrutements/piece-global-recrutement';
import { PieceGlobalRecrutement } from
'../../../../core/models/entities/recrutement/piece-recrutement.model';
declare let TresorPay: any;

@Component({
    selector: 'app-details-candidature',
    imports: [
        CommonModule,
        ReactiveFormsModule,
        FormsModule,
        PieceCandidatureModalComponent,
        PaymentReportModalComponent,
        LucideAngularModule,
        Select2
    ],
    templateUrl: './details.component.html',
    styleUrl: './details.component.css'
})
export class DetailsCandidatureComponent {
```

```
    pageOptions: PaginationOptions = new StubPagination().build();
    onSelectMenu($event: Select2UpdateEvent<Select2UpdateValue>) {
        throw new Error('Method not implemented.');
    }

    constructor(
        private router: Router,
        private route: ActivatedRoute,
        private fb: FormBuilder,
        private sanitizer: DomSanitizer,

        private service: EspaceCandidatService,
        private filePreviewService: FilePreviewService,
        private plangeoRecrutementService:
PlanGeographiqueRecrutementService,
        private centreCompositionService: CentreCompositionService,
        private storageService: StorageService,
        private typeHandicapService: TypeHandicapService,
        private pieceRecrutementService: PieceGlobalRecrutementService
    ) { }



    showPaymentReportModal: boolean = false;

    widget: any;
    htmlMessage: string;
    isPCandidatureFormOpen: boolean;
    paymentTransactionStatus!: paymentStatus;
    btnSubmitIsVisible: boolean = false;
    file: Blob;
    fileUrl: any;
    loading: any;
    uploadedFiles: UploadedFile[] = [];
    showUploadModal: boolean = false;
    newDocument: Document;
    selectedFile: File | null = null;
    selectData: Select2Data[] = [];
    submitted: boolean = false;
    documentTypes: DocumentType[] = [];
    form: FormGroup;
    form1!: FormGroup;
    formAddOtherFile: FormGroup;
```

```
    corps: CorpsARecruter = null;
    listeFile: any[] = [];
    epreuves: any[];
    epreuvesFacultatif: any[];
    isCandidatureFormOpen: boolean = false;
    isChoixCorpsFormOpen: boolean = false;
    isAddFileFormOpen: boolean = false;
    isAddFileAdd: boolean = false;
    loadingSubmission: boolean = false;
    loadingPayment: boolean = false;
    concours: RecrutementVueCandidat = {} as RecrutementVueCandidat;
    recrutement: Recrutement;
    candidature: FullCandidature;
    corpsRecruter: CorpsARecruter;
    utilisateur: Utilisateur;
    isAddFileAddCertificat: boolean = false;
    isAddFileAddActeNaissance: boolean = false;
    isAddFileAddDiplome: boolean = false;
    isAddFileAddCasierJudiciaire: boolean = false;
    isNextForm: boolean = false;
    isPreviousForm: boolean = true;
    isNextFormSignal: boolean = true;
    activeTab = 0;
    selectedFiles: { [key: string]: File } = {};
    formDepot: FormGroup;
    formCommune: FormGroup;
    formEpreuve: FormGroup;
    showPieceModal = false;
    editingPieceIndex: number | null = null;
    pieceModalForm: FormGroup;
    pieceCaracteristiqueForm: FormGroup;
    currentPiece: Piece;
    currentCorpsLabel: string = '';
    currentCorps: CorpsARecruter;
    selectedOptionnel: any = '';
    othersPiece: PieceGlobalRecrutement[] = [];
    lieuxDepot = [
        { code: 'ATL', libelle: 'Direction Régionale Atlantique' },
        { code: 'BOR', libelle: 'Direction Régionale Borgou' },
        { code: 'COU', libelle: 'Direction Régionale Couffo' },
        { code: 'ALI', libelle: 'Direction Régionale Alibori' }
    ];
    centreComposition = [];
```

```
    tabs = [
        { label: 'Informations Personnelles', icon: 'ki-user' },
        { label: 'Pièces à Fournir', icon: 'ki-file-up' },
        { label: 'Lieu de dépôt', icon: 'ki-map' },
        { label: 'Commune composition', icon: 'ki-map' },
        { label: 'Epreuves choisies', icon: 'ki-abstract-26' }
    ];
    pieceCourante: Piece = null;
    valeursExistantes: any = {};
    isLoading = false;
    shouldRequiredAllEpreuves: boolean = false;
    isHandicape: boolean = false;
    showJustificationForm: boolean = false;
    observation: string = '';
    acteJustificatifFile: File | null = null;
    pieceHandicapForm: FormGroup;

    onSelect(event: Select2UpdateEvent, formControl: string) {
        this.form.get(formControl)?.setValue(event.value);
    }

    onFileSelected(event: Event): void {
        const input = event.target as HTMLInputElement;
        if (input.files && input.files.length > 0) {
            this.acteJustificatifFile = input.files[0];
        }
    }

    enregistrerJustification(): void {
        if (!this.acteJustificatifFile || !this.observation) {
            alert('Veuillez fournir tous les champs requis.');
            return;
        }

        // Simulation de soumission
        console.log('Fichier:', this.acteJustificatifFile);
        console.log('Observation:', this.observation);

        // TODO : envoyer les données au backend via un service
        alert('Justification enregistrée avec succès.');
    }

    getData(uuid: string): void {
```

```
        const customFilters = [
            {
                field: 'recrutement.uuid',
                condition: 'eq',
                value: uuid
            }
        ];
        this.pieceRecrutementService.all$({ page: 1 },
customFilters).subscribe({
            next: (response: any) => {
                console.log(response);
                this.othersPiece = response.content;
            }
        });
    }

    ngOnInit(): void {
        this.loading = true;
        const uuid = this.route.snapshot.paramMap.get('id');



        this.service.detailsCandidature(uuid).subscribe({
            next: (data0) => {
                this.candidature = data0.info;
                console.log(this.candidature?.etape?.code);
                this.getData(this.candidature?.recrutement?.uuid);
                this.service

.detailsConcours(this.candidature?.recrutement?.uuid)
                    .subscribe({
                        next: (data) => {
                            this.concours = data;
                            this.btnSubmitIsVisible =
                                this.candidature?.etape?.code ===
'CD1E' ||
                                this.candidature?.etape?.code !==
'CEN';

                            this.currentCorps =
this.concours.corps.find(
                                (cr) =>
                                    cr.corps.id ==
```

```
this.candidature?.corpsSoumission?.corps?.id
                                    );
                                    this.currentCorpsLabel =
this.currentCorps.libelle;
                                    this.epreuvesFacultatif =
                                        this.currentCorps.epreuves.filter(
                                            (ep) => ep.facultatif
                                        );
                                    this.epreuves =
this.currentCorps.epreuves.filter(
                                        (ep) => !ep.facultatif
                                    );

                                    // console.log('Corps courant:',
this.currentCorps);

                                    const trueDiplome = [];
                                    const diplomes = this.currentCorps.diplomes
|| [];
                                    const diplomesCorpsRef =
                                        this.currentCorps.corps?.diplomes ||
[];
                                    if (
                                        diplomes.length > 0 &&
                                        diplomesCorpsRef.length > 0
                                    ) {
                                        diplomesCorpsRef.forEach((liaison) => {
                                            // console.log('Diplome : ',
liaison);
                                            //currentCorps.grade.id ==
liaison.grade.id
                                            if (
                                                diplomes.findIndex(
                                                    (d) => d.id ===
liaison.diplome.id
                                                ) !== -1
                                            ) {
                                                if (
                                                    trueDiplome.findIndex(
                                                        (d) =>
                                                            d.id ===
liaison.diplome.id
```

```
                                    ) === -1
                                ) {
                                    trueDiplome.push(liaison);
                                }
                            }
                        });
                    }

                    // console.log('True diplome:',
trueDiplome);

                    this.documentTypes =
this.currentCorps.pieces
                        .filter(pi => {
                            return !(pi.requisHandicap === true
&& this.candidature?.candidat?.handicap == 'non');
                        })
                        .map(
                            (pi) => ({
                                pieceId: pi.piece.id,
                                libelle: pi.piece.libelle,
                                avis: pi.observation,
                                caracteristiques:
                                    pi.piece.caracteristiques
|| [],
                                requis:
                                    pi.requisExterne ||
                                    pi.requisInterne ||
                                    pi.requisHandicap,
                                file: null
                            })
                        );

                    console.log("Default piece",
this.currentCorps.pieces);


                    if (this.othersPiece.length > 0) {
                        console.log("Others piece",
this.othersPiece);

                        this.othersPiece.forEach(
                            (piece) => {
```

```
                                        if
(this.documentTypes.findIndex(d => d.pieceId == piece.piece.id) == -1)
{
                                                this.documentTypes.push({
                                                    pieceId:
piece.piece.id,
                                                    libelle:
piece.piece.libelle,
                                                    avis:
piece.observation,
                                                    caracteristiques:

piece.piece.caracteristiques || [],
                                                    requis:

piece.requiredForExternal ||

piece.requiredForInternal ||

piece.requiredForHandicap,
                                                    file: null
                                                })
                                            }
                                        }
                                    )
                                }

                                this.selectData['diplomes'] =
trueDiplome.map(
                                    (item: any) => {
                                        return {
                                            value: item.id,
                                            label: item.diplome
                                                ? item.diplome.libelle
                                                : 'Diplôme [non défini]'
                                        };
                                    }
                                );
                                // const pieceDiplome = {
                                //     pieceId: 0,
                                //     libelle: null,
                                //     avis: null,
                                //     caracteristiques: [],
```

```
                                //     requis: true,
                                //     options: true,
                                //     file: null
                                // }

                                // this.documentTypes.push(pieceDiplome)

                                this.initForm();
                                this.initFormForCandidatureDocs(
                                    this.candidature?.pieces
                                );
                                this.initLieuDepot();
                                this.initCentreComposition();
                                this.initEpreuves();
                                this.loading = false;
                            },
                            error: (err) => {
                                console.error(
                                    'Error fetching concours details:',
                                    err
                                );
                            }
                        });
                    this.initTresorPay();
                },
                error: (err) => {
                    console.error('Error fetching concours details:', err);
                }
            });
    }

    shouldSubmit(): boolean {
        return ['CD1E', 'CDN',
'CPA'].includes(this.candidature?.etape?.code);
    }

    initForm() {
        // Charger d'abord les types de handicap
        this.typeHandicapService.all$(this.pageOptions).subscribe({
            next: (response) => {
                this.selectData['typeHandicap'] =
response.content.map((item) => ({
                    value: item.id,
```

```
                    label: item.libelle
                }));

                // Initialiser le formulaire APRÈS le chargement des
données
                this.initializeForm();
            },
            error: (err) => {
                console.error('Erreur lors du chargement des types de
handicap :', err);
                this.selectData['typeHandicap'] = [];
                // Initialiser quand même le formulaire
                this.initializeForm();
            }
        });
    }

    private initializeForm() {
        console.log("Candidat :", this.candidature?.candidat);
        console.log("Candidature Type handicap :",
this.candidature?.candidat?.typeHandicap?.id);

        const typeHandicapValue =
this.candidature?.candidat?.typeHandicap?.id || '';

        this.form = this.fb.group({
            // IDENTIFICATION DU CANDIDAT
            nom: [
                this.candidature?.candidat?.lastName || '',
                Validators.required
            ],
            prenoms: [
                this.candidature?.candidat?.firstName || '',
                Validators.required
            ],
            sexe: [
                this.candidature?.candidat?.gender || 'M',
                Validators.required
            ],
            handicap: [
                this.candidature?.candidat?.handicap || 'non',
                Validators.required
            ],
```

```
            typeHandicap: [typeHandicapValue], // Valeur définie ici
            dateNaissance: [
                this.candidature?.candidat?.birthDate || '',
                Validators.required
            ],
            lieuNaissance: [
                this.candidature?.candidat?.birthPlace || '',
                Validators.required
            ],
            adresse: [
                this.candidature?.candidat?.adresse || '',
                Validators.required
            ],
            telephone: [
                this.candidature?.candidat?.phone || '',
                Validators.required
            ]
        });

        // Forcer la détection des changements si nécessaire
        if (typeHandicapValue) {
            setTimeout(() => {
this.form.get('typeHandicap').setValue(typeHandicapValue);
            }, 100);
        }

        const selectedDepotCode = this.candidature?.lieuDepot?.id ||
'';
        const selectedCommuneCode = this.candidature?.commune?.code ||
'';
        const selectedEpreuve = this.candidature?.epreuveFacultative ||
0;

        this.formDepot = this.fb.group({
            lieuDepot: [selectedDepotCode, Validators.required]
        });

        this.formCommune = this.fb.group({
            commune: [selectedCommuneCode, Validators.required]
        });

        this.formEpreuve = this.fb.group({
```

```
            epreuveFacultative: [
                selectedEpreuve ? selectedEpreuve : null,
                Validators.required
            ]
        });
    }


    initFormForCandidatureDocs(candidatureDocs: PieceSoumise[]): void {
        let docs = candidatureDocs || [];
        let newDocs = []


        if (this.candidature?.candidat?.handicap == 'non') {
            newDocs = docs.filter(doc => {
                const piece = this.othersPiece.find(pi => pi.piece.id
=== doc.piece.id);
                if (piece) {
                    return piece.requiredForExternal ||
piece.requiredForInternal;
                }
                return false;
            });

            this.currentCorps.pieces.forEach((pi) => {
                const exists = docs.find(doc => doc.piece.id ===
pi.piece.id);
                if (exists) {
                    newDocs.push(exists)
                }
            })

            console.log("NEW DOC", newDocs);


        }
        if (this.candidature?.candidat?.handicap == 'oui') {
            newDocs = docs.filter(doc => {
                const piece = this.othersPiece.find(pi => pi.piece.id
=== doc.piece.id);
                if (piece) {
```

```
                    return (piece.requiredForExternal ||
piece.requiredForInternal || piece.requiredForExternal ||
piece.requiredForInternal) && piece.requiredForHandicap;
                }
                return false;
            });

            console.log("new doc", newDocs);

            this.currentCorps.pieces.forEach((pi) => {
                if (pi.requisHandicap) {
                    const exists = docs.find(doc => doc.piece.id ===
pi.piece.id);
                    if (exists) {
                        newDocs.push(exists)
                    }
                }
            })
        }

        // docs =  ? docs.filter(doc =>
this.othersPiece.includes(doc.piece.id) && ) : docs;

        const documentsFormArray = this.fb.array(
            this.documentTypes
                .map((docType) => {
                    const existingDoc = newDocs.find(
                        (d) => d.piece.id === docType.pieceId
                    );

                    console.log("existingDoc", existingDoc);

                    const caracteristiqueControls =
this.buildCaracteristiqueControls(
                        docType.caracteristiques,
                        existingDoc?.meta || []
                    );

                    return this.fb.group({
                        naturePiece: [docType.pieceId],
                        libelle: [docType.libelle],
                        requis: [!!docType.requis],
                        ...caracteristiqueControls,
```

```
                        fichier: [
                            existingDoc?.fichierId ?? null,
                            docType.requis ? Validators.required : [],
                        ],
                        avis: [existingDoc?.motifPieces || []],
                    });
                })
        );

        this.form1 = this.fb.group({
            documents: documentsFormArray,
        });
    }

    private buildCaracteristiqueControls(
        caracteristiques: any[],
        meta: any[]
    ): Record<string, FormControl> {
        return caracteristiques.reduce((controls, carac) => {
            const cr = caracteristiques.find(c => c.code ===
carac.code);
            const existingMeta = meta.find((m) => m.caracteristiqueUuid
=== cr.uuid);

            let valeur: any = '';
            if (existingMeta) {
                switch (existingMeta.type) {
                    case 'string':
                        valeur = existingMeta.valeur ?? '';
                        break;
                    case 'integer':
                        valeur = existingMeta.valeurNumeric ?? '';
                        break;
                    case 'date':
                        valeur = existingMeta.valeurDate ?? '';
                        break;
                    default:
                        valeur = '';
                }
            } else {
                // Génération par défaut (au besoin)
                valeur = generateUUIDv4().substring(0,
8).toUpperCase();
```

```
            }

            controls[carac.code] = new FormControl(
                valeur,
                carac.requis ? Validators.required : []
            );
            return controls;
        }, {} as Record<string, FormControl>);
    }


    initTresorPay(): void {
        if (typeof TresorPay !== 'undefined') {
            TresorPay.init('#pay-btn', {
                public_key: 'pk_live_ZVmuUtknwrYiCpAFUiisfmNi',
                transaction: {
                    amount: 1,
                    description: 'Payer droit de candidature',
                    custom_metadata: {
                        equittance: {
                            receipe_nature: 'national',
                            receipe_class: 1130,
                            ts_account_number:
'BJ010010000001234567890S2'
                        }
                    }
                },
                // transaction: {
                //   amount: 1,
                //   description: 'Payer droit de candidature',
                // },
                customer: {
                    email: 'akowedarius@gmail.com',
                    lastname: 'Darius'
                }
            });
            // console.log('TresorPay initialisé avec succès');
        } else {
            console.error('TresorPay non chargé');
        }
    }

    initLieuDepot() {
```

```
        this.plangeoRecrutementService
            .listLieuDepot$(1, '', this.candidature?.recrutement?.uuid)
            .subscribe({
                next: (data) => {
                    this.lieuxDepot = data.content.map((data0) => ({
                        code: data0.id,
                        libelle: data0.detailUa.libelle
                    }));
                },
                error: (err) => {
                    console.error(
                        'Erreur lors de la récupération des lieux de
dépôt:',
                        err
                    );
                }
            });
    }

    initEpreuves() {
        this.selectData['epreuves'] = this.currentCorps.epreuves
            .filter((e) => !e.facultatif)
            .map((e) => ({
                label: e.epreuveRecrutement.libelle, // ou e.id si tu
as un id unique
                value: e.id
            }));
    }

    initCentreComposition() {
        this.centreCompositionService
            .listCentreComposition$(1, '',
this.candidature?.recrutement?.uuid)
            .subscribe({
                next: (data) => {
                    data.content.forEach((data0) => {
                        const code = data0.centre.planGeo.id;
                        const libelle = data0.centre.planGeo.libelle;
                        const existe = this.centreComposition.some(
                            (item) => item.code === code
                        );

                        if (!existe) {
```

```
                                this.centreComposition.push({
                                    code,
                                    libelle
                                });
                                // centre: data0.centre.id

                            }
                        });
                    },
                    error: (err) => {
                        console.error(
                            'Erreur lors de la récupération des centres de
compositions:',
                            err
                        );
                    }
                });
    }
    onCheckboxChange(event: any, epreuve: any) {
        const epreuvesArray = this.form.get(
            'epreuvesSelectionnees'
        ) as FormArray;

        if (event.target.checked) {
            epreuvesArray.push(this.fb.control(epreuve.id));
        } else {
            const index = epreuvesArray.controls.findIndex(
                (x) => x.value === epreuve.id
            );
            if (index !== -1) {
                epreuvesArray.removeAt(index);
            }
        }
    }
    openPieceModal(index: number) {
        this.editingPieceIndex = index;
        const docControl = this.documentFormGroups[index];
        const piece: any = this.documentTypes.find(
            (p) => p.pieceId === docControl.get('naturePiece')?.value
        );

        this.pieceCourante = piece;
```

```
        const group: any = {};

        (piece.caracteristiques || []).forEach((carac) => {
            // const toDateInputFormat = (value: string): string => {
            //   const date = new Date(value);
            //   return isNaN(date.getTime()) ? '' :
date.toISOString().split('T')[0];
            // };
            // console.log('Doc control', docControl);

            const valeur = docControl.get(carac.code)?.value || '';

            group[carac.code] = [valeur, []];
            this.valeursExistantes[carac.code] = [valeur, []];
        });

        group['fichier'] = [
            docControl.get('fichier')?.value || null,
            Validators.required
        ];
        this.valeursExistantes['fichier'] = [
            docControl.get('fichier')?.value || null,
            Validators.required
        ];
        this.pieceCaracteristiqueForm = this.fb.group(group);
        this.showPieceModal = true;
    }
    onSelectOptionnelles(event: Select2UpdateEvent): void {
        const selected = event.value;

        this.selectedOptionnel = selected;

        // Mise à jour du FormControl correctement

this.formEpreuve.get('epreuvesFacultatives')?.setValue(selected);

        // console.log('Épreuves :', this.currentCorps.epreuves);
        // console.log('Optionnelles choisies :',
this.selectedOptionnel);
    }

    onSelectDiplome(event: Select2UpdateEvent, index: number) {
        this.form1.value.documents[index].setValue(event.value);
```

```
    }
    closePieceModal() {
        this.showPieceModal = false;
        this.editingPieceIndex = null;
        this.pieceCaracteristiqueForm = null;
    }

    handlePieceFileInput(event: Event) {
        const input = event.target as HTMLInputElement;
        if (input.files && input.files.length > 0) {
            const file = input.files[0];
            this.pieceCaracteristiqueForm.patchValue({ fichier: file
});
        }
    }

    handleDochandicapFileInput(event: Event) {
        const input = event.target as HTMLInputElement;

        if (input.files && input.files.length > 0) {
            const file = input.files[0];

            // On met à jour uniquement le champ 'fichier' du
formulaire
            this.form.patchValue({
                fichier: file
            });

            // Et on marque le champ comme "touché" pour la validation
            this.form.get('fichier')?.markAsDirty();
        }
    }



    onPieceSave(form: any) {
        if (form && this.editingPieceIndex !== null) {
            const resultOld = {
                caracteristiques: Object.entries(form)
                    .filter(([key]) => key !== 'fichier')
                    .map(([uuid, valeur]) => ({ uuid, valeur })),
                fichier: form.fichier
            };
```

```
            console.log("Form ...", form)

            let result = { fichier: form.fichier };
            Object.entries(form)
                .filter(([key]) => key !== 'fichier')
                .map(([code, valeur]) => ({ code, valeur }))
                .forEach((e) => {
                    result = { ...result, [e.code]: e.valeur };
                });



            const docControl =
this.documentFormGroups[this.editingPieceIndex];
            Object.keys(result).forEach((key) => {
                docControl.get(key)?.setValue(result[key]);
            });

            console.log("Form piece sauvegardé :", result);
            console.log("Doc true", this.documentFormGroups);


            this.showPieceModal = false;
        }
    }
    // onRadioChange(epreuveId: number) {
    //     // Pour les radios, remplacer tout le tableau par un seul
élément
    //     this.formEpreuve.patchValue({
    //         epreuveFacultative: [epreuveId] // Tableau avec un seul
élément
    //     });
    // }
    checkedEpreuveFacultative(id: number): boolean {
        return this.formEpreuve.get('epreuveFacultative')?.value ===
id;
    }

    // méthode appelée quand on clique sur un toggle
    onToggleFacultative(id: number, event: Event) {
        const input = event.target as HTMLInputElement;
        const ctrl = this.formEpreuve.get('epreuveFacultative');
```

```
        if (!ctrl) return;

        if (input.checked) {
            // sélection exclusive → on remplace par l'id
            ctrl.setValue(id);
        } else {
            // si on désactive le toggle déjà sélectionné → valeur à
null
            if (ctrl.value === id) {
                ctrl.setValue(null);
            }
        }
    }

    get documents(): FormArray {
        return this.form1.get('documents') as FormArray;
    }

    get progression(): number {
        const formValid = this.form?.valid ?? false;
        const docsCompleted = this.completionPercentage >= 100;
        const depotValid = this.formDepot?.valid ?? false;
        const communeValid = this.formCommune?.valid ?? false;
        const epreuveValid =
            this.formEpreuve?.valid || this.epreuvesFacultatif?.length
=== 0;

        let score = 0;
        if (formValid) score += 1;
        if (docsCompleted) score += 1;
        if (depotValid) score += 1;
        if (communeValid) score += 1;
        if (epreuveValid) score += 1;

        return Math.round((score / 5) * 100);
    }

    get documentFormGroups(): FormGroup[] {
        return this.documents.controls as FormGroup[];
    }

    get progressionMessage(): string {
        const pct = this.progression;
```

```
        if (pct < 100) return `Progression partielle (${pct}%)`;
        return 'Progression complète';
    }

    get isFormLocked(): boolean {
        return this.candidature?.etape?.code == 'CEN' ||
this.progression > 100;
    }


    handleFileInput(event: Event, index: number): void {
        const input = event.target as HTMLInputElement;
        const fichierControl = this.documents.at(index).get('fichier');

        if (input.files && input.files.length > 0) {
            const file = input.files[0];
            const tailleMax = 5 * 1024 * 1024; // 5 Mo en octets

            // --- 1. Vérification du Type de Fichier (PDF ou Image)
---
            const typesAutorises = [
                'application/pdf',
                'image/jpeg',
                'image/png'
            ];

            if (!typesAutorises.includes(file.type)) {
                errorToaster(
                    'Seuls les fichiers PDF et les images (JPEG, PNG,
GIF, WEBP) sont autorisés.'
                );
                fichierControl?.setValue(null);
                input.value = ''; // Réinitialiser le champ
                return; // Sortir de la fonction
            }
            // ----------------------------------------------------------

            // --- 2. Vérification de la Taille (5 Mo) ---
            if (file.size > tailleMax) {
                errorToaster(
                    'Le fichier dépasse la taille maximale autorisée de
5 Mo.'
                );
```

```
                    fichierControl?.setValue(null);
                    input.value = ''; // Réinitialiser le champ
                } else {
                    fichierControl?.setValue(file);
                }
                // ----------------------------------------


            } else {
                fichierControl?.setValue(null);
            }

            fichierControl?.markAsTouched();
        }

        isTabValid(index: number): boolean {
            switch (index) {
                case 0:
                    return this.form?.valid;
                case 1:
                    const requiredDocs = this.documentFormGroups.filter(
                        (docGroup) => docGroup.get('requis')?.value
                    );

                    // Si aucune pièce n'est requise, l'onglet est
considéré comme valide par défaut.
                    if (requiredDocs.length === 0) {
                        return false; // <--- C'est cette ligne qui fait
que l'icône sera un CHECKMARK si 0 pièces requises.
                    }

                    // Sinon, vérifiez si TOUS les FormGroups des documents
requis sont valides.
                    const allRequiredDocumentsAreValid =
requiredDocs.every(
                        (docGroup) => docGroup.valid
                    );

                    return allRequiredDocumentsAreValid;
                case 2:
                    return this.formDepot?.valid;
                case 3:
                    return this.formCommune?.valid;
                case 4:
```

```
                return this.formEpreuve?.valid ||
this.epreuvesFacultatif.length === 0;
            default:
                return false;
        }
    }

    get completionPercentage(): number {
        const requiredPiecesControls = this.documents.controls.filter(
            (c: FormGroup) => c.get('requis')?.value // Filtre pour
obtenir seulement les pièces qui sont marquées comme 'requises'
        );
        const total = requiredPiecesControls.length;

        // Compte le nombre de pièces requises dont le FormGroup est
entièrement valide
        const completedRequiredPieces = requiredPiecesControls.filter(
            (c: FormGroup) => c.valid // Vérifie la validité du
FormGroup entier de la pièce
        ).length;

        // Si aucune pièce n'est requise, le pourcentage est 100%
        return total === 0
            ? 100
            : Math.round((completedRequiredPieces / total) * 100);
    }
    resetFileInput(input: HTMLInputElement) {
        input.value = '';
    }

    get uniqueUploadedTypesCount(): number {
        return this.documents.controls.filter(
            (c) => c.get('requis')?.value && c.get('fichier')?.value
        ).length;
    }

    /**
     * Submit the first part of the candidature (personal information)
     * This method is called when the user submits the first form of
the candidature.
     * It checks if the form is valid and not locked, then sends the
data to the server.
```

```
     * If the submission is successful, it shows a success message and
reloads the page
     * to reflect the changes. If the submission fails, it shows an
error message.
     * If the form is invalid, it shows an error message prompting the
user to fill in
     * the required fields.
     * @returns {void}
     * @memberof DetailsCandidatureComponent
     * @description This method handles the submission of the first
part of the candidature.
     */
    async submitCandidature0() {
        if (this.form.valid && !this.isFormLocked) {
            this.loadingSubmission = true;
            const uuid = this.route.snapshot.paramMap.get('id');
            const data = this.form.value;

            const formData = new FormData();



            formData.append('nom', this.form.value.nom);
            formData.append('prenoms', this.form.value.prenoms);
            formData.append('sexe', this.form.value.sexe);
            formData.append('handicap', this.form.value.handicap);
            formData.append('typeHandicap',
this.form.value.typeHandicap && this.form.value.handicap == 'oui' ?
this.form.value.typeHandicap : '99');
            formData.append('dateNaissance',
this.form.value.dateNaissance);
            formData.append('lieuNaissance',
this.form.value.lieuNaissance);
            formData.append('adresse', this.form.value.adresse);
            formData.append('telephone', this.form.value.telephone);
            formData.append('naturePiece', '15')

            this.loading = true
            this.service.editerCandidatCandidature(uuid,
formData).subscribe({
                next: (data) => {
                    if (data) {
                        successToaster(
```

```
                                'Candidature éditée avec succès : Informations personnelles enregistrées'
                            );
                            this.loadingSubmission = false;
                            this.loading = false
                            this.refreshData(uuid);
                        } else {
                            errorToaster('Vous avez déjà postulé à ce concours');
                            this.loadingSubmission = false;
                            this.loading = false
                        }
                    },
                    error: (err) => {
                        console.error('Error fetching concours details:', err);
                        this.loadingSubmission = false;
                        this.loading = false
                    }
                });
            } else {
                errorToaster('Veuillez renseigner la référence des pièces');
                this.loadingSubmission = false;
            }
        }

        /**
         * Submit the second part of the candidature (documents)
         * This method is called when the user submits the second form of the candidature.
         * It checks if the form is valid and not locked, then processes the documents
         * and sends them to the server. If the submission is successful, it shows a success
         * message and reloads the page to reflect the changes. If the submission fails,
         * it shows an error message. If the form is invalid, it shows an error message
         * prompting the user to fill in the required fields.
         * @returns {Promise<void>}
         * @memberof DetailsCandidatureComponent
```

```
     * @description This method handles the submission of the second
part of the candidature.
     */
    async submitCandidature1(): Promise<void> {
        if (this.form1.valid && !this.isFormLocked) {
            this.loadingSubmission = true;
            this.loading = true;
            const uuid = this.route.snapshot.paramMap.get('id');
            const data = this.form1.value.documents;

            console.log("Data ...", data)

            const formData = new FormData();

            if (data.length > 0) {
                const appendTasks = data.map(async (doc: any, index:
number) => {
                    const docType = this.documentTypes.find(
                        (d) => d.pieceId === doc.naturePiece
                    );
                    const caracteristiques = (docType?.caracteristiques
|| []).map(
                        (carac: any) => ({
                            uuid: carac.uuid,
                            valeur:
                                doc[carac.code] != null
                                    ? String(doc[carac.code])
                                    : ''
                        })
                    );

                    console.log(`Ajout de la pièce ${index}:`, doc,
caracteristiques);

                    formData.append(
                        `pieces[${index}].naturePiece`,
                        doc.naturePiece
                    );
                    formData.append(`pieces[${index}].libelle`,
doc.libelle);
                    formData.append(`pieces[${index}].requis`,
doc.requis);
```

```
                    caracteristiques.forEach((carac, i) => {
                        formData.append(
`pieces[${index}].caracteristiques[${i}].uuid`,
                            carac.uuid || ''
                        );
                        formData.append(
`pieces[${index}].caracteristiques[${i}].valeur`,
                            carac.valeur || ''
                        );
                    });

                    // Fichier
                    if (doc.fichier) {
                        if (typeof doc.fichier === 'string') {
                            const blob = await
this.loadFileAsBlob(doc.fichier);
                            const file = new File(
                                [blob],
`${doc.libelle}.${getExtension(blob.type)}`,
                                { type: blob.type }
                            );
                            formData.append(`pieces[${index}].fichier`,
file);
                        } else {
                            formData.append(
                                `pieces[${index}].fichier`,
                                doc.fichier
                            );
                        }
                    }
                });

                try {
                    await Promise.all(appendTasks);
                    this.loading = true
                    this.service
                        .editerDocumentsCandidature(uuid, formData)
                        .subscribe({
                            next: (data) => {
                                if (data) {
```

```
                successToaster(
                    'Candidature éditée avec succès : Documents enregistrés'
                );
                this.refreshData(uuid);
            } else {
                errorToaster(
                    'Vous avez déjà postulé à ce concours'
                );
            }
            this.loadingSubmission = false;
            this.loading = false;
        },
        error: (err) => {
            console.error(
                'Erreur lors de la soumission :',
                err
            );
            errorToaster(
                'Une erreur est survenue lors de la soumission'
            );
            this.loadingSubmission = false;
            this.loading = false;

        }
    });
} catch (error) {
    errorToaster('Erreur lors du chargement d'un fichier distant');
    console.error(error);
    this.loadingSubmission = false;
    this.loading = false;
}
}
else {
    errorToaster('Soumission impossible : Aucune pièce à soumettre');
}

} else {
    errorToaster('Ce formulaire ne peut pas être soumis');
```

```
            this.form1.markAllAsTouched();
        }
    }

    /**
     * Submit the third part of the candidature (lieu de dépôt)
     * This method is called when the user submits the third form of
the candidature.
     * It checks if the form is valid and not locked, then sends the
data to the server.
     * If the submission is successful, it shows a success message and
reloads the page
     * to reflect the changes. If the submission fails, it shows an
error message.
     * If the form is invalid, it shows an error message prompting the
user to fill in
     * the required fields.
     * @returns {Promise<void>}
     * @memberof DetailsCandidatureComponent
     */
    async submitCandidature2(): Promise<void> {
        if (this.formDepot.valid && !this.isFormLocked) {
            this.loadingSubmission = true;
            this.loading = true;
            const uuid = this.route.snapshot.paramMap.get('id');
            const lieuDepot = this.formDepot.value.lieuDepot;

            const data = {
                lieuDepot
            };

            this.service.editerLieuDepotCandidature(uuid,
data).subscribe({
                next: (data) => {
                    if (data) {
                        successToaster(
                            'Candidature éditée avec succès : Lieu de
dépôt enregistrées'
                        );
                        this.refreshData(uuid);
                        this.loadingSubmission = false;
                        this.loading = false;
                    } else {
```

```
                        errorToaster('Vous avez déjà postulé à ce
concours');
                        this.loadingSubmission = false;
                        this.loading = false;
                    }
                },
                error: (err) => {
                    console.error('Error fetching concours details:',
err);
                    this.loadingSubmission = false;
                    this.loading = false;
                }
            });
        } else {
            errorToaster('Veuillez renseigner la référence des
pièces');
            this.form1.markAllAsTouched();
        }
    }

    async submitCandidature3(): Promise<void> {
        if (this.formCommune.valid && !this.isFormLocked) {
            this.loadingSubmission = true;
            this.loading = true;
            const uuid = this.route.snapshot.paramMap.get('id');
            const commune = this.formCommune.value.commune;
            const centre = this.centreComposition.find(
                (c) => c.code === commune
            )?.centre;

            const data = {
                commune,
                centre
            };

            this.service
                .editerCommuneCompositionCandidature(uuid, data)
                .subscribe({
                    next: (data) => {
                        if (data) {
                            successToaster(
                                'Candidature éditée avec succès :
Commune de composition enregistrée'
```

```
                                );
                                this.refreshData(uuid);
                                this.loadingSubmission = false;
                                this.loading = false;

                            } else {
                                errorToaster(
                                    'Vous avez déjà postulé à ce concours'
                                );
                                this.loadingSubmission = false;
                                this.loading = false;
                            }
                        },
                        error: (err) => {
                            console.error('Error fetching concours
details:', err);
                            this.loadingSubmission = false;
                            this.loading = false;
                        }
                    });
            } else {
                errorToaster('Veuillez renseigner la référence des
pièces');
                this.form1.markAllAsTouched();
            }
        }

    async submitCandidature4(): Promise<void> {
            if (this.formEpreuve.valid && !this.isFormLocked) {
                this.loadingSubmission = true;
                this.loading = true;
                const uuid = this.route.snapshot.paramMap.get('id');
                const selectedFacultativeId =
                    this.formEpreuve.value.epreuveFacultative;

                const epreuves = []
                let epreuveFacultativeChoisie = 0
                this.epreuves.forEach(
                    epre => {
                        epreuves.push(epre.id)
                    }
                )
                epreuves.push(selectedFacultativeId)
```

```
            epreuveFacultativeChoisie = selectedFacultativeId

            const data = {
                epreuves,
                epreuveFacultativeChoisie
            };

            this.service.editerEpreuveCandidature(uuid,
data).subscribe({
                next: (data) => {
                    if (data) {
                        successToaster(
                            'Candidature éditée avec succès : Epreuves
choisies enregistrée'
                        );
                        this.refreshData(uuid);
                        this.loadingSubmission = false;
                        this.loading = false;
                    } else {
                        errorToaster('Vous avez déjà postulé à ce
concours');
                        this.loadingSubmission = false;
                        this.loading = false;
                    }
                },
                error: (err) => {
                    console.error('Error fetching concours details:',
err);
                    this.loadingSubmission = false;
                }
            });
        } else {
            errorToaster('Veuillez choisir une épreuve facultative');
            this.form1.markAllAsTouched();
        }
    }

    previewFile(file: string | File): void {
        if (!file) return;

        // Si c'est une URL (fichier du serveur déjà uploadé)
        if (typeof file === 'string') {
            this.filePreviewService.preview(file);
```

```
        }
        if (file instanceof File) {
            this.filePreviewService.previewFromBlob(file);
        }
    }

    private loadFileAsBlob(fileUUID: string): Promise<Blob> {
        return new Promise((resolve, reject) => {
            this.storageService.findFile(fileUUID).subscribe({
                next: (response: Blob) => resolve(response),
                error: (err) => reject(err)
            });
        });
    }

    payerCandidature() {
        const transaction = {
            public_key: environment.publicTresorPayApiKey,
            transaction: {
                amount: `${this.candidature?.montantQuittance || 5}`,
                description: `Paiement candidature
#${this.candidature?.uuid}`,
                custom_metadata: {
                    ...environment.tresorPayMetaData,
                    candidatureId: this.candidature?.uuid
                }
            },
            customer: {
                email: this.candidature?.candidat?.email,
                lastname: this.candidature?.candidat?.lastName,
                firstname: this.candidature?.candidat?.firstName,
                phone_number: {
                    country: 'BJ',
                    number: this.candidature?.candidat?.phone
                }
            },
            currency: {
                iso: 'XOF'
            },
            onComplete: this.onCompleteTransaction
        };
        console.log(transaction);
```

```
        this.isLoading = true;
        const widget = window.TresorPay.init(transaction);
        widget.open();
        this.isLoading = false;
    }

    onCompleteTransaction = (response: any) => {
        const trx = response.transaction;
        const info = {
            idTrx: trx.id,
            refTrx: trx.reference,
            refQuittance: ''
        };
        let link = '';
        if (response.reason === window.TresorPay.DIALOG_DISMISSED) {
            this.paymentTransactionStatus = 'aborted';
            this.htmlMessage = `<p>Widget fermé par le client. Cliquez
le bouton de paiement pour relancer si besoin</p>`;
            this.onTogglePaymentReportModal();
            this.afterPaymentComplete(info);
            return;
        }
        switch (trx.status) {
            case 'approved':
                link =
`${environment.eQuittanceApiUrl}/tresorpay/?receipt=${trx.reference}`;
                this.paymentTransactionStatus = 'approved';
                this.htmlMessage = `
                    <p style='text-align: center'>Référence : <b>
${trx.id} (${trx.reference})</b></p>
                    Le paiement a été efectué avec succès. Une copie de
la quittance a été envoyée dans votre boite e-mail
                    <b>${trx.customer.email}</b> <br>NB : <span
style='color: red;' >N'oubliez pas de vérifier également vos
spams</span>.
                    <br>Si le message n'est pas délivré et que vous
êtes quand même débité, utilisez le module de réclamation en bas de la
page.<br/> 
                    <p><a target="_blank" href="${link}"
style="margin-bottom:20px;" class="btn btn-success bg-[#0054A0] px-4
py-2 rounded-2xl mb-8 text-white font-semibold">Télécharger la
quittance.</a></p>
                `;
```

```
                this.onTogglePaymentReportModal();
                this.afterPaymentComplete(info);
                break;

            case 'declined':
            case 'canceled':
                this.paymentTransactionStatus = 'canceled';
                this.htmlMessage = `<p>La transaction a été annulée ou
n'a pas abouti. Merci de réessayer.</p> <br/><br/> <p><a href=""
style="margin-bottom:20px;" class="btn btn-success bg-[#0054A0] px-4
py-2 rounded-2xl mb-8 text-white font-semibold">D'accord</a></p>`;
                this.onTogglePaymentReportModal();
                break;
        }
    };

    private afterPaymentComplete(trx: any) {
        this.service.payerCandidature(this.candidature?.uuid,
trx).subscribe({
            next: (response: any) => {
                successToaster(response.message);
                this.refreshData(this.candidature?.uuid);
            },
            error: (err: ErrorType) => {
                errorMessage('Erreur!', err.errorMessage);
            }
        });
    }

    envoyerCandidature() {
        const uuid = this.route.snapshot.paramMap.get('id');
        const ids = [uuid];
        const message =
            ids.length <= 1
                ? 'Voulez-vous envoyer votre candidature ? Une fois
envoyée, vous ne pourrez plus la modifier.'
                : 'Voulez-vous supprimer en masse toutes les lignes
sélectionnées ?';
        Swal.fire({
            title: 'Confirmation',
            text: message,
            icon: 'question',
            backdrop: true,
```

```
            showCancelButton: true,
            confirmButtonColor: '#2563eb',
            cancelButtonColor: '#d33',
            confirmButtonText: 'Oui, envoyer',
            cancelButtonText: 'Annuler',
            allowEscapeKey: false,
            allowOutsideClick: false
        }).then((result) => {
            if (result.isDismissed) return;
            this.service.envoyerCandidature(ids[0]).subscribe({
                next: (response: any) => {
                    successToaster(response.message);
                    this.refreshData(uuid);
                },
                error: (err: ErrorType) => {
                    errorMessage('Erreur!', err.errorMessage);
                }
            });
        });
    }

    // Remplace ta méthode refreshData(uuid) actuelle par celle-ci :
    refreshData(uuid: string): void {
        this.loading = true;
        this.service.detailsCandidature(uuid).subscribe({
            next: (data: any) => {
                this.candidature = data;
                // Ré-initialise tes formulaires avec les nouvelles
données si nécessaire
                this.patchForms(data);
                this.loading = false;
                this.loadingSubmission = false;
            },
            error: (err) => {
                this.loading = false;
                this.loadingSubmission = false;
                console.error('Erreur lors du rafraîchissement', err);
            }
        });
    }

    private patchForms(data: any): void {
        if (!data) return;
```

```
        // 1. Mise à jour des infos personnelles
        if (this.form && data.candidat) {
            this.form.patchValue({
                nom: data.candidat.lastName,
                prenoms: data.candidat.firstName,
                sexe: data.candidat.sexe,
                handicap: data.candidat.handicap,
                typeHandicap: data.candidat.typeHandicap,
                dateNaissance: data.candidat.dateNaissance,
                lieuNaissance: data.candidat.lieuNaissance,
                adresse: data.candidat.adresse,
                telephone: data.candidat.phone
            });
        }

        // 2. Mise à jour du lieu de dépôt
        if (this.formDepot) {
            this.formDepot.patchValue({
                lieuDepot: data.lieuDepot
            });
        }

        // 3. Mise à jour de la commune
        if (this.formCommune) {
            this.formCommune.patchValue({
                commune: data.commune
            });
        }

        // 4. Mise à jour des documents (Important pour la progression)
        if (data.pieces && data.pieces.length > 0) {
            // Ici, on met à jour le FormArray 'documents'
            // Note : Selon ta logique, tu as peut-être besoin de
reconstruire l'array
            // ou simplement de patcher les valeurs existantes.
            this.documents.patchValue(data.pieces);
        }

        // 5. Mise à jour de l'épreuve facultative
        if (this.formEpreuve) {
            this.formEpreuve.patchValue({
                epreuveFacultative: data.epreuveFacultativeChoisie
```

```
            });
        }
    }

    onTogglePaymentReportModal() {
        this.showPaymentReportModal = !this.showPaymentReportModal;
    }

    downloadReceipt(candidature: FullCandidature) {
        const link =
`${environment.eQuittanceApiUrl}/tresorpay/?receipt=${candidature?.tran
sactionReference}`;
        window.open(link, '_blank');
    }
}
```

```
@if (loading) {
<!-- SECTION: ÉCRAN DE CHARGEMENT MINIMALISTE -->
<div
    class="fixed inset-0 bg-white/80 backdrop-blur-sm flex flex-col
items-center justify-center z-[100] transition-opacity duration-300">
    <div class="w-12 h-12 border-2 border-gray-300 border-t-gray-800
rounded-full animate-spin"></div>
    <p class="mt-4 text-sm font-medium text-gray-600">
        Traitement en cours...
    </p>
</div>
} @else {
<div class="bg-slate-50 rounded-2xl overflow-hidden min-h-screen">
    <!-- LAYOUT PRINCIPAL PLEINE LARGEUR -->
    <div class="flex flex-col lg:flex-row">
        <!--
============================================================ -->
```

```
        <!-- ||          COLONNE DE NAVIGATION LATÉRALE FIXE
|| -->
        <!--
========================================================== -->
        <aside
            class="w-full lg:w-72 xl:w-80 bg-white border-r
border-slate-200 lg:h-screen lg:sticky lg:top-0 shrink-0">
            <div class="p-5 flex flex-col h-full">
                <!-- Logo/Titre -->
                <!-- <div class="pb-5 border-b border-slate-200">
                    <h1 class="text-xl font-bold text-slate-800
tracking-tight">
                        Mon Dossier
                    </h1>
                    <p class="text-xs text-slate-500">
                        Concours Fonction Publique
                    </p>
                </div> -->

                <!-- Carte Candidat -->
                <div class="">
                    <div class="flex items-center gap-4">
                        <div
                            class="w-10 h-10 rounded-full bg-slate-100
flex items-center justify-center text-slate-500 font-bold">
                            {{
                            candidature?.candidat?.firstName
                            | slice: 0 : 1
                            }}{{
                            candidature?.candidat?.lastName | slice: 0
: 1
                            }}
                        </div>
                        <div>
                            <p class="font-semibold text-sm
text-slate-800">
                                {{ candidature?.candidat?.firstName }}
                                {{ candidature?.candidat?.lastName }}
                            </p>
                            <p class="text-xs text-slate-500">
                                Réf: #{{ candidature?.id }} ({{
                                candidature?.code
                                }})
```

```
                                </p>
                            </div>
                        </div>
                    </div>
                    <!-- Actions finales en bas -->
                    <div class="mt-4 pt-6 border-t border-slate-200
space-y-4">
                        <!-- Barre de progression -->
                        <div>
                            <div class="flex justify-between items-center
mb-1">
                                <span class="text-xs font-medium
text-slate-600">{{ progressionMessage }}</span>
                                <span class="text-xs font-semibold
text-green-700">{{ progression }}%</span>
                            </div>
                            <div class="w-full bg-slate-200 rounded-full
h-1.5">
                                <div class="bg-green-600 h-1.5
rounded-full" [style.width.%]="progression"></div>
                            </div>
                        </div>

                        @if (btnSubmitIsVisible) {
                        <button [disabled]="
                                    (candidature?.etape?.code != 'CPA'
&&
                                        candidature?.etape?.code !=
'CD1E') ||
                                    progression < 100
                                " (click)="envoyerCandidature()"
class="btn-action-light btn-submit w-full">
                            <i class="ki-duotone ki-send text-lg"></i>
                            @if(candidature?.etape?.code != 'CD1E'){
                            <span class="text-white">Soumettre ma
candidature</span>
                            }
                            @else(){
                            <span class="text-white">Soumettre à
nouveau</span>
                            }
                        </button>
```

```
                    <button *ngIf="candidature?.etape?.code == 'CDN'"
id="pay-btn" class="btn-action-light btn-pay w-full"
                            [disabled]="loadingPayment"
(click)="payerCandidature()">
                            <ng-container *ngIf="!loadingPayment">
                                <i class="ki-duotone ki-wallet
text-lg"></i>
                                <span>Payer les frais</span>
                            </ng-container>
                            <ng-container *ngIf="loadingPayment">
                                <span class="spinner-border
spinner-border-sm" role="status" aria-hidden="true"></span>
                                <span>Paiement en cours...</span>
                            </ng-container>
                        </button>
                    } @else {
                        <button (click)="downloadReceipt(candidature)"
class="btn-action-light btn-submit w-full">
                            <i class="ki-duotone ki-send text-lg"></i>
                            <span class="text-white">Télécharger la
quittance</span>
                        </button>
                    }
                </div>
                <!-- Menu de navigation -->
                <nav class="flex flex-col space-y-2 mt-8">
                    <button *ngFor="let tab of tabs; let i = index"
(click)="activeTab = i"
                            class="tab-button-light flex justify-between
items-center" [class.active]="activeTab === i">
                            <div class="flex items-center gap-2">
                                <i class="ki-duotone {{
                                        tab.icon || 'ki-right'
                                    }} text-xl"></i>
                                <span class="font-medium text-sm">{{
                                    tab.label
                                }}</span>
                            </div>

                            <ng-container *ngIf="isTabValid(i); else
notValid">
                                <i class="ki-duotone ki-check-circle
text-green-600 text-lg"></i>
```

```
                        </ng-container>
                        <ng-template #notValid>
                            <i class="ki-duotone ki-information-1
text-rose-400 text-lg"></i>
                        </ng-template>
                    </button>
                </nav>

                <!-- Espace flexible pour pousser les éléments vers le
bas -->
                <div class="flex-grow"></div>
            </div>
        </aside>

        <!--
========================================================== -->
        <!-- ||                CONTENU PRINCIPAL (DROITE)
|| -->
        <!--
========================================================== -->
        <main class="w-full p-6 sm:p-8 lg:p-10">
            <!-- En-tête du contenu -->
            <header class="mb-8">
                <h2 class="text-3xl font-bold text-slate-800
tracking-tight">
                    {{ candidature?.recrutement?.libelle }}
                </h2>
                <p class="mt-1 text-base text-slate-800">
                    Corps postulé :
                    <span class="font-medium text-blue-950">
                        {{ currentCorpsLabel }}
                    </span>
                </p>
            </header>

            <!-- SECTION D'ALERTES VISIBLES -->
            <div class="space-y-4 mb-8">
                @if (
                candidature?.etape?.code === 'CR1E' ||
                candidature?.etape?.code === 'CR2E'
                ) {
                <div class="bg-red-50 border border-red-200 rounded-lg
p-4 flex items-start gap-4">
```

```
                    <div class="w-6 h-6 text-red-500 shrink-0 mt-0.5">
                        <i class="ki-duotone ki-wallet text-2xl"><span
class="path1"></span><span
                                class="path2"></span><span
class="path3"></span><span class="path4"></span></i>
                    </div>
                    <div>
                        <h3 class="font-semibold text-red-800">
                            Votre candidature a été rejetée
                        </h3>
                        <div class="text-sm text-yellow-700 mt-1 flex
items-center ">
                            <div class="text-sm">
                                Motif :
                            </div>
                            <div class="text-sm ml-2">
                                {{ candidature?.etape?.observation }}
                            </div>
                        </div>
                        <p class="text-sm text-red-700 mt-1">
                            Désolé, votre candidature n'a pas été
                            retenue.
                        </p>
                    </div>
                </div>
                }

                @if (candidature?.etape?.code === 'CEN') {
                <div class="bg-green-50 border border-green-200
rounded-lg p-4 flex items-start gap-4">
                    <div class="w-6 h-6 text-green-600 shrink-0
mt-0.5">
                        <i class="ki-duotone ki-check-circle text-2xl">
                            <span class="path1"></span><span
class="path2"></span>
                        </i>
                    </div>
                    <div>
                        <h3 class="font-semibold text-green-800">
                            Candidature soumise avec succès
                        </h3>
                        <p class="text-sm text-green-700 mt-1">
                            Votre candidature a été transmise avec
```

```
                succès. Vous ne pouvez plus la modifier.
                </p>
            </div>
        </div>
    }


    @if (candidature?.etape?.code === 'CDVP') {
    <div class="bg-green-50 border border-green-200
rounded-lg p-4 flex items-start gap-4">
            <div class="w-6 h-6 text-green-600 shrink-0
mt-0.5">
                <i class="ki-duotone ki-check-circle text-2xl">
                <span class="path1"></span><span
class="path2"></span>
                </i>
            </div>
            <div>
                <h3 class="font-semibold text-green-800">
                Candidature validé avec succès
                </h3>
                <p class="text-sm text-green-700 mt-1">
                Votre candidature a été validé.Vous êtes
retenues pour la phase de composition.
                </p>
            </div>
        </div>
    }


    @if (candidature?.etape?.code === 'CDVP') {
    <div class="bg-orange-50 border border-orange-200
rounded-lg p-4 flex items-start gap-4">
            <div class="w-6 h-6 text-orange-600 shrink-0
mt-0.5">
                <i class="ki-duotone ki-check-circle text-2xl">
                <span class="path1"></span><span
class="path2"></span>
                </i>
            </div>
            <div>
                <h3 class="font-semibold text-orange-800">
                Paiement non effectué
                </h3>
```

```
                        <p class="text-sm text-orange-700 mt-1">
                            Vous n'avez pas encore payé les frais de
                            dossier pour cette candidature.
                        </p>
                    </div>
                </div>
                }
            </div>

            <!-- CONTENU DES ONGLETS -->
            <div class="bg-white border border-slate-200 rounded-xl
shadow-sm">
                <ng-container [ngSwitch]="activeTab">
                    <!-- ONGLET 0: INFORMATIONS PERSONNELLES -->
                    <div *ngSwitchCase="0" class="p-6 sm:p-8
animate-fadeIn">
                        <form *ngIf="form" [formGroup]="form"
(ngSubmit)="submitCandidature0()" class="space-y-8">
                            <fieldset class="space-y-5">
                                <legend class="form-legend-light">
                                    Identification
                                </legend>
                                <div class="grid grid-cols-1
md:grid-cols-3 gap-5">
                                    <div class="form-group-light">
                                        <label for="nom"
class="form-label-light">Nom de famille *</label>
                                        <input type="text" id="nom"
formControlName="nom" class="form-input-light"
                                            placeholder="Votre nom" />
                                    </div>
                                    <div class="form-group-light">
                                        <label for="prenoms"
class="form-label-light">Prénoms *</label>
                                        <input type="text" id="prenoms"
formControlName="prenoms"
                                            class="form-input-light"
placeholder="Vos prénoms" />
                                    </div>
                                    <div class="form-group-light">
                                        <label
class="form-label-light">Sexe *</label>
```

```
                                        <div class="flex items-center
gap-6 pt-2">
                                            <label
class="radio-label-light"><input type="radio" formControlName="sexe"
                                                value="F"
class="form-radio-light" /><span>Féminin</span></label>
                                            <label
class="radio-label-light"><input type="radio" formControlName="sexe"
                                                value="M"
class="form-radio-light" /><span>Masculin</span></label>
                                        </div>
                                    </div>
                                    <div class="form-group-light">
                                        <label
class="form-label-light">Situation de handicap</label>
                                        <div class="flex items-center
gap-6 pt-2">
                                            <label
class="radio-label-light"><input type="radio"

formControlName="handicap" value="non"

class="form-radio-light" /><span>Non</span></label>
                                            <label
class="radio-label-light"><input type="radio"

formControlName="handicap" value="oui"

class="form-radio-light" /><span>Oui</span></label>
                                        </div>
                                    </div>
                                    <!-- Formulaire personnalisé de
justification -->
                                    <div
*ngIf="form.get('handicap').value === 'oui'" class="form-group-light
w-full">
                                        <label
class="form-label-light">Type de handicap</label>
                                        <div class="flex items-center
gap-6 pt-2 w-full">
                                            <select2
[data]="selectData['typeHandicap']"
```

```
[value]="form.get('typeHandicap').value" formControlName="typeHandicap"

(update)="onSelect($event,'typeHandicap')"

placeholder="Sélectionner un type..." [minCountForSearch]="2"
                                                    class="form-group-light
w-full" />
                                            </div>
                                        </div>
                                    </div>
                                </fieldset>

                                <fieldset class="space-y-5">
                                    <legend class="form-legend-light">
                                        Coordonnées et Naissance
                                    </legend>
                                    <div class="grid grid-cols-1
md:grid-cols-2 gap-5">
                                        <div class="form-group-light">
                                            <label for="dateNaissance"
class="form-label-light">Date de naissance *</label>
                                            <input type="date"
id="dateNaissance" formControlName="dateNaissance"
                                                class="form-input-light" />
                                        </div>
                                        <div class="form-group-light">
                                            <label for="lieuNaissance"
class="form-label-light">Lieu de naissance *</label>
                                            <input type="text"
id="lieuNaissance" formControlName="lieuNaissance"
                                                class="form-input-light"
placeholder="Ex: Cotonou, Bénin" />
                                        </div>
                                        <div class="md:col-span-2
form-group-light">
                                            <label for="adresse"
class="form-label-light">Adresse complète *</label>
                                            <textarea id="adresse"
formControlName="adresse" class="form-input-light"
                                                rows="3"
placeholder="Quartier, rue, ville..."></textarea>
                                        </div>
```

```
                                    <div class="form-group-light">
                                        <label for="telephone"
class="form-label-light">Numéro de téléphone *</label>
                                        <input type="tel"
id="telephone" formControlName="telephone"
                                            class="form-input-light"
placeholder="+229 XX XX XX XX" />
                                    </div>
                                </div>
                            </fieldset>

                            @if (shouldSubmit()) {
                            <div class="form-actions-light">
                                <button type="submit" [disabled]="
                                        !form.valid ||
                                        loadingSubmission ||
                                        isFormLocked
                                    " class="btn-action-light
btn-primary">
                                    @if (loadingSubmission) {
                                    <span class="spinner-border
spinner-border-sm" role="status"
                                        aria-hidden="true"></span>
                                    Enregistrement en cours...
                                    } @else {
                                    Enregistrer vos informations
                                    }
                                </button>
                            </div>
                            }
                        </form>
                    </div>

                    <!-- ONGLET 1: PIÈCES À FOURNIR -->
                    <div *ngSwitchCase="1" class="animate-fadeIn">
                        <form *ngIf="form1" [formGroup]="form1"
(ngSubmit)="submitCandidature1()">
                            <div class="p-6 sm:p-8">
                                <legend class="form-legend-light mb-5">
                                    Pièces du Dossier
                                </legend>
                                <p class="text-sm text-slate-700">
```

```
                                    Veuillez joindre les documents
requis.
                                    Les fichiers acceptés sont PDF,
PNG, JPG
                                    (max 5Mo).
                                </p>
                            </div>

                            <div class="p-8 space-y-3">
                                <h4 class="text-md font-semibold
text-gray-800">Avis sur les pièces</h4>
                                @for (docControl of documentFormGroups;
track $index) {
                                @if (!docControl.get('avis')?.value ) {
                                <div class="bg-green-50 border
border-green-200 rounded-lg p-4 flex items-start gap-4">
                                    <div class="w-6 h-6 text-green-600
shrink-0 mt-0.5">
                                        <i class="ki-duotone
ki-check-circle text-2xl">
                                            <span
class="path1"></span><span class="path2"></span>
                                        </i>
                                    </div>
                                    <div>
                                        <h3 class="font-semibold
text-green-800">
                                            {{
docControl.get('libelle')?.value }}
                                        </h3>
                                        <p class="text-sm
text-green-700 mt-1">

{{docControl.get('avis')?.value || 'Aucune observation'}}
                                        </p>
                                    </div>
                                </div>
                                }
                                @else {

@if(docControl.get('avis')?.value[0]?.motif == 'V'){
                                <div class="bg-green-50 border
border-green-200 rounded-lg p-4 flex items-start gap-4">
```

```
                                    <div class="w-6 h-6 text-green-600
shrink-0 mt-0.5">
                                        <svg
xmlns="http://www.w3.org/2000/svg" class="w-5 h-5" viewBox="0 0 24 24">
                                            <path d="M20 6.5L9.5 18 4
12.5" fill="none" stroke="#16A34A"
                                                stroke-width="2.5"
stroke-linecap="round" stroke-linejoin="round" />
                                        </svg>

                                    </div>
                                    <div>
                                        <h3 class="font-semibold
text-green-800">
                                            {{
docControl.get('libelle')?.value }}
                                        </h3>
                                        <p class="text-sm
text-green-700 mt-1">

@if(docControl.get('avis')?.value[0]?.motif == 'V'){
                                        <div>

{{docControl.get('avis')?.value[0]?.observation }}
                                        </div>
                                        }
                                        </p>
                                    </div>
                                </div>
                                }

@if(docControl.get('avis')?.value[0]?.motif == 'R') {
                                <div class="bg-rose-50 border
border-rose-200 rounded-lg p-4 flex items-start gap-4">
                                    <div class="w-6 h-6 text-rose-600
shrink-0 mt-0.5">
                                        <svg
xmlns="http://www.w3.org/2000/svg" class="w-5 h-5" viewBox="0 0 24 24">
                                            <line x1="4" y1="4" x2="20"
y2="20" stroke="#E02424" stroke-width="2.5"
                                                stroke-linecap="round"
/>
```

```
                                                <line x1="20" y1="4" x2="4"
y2="20" stroke="#E02424" stroke-width="2.5"
                                                      stroke-linecap="round"
/>
                                            </svg>
                                        </div>
                                        <div>
                                            <h3 class="font-semibold
text-rose-800">
                                                {{
docControl.get('libelle')?.value }}
                                            </h3>
                                            <ul class="list-disc
list-inside text-sm text-rose-700">
                                                @for (motif of
docControl.get('avis')?.value[0]?.avisPiece; track $index) {
                                                <li>{{ motif?.avis?.libelle
}}</li>
                                                }
                                            </ul>
                                        </div>
                                    </div>
                                    }



                                }
                                }
                            </div>


                            <div class="grid grid-cols-4 pb-8 gap-4
px-5">
                                @for (docControl of documentFormGroups;
track $index) {
                                <div
                                    class="rounded-2xl bg-white border
flex flex-col justify-between space-y-2 relative group
overflow-hidden">
                                    <div
                                        class="opacity-0
group-hover:opacity-100 group-hover:translate-x-0 transition-all
duration-300 delay-100 ease-in-out absolute z-[1000] w-full h-full
```

```
top-0 left-0 bg-white/50 backdrop-blur-sm rounded-2xl flex items-center
space-x-4 justify-center">
                                                @if
(docControl.get('fichier')?.value) {
                                                <button type="button"
                                                        class="bg-green-600 px-4
rounded-full w-[40px] h-[40px] flex justify-center items-center"
                                                        title="Aperçu"
(click)="previewFile(docControl.get('fichier')?.value)">
                                                        <i class="ki-duotone ki-eye
text-white text-xl"></i>
                                                </button>
                                                }

                                                @if (shouldSubmit()) {
                                                <button type="button"
                                                        class="bg-green-600 px-4
rounded-full w-[40px] h-[40px] flex justify-center items-center"
                                                        title="Aperçu"
(click)="openPieceModal($index)">
                                                        <i class="ki-duotone
ki-pencil text-white text-xl"></i>
                                                </button>
                                                }
                                        </div>

                                        @if
(docControl.get('requis')?.value) {
                                        <div
                                                class="absolute font-medium
top-0 right-1 px-2 py-0 text-center flex items-center justify-center
text-red-500 z-[1500] rounded-3xl">
                                                <span class="text-sm
text-inherit font-bold">Requis</span>
                                        </div>
                                        }

                                        <div class="flex flex-col
space-y-2">
                                                <div class="p-4">
                                                        @if
(docControl.get('fichier')?.value) {
```

```
                                        <lucide-icon
name="file-check-2" size="100" strokeWidth="1"

class="stroke-green-600" />
                                        } @else {
                                        <lucide-icon
name="file-plus-2" size="100" strokeWidth="1"
                                            class="stroke-gray-400"
/>
                                        }
                                    </div>

                                    @if
(docControl.get('fichier')?.value) {
                                    <div class="px-4">
                                        {{typeof
docControl.get('fichier')?.value === 'string'
                                            ?
docControl.get('libelle')?.value + '.' +

(docControl.get('extension')?.value || 'pdf')
                                            :
docControl.get('fichier')?.value.name
                                        }}
                                    </div>
                                    }
                                </div>

                                <div class="border-t px-4 py-2
bg-gray-500 text-white rounded-b-2xl line-clamp-2">

{{docControl.get('libelle')?.value}}
                                </div>
                            </div>
                            }
                        </div>

                        <!-- <div class="overflow-x-auto px-6">
                            <table class="w-full text-sm">
                                <thead class="bg-slate-50">
                                    <tr>
                                        <th
class="table-th-light">
```

```
                                                    Pièce à fournir
                                                </th>
                                                <th
class="table-th-light">
                                                    Observation
                                                </th>
                                                <th
class="table-th-light">
                                                    Actions
                                                </th>
                                            </tr>
                                        </thead>
                                        <tbody>
                                            <ng-container
                                                *ngFor="
                                                    let docControl of
documentFormGroups;
                                                    let i = index
                                                "
                                            >
                                                <tr
                                                    class="border-t
border-slate-200"

[formGroup]="docControl"
                                                >
                                                    <td

class="table-td-light font-medium text-slate-700"
                                                    >
                                                        @if (

docControl.get(

'options'
                                                            )?.value
                                                        ) {
                                                            <select2

[data]="

selectData[
```

```
'diplomes'
]
"
[value]="
docControl.get(
'libelle'
).value
"
(update)="
onSelectDiplome(
$event,
i
)
"
[minCountForSearch]="
2
"
class=""
/>
} @else {
{{
docControl.get(
'libelle'
)?.value
}}
}
<span
class="text-red-500"
```

```
                                                                *ngIf="
docControl.get(
'requis'
)?.value
                                                                "
                                                                >*</span
                                                            >
                                                        </td>
                                                        <td
class="table-td-light font-medium text-slate-700"
                                                        >
                                                            {{
docControl.get(
                                                                'avis'
                                                            )?.value ??
                                                                'Aucune
observation'
                                                            }}
                                                        </td>
                                                        <td
class="table-td-light">
                                                            <ng-container
                                                                *ngIf="
docControl.get(
'fichier'
)?.value
                                                                "
                                                            >
                                                                <div
class="flex items-center gap-2 text-green-700"
                                                                >
                                                                    <i
class="ki-duotone ki-check-circle text-lg"
```

```
></i>
<span
class="font-medium text-xs truncate max-w-xs"
>
{{
typeof docControl.get(
'fichier'
)
?.value ===
'string'
? docControl.get(
'libelle'
)
?.value +
'.' +
(docControl.get(
'extension'
)
?.value ||
'pdf')
: docControl.get(
'fichier'
)
```

```
?.value
.name
                                                                                }}
                                                                            </span>
                                                                            <button
type="button"
class="btn-icon-light"
title="Aperçu"
(click)="
previewFile(
docControl.get(
'fichier'
)
?.value
)
                                                                                "
                                                                            >
                                                                                <i
class="ki-duotone ki-eye"
></i>
</button>
                                                                        </div>
                                                                    </ng-container>
                                                                    @if
(shouldSubmit()) {
                                                                        <button
type="button"
```

```
class="btn-icon-light"

title="Ajouter/Modifier"

(click)="

openPieceModal(

i
                                                                                )
                                                                                "
                                                                                >
                                                                                <i

class="ki-duotone ki-pencil"
                                                                                ></i>
                                                                                </button>
                                                                                }
                                                                                </td>
                                                                                </tr>
                                                                                </ng-container>
                                                                                </tbody>
                                                                                </table>
                                                                                </div> -->

                                                                                @if (shouldSubmit()) {
                                                                                <div class="p-6 sm:p-8 form-actions-light">
                                                                                <button type="submit"
[disabled]="!form1.valid || loadingSubmission"
                                                                                class="btn-action-light
btn-primary">
                                                                                @if (loadingSubmission) {
                                                                                <span class="spinner-border
spinner-border-sm" role="status"
                                                                                aria-hidden="true"></span>
                                                                                Enregistrement en cours...
                                                                                } @else {
                                                                                Enregistrer les pièces
                                                                                }
                                                                                </button>
                                                                                </div>
                                                                                }
```

```
                        </form>
                    </div>

                    <!-- ONGLET 2: LIEU DE DÉPÔT -->
                    <div *ngSwitchCase="2" class="p-6 sm:p-8
animate-fadeIn">
                        <form *ngIf="formDepot" [formGroup]="formDepot"
(ngSubmit)="submitCandidature2()"
                              class="space-y-8">
                            <fieldset class="space-y-5">
                                <legend class="form-legend-light">
                                    Sélectionnez un lieu de dépôt
                                </legend>

                                <div class="space-y-4">
                                    <label *ngFor="let lieu of
lieuxDepot"
                                           class="radio-label-light flex
items-center gap-3">
                                        <input type="radio"
formControlName="lieuDepot" [value]="lieu?.code"
                                               [disabled]="isFormLocked"
class="form-radio-light" />
                                        <span>{{ lieu.libelle }}</span>
                                    </label>
                                </div>
                            </fieldset>

                            @if (shouldSubmit()) {
                            <div class="form-actions-light">
                                <button type="submit" [disabled]="
                                        !formDepot.valid ||
                                        loadingSubmission ||
                                        isFormLocked
                                    " class="btn-action-light
btn-primary">
                                    @if (loadingSubmission) {
                                    <span class="spinner-border
spinner-border-sm" role="status"
                                          aria-hidden="true"></span>
                                    Enregistrement en cours...
                                    } @else {
                                    Enregistrer le lieu de dépôt
```

```
                                    }
                                </button>
                            </div>
                            }
                        </form>
                    </div>

                    <!-- ONGLET 3: CENTRE DE COMPOSITION -->
                    <div *ngSwitchCase="3" class="p-6 sm:p-8
animate-fadeIn">
                        <form *ngIf="formCommune"
[formGroup]="formCommune" (ngSubmit)="submitCandidature3()"
                            class="space-y-8">
                            <fieldset class="space-y-5">
                                <legend class="form-legend-light">
                                    Sélectionnez une commune de
composition
                                </legend>

                                <div class="space-y-4">
                                    <label *ngFor="
                                        let centre of
centreComposition
                                    " class="radio-label-light
flex items-center gap-3">
                                        <input type="radio"
formControlName="commune" [value]="centre?.code"
                                            [disabled]="isFormLocked"
class="form-radio-light" />
                                        <span>{{ centre?.libelle
}}</span>
                                    </label>
                                </div>
                            </fieldset>

                            @if (shouldSubmit()) {
                            <div class="form-actions-light">
                                <button type="submit" [disabled]="
                                    !formDepot.valid ||
                                    loadingSubmission ||
                                    isFormLocked
                                " class="btn-action-light
btn-primary">
```

```
                                    @if (loadingSubmission) {
                                    <span class="spinner-border
spinner-border-sm" role="status"
                                        aria-hidden="true"></span>
                                    Enregistrement en cours...
                                    } @else {
                                    Enregistrer la commune de
                                    composition
                                    }
                                </button>
                            </div>
                            }
                        </form>
                    </div>

                    <!-- ONGLET 4: EPREUVE DE COMPOSITION -->
                    <div *ngSwitchCase="4" class="p-6 sm:p-8
animate-fadeIn">
                        <form *ngIf="formEpreuve"
[formGroup]="formEpreuve" (ngSubmit)="submitCandidature4()"
                            class="space-y-8">
                            <fieldset class="space-y-5">
                                @if(epreuvesFacultatif.length > 0) {
                                <legend class="form-legend-light">
                                    Choisissez une épreuve facultative
dans
                                    laquelle vous voulez composer
                                </legend>
                                }
                                @else {
                                <legend class="form-legend-light">
                                    Liste des épreuves à composer
                                </legend>
                                }

                                <div class="overflow-x-auto">
                                    <table
                                        class="min-w-full divide-y
divide-gray-200 border border-gray-300 shadow-sm rounded-lg">
                                        <thead class="bg-gray-100">
                                            <tr>
                                                <th class="px-4 py-2
text-left text-sm font-semibold text-gray-700">
```

```
                                                    Épreuve
                                                </th>
                                                <th class="px-4 py-2
text-left text-sm font-semibold text-gray-700">
                                                    Sélectionner
                                                </th>
                                            </tr>
                                        </thead>
                                        <tbody class="divide-y
divide-gray-200 bg-white">
                                            <!-- Épreuves obligatoires
-->
                                            <tr *ngFor="let epreuve of
epreuves" class="hover:bg-gray-50 transition">
                                                <td class="px-4 py-2
text-sm text-gray-800">
                                                    {{
epreuve?.epreuveRecrutement?.libelle }}
                                                    <span
                                                        class="ml-2
inline-flex items-center px-2 py-0.5 rounded bg-red-100 text-red-700
text-xs">
                                                        Obligatoire
                                                    </span>
                                                </td>
                                                <td class="px-4 py-2">
                                                    <!-- Toggle
désactivé pour obligatoire -->
                                                    <label
class="inline-flex items-center cursor-not-allowed">
                                                        <input
type="checkbox" checked disabled class="sr-only peer" />
                                                        <div
                                                            class="w-11
h-6 bg-gray-300 rounded-full relative transition
peer-checked:bg-gray-400">
                                                            <div

class="absolute top-0.5 left-0.5 bg-white w-5 h-5 rounded-full
transition-transform translate-x-5">
                                                            </div>
                                                        </div>
                                                    </label>
```

```
                                    </td>
                                </tr>

                                <!-- Épreuves facultatives -->
                                <tr *ngFor="let epreuve of epreuvesFacultatif"
                                    class="hover:bg-gray-50 transition">
                                    <td class="px-4 py-2 text-sm text-gray-800">
                                        {{ epreuve?.epreuveRecrutement?.libelle }}
                                    </td>
                                    <td class="px-4 py-2">
                                        <label class="inline-flex items-center cursor-pointer">
                                            <input type="checkbox" class="sr-only"
                                                [checked]="checkedEpreuveFacultative(epreuve.id)"
                                                (change)="onToggleFacultative(epreuve.id, $event)" />
                                            <!-- Toggle -->
                                            <div class="w-11 h-6 rounded-full relative transition-colors"
                                                [ngClass]="checkedEpreuveFacultative(epreuve.id) ? 'bg-blue-600' : 'bg-gray-200'">
                                                <div class="absolute top-0.5 left-0.5 w-5 h-5 bg-white rounded-full transition-transform duration-200 ease-in-out"
                                                    [ngClass]="{ 'translate-x-5': checkedEpreuveFacultative(epreuve.id) }">
                                                </div>
                                            </div>
                                        </label>
                                    </td>
                                </tr>
                            </tbody>
                        </table>
                    </div>
                </fieldset>
```

```
                                @if (shouldSubmit()) {
                                <div class="form-actions-light">
                                    <button type="submit" [disabled]="
                                                !formEpreuve.valid ||
                                                loadingSubmission ||
                                                isFormLocked ||
epreuvesFacultatif?.length == 0
                                            " class="btn-action-light
btn-primary">
                                        @if (loadingSubmission) {
                                        <span class="spinner-border
spinner-border-sm" role="status"
                                            aria-hidden="true"></span>
                                        Enregistrement en cours...
                                        } @else {
                                        Enregistrer les épreuves
                                        }
                                    </button>
                                </div>
                                }
                            </form>
                        </div>
                    </ng-container>
                </div>
            </main>
        </div>
    </div>

    @if (showPieceModal) {
    <app-piece-candidature-modal [show]="showPieceModal"
    [piece]="pieceCourante" [valeursExistantes]="valeursExistantes"
        [isLoading]="isLoading" [pageLabel]="'Ajout d\'une pièce'"
        [pageDescription]="'Merci de renseigner les informations
demandées'" (closeModal)="showPieceModal = false"
        (save)="onPieceSave($event)" />
    }
    }

    @if (showPaymentReportModal) {
    <app-payment-report-modal [display]="showPaymentReportModal"
    (onClick)="onTogglePaymentReportModal()"
        [status]="paymentTransactionStatus" [message]="htmlMessage" />
```

```
}
```

voici le sous composant information personnel

```
<!-- informations-personnelles/informations-personnelles.component.html
-->
<div class="p-6 sm:p-8 animate-fadeIn">
  <form *ngIf="form" [formGroup]="form" (ngSubmit)="onSubmit()"
class="space-y-8">
    <fieldset class="space-y-5">
      <legend class="form-legend-light">Identification</legend>

      <div class="grid grid-cols-1 md:grid-cols-3 gap-5">
        <div class="form-group-light">
          <label for="nom" class="form-label-light">Nom de famille
*</label>
          <input
            type="text"
            id="nom"
            formControlName="nom"
            class="form-input-light"
            placeholder="Votre nom"
          />
        </div>

        <div class="form-group-light">
          <label for="prenoms" class="form-label-light">Prénoms
*</label>
          <input
            type="text"
            id="prenoms"
            formControlName="prenoms"
            class="form-input-light"
            placeholder="Vos prénoms"
          />
        </div>

        <div class="form-group-light">
          <label class="form-label-light">Sexe *</label>
          <div class="flex items-center gap-6 pt-2">
            <label class="radio-label-light">
              <input
                type="radio"
```

```
                formControlName="sexe"
                value="F"
                class="form-radio-light"
              />
              <span>Féminin</span>
            </label>
            <label class="radio-label-light">
              <input
                type="radio"
                formControlName="sexe"
                value="M"
                class="form-radio-light"
              />
              <span>Masculin</span>
            </label>
          </div>
        </div>

        <div class="form-group-light">
          <label class="form-label-light">Situation de handicap</label>
          <div class="flex items-center gap-6 pt-2">
            <label class="radio-label-light">
              <input
                type="radio"
                formControlName="handicap"
                value="non"
                class="form-radio-light"
              />
              <span>Non</span>
            </label>
            <label class="radio-label-light">
              <input
                type="radio"
                formControlName="handicap"
                value="oui"
                class="form-radio-light"
              />
              <span>Oui</span>
            </label>
          </div>
        </div>
```

```
        <div *ngIf="isHandicape" class="form-group-light w-full
md:col-span-2">
          <label class="form-label-light">Type de handicap</label>
          <div class="flex items-center gap-6 pt-2 w-full">
            <select2
              [data]="selectData['typeHandicap']"
              [value]="form.get('typeHandicap')?.value"
              formControlName="typeHandicap"
              (update)="onSelect($event,'typeHandicap')"
              placeholder="Sélectionner un type..."
              [minCountForSearch]="2"
              class="form-group-light w-full"
            />
          </div>
        </div>
      </div>
    </fieldset>

    <fieldset class="space-y-5">
      <legend class="form-legend-light">Coordonnées et
Naissance</legend>

      <div class="grid grid-cols-1 md:grid-cols-2 gap-5">
        <div class="form-group-light">
          <label for="dateNaissance" class="form-label-light">Date de
naissance *</label>
          <input
            type="date"
            id="dateNaissance"
            formControlName="dateNaissance"
            class="form-input-light"
          />
        </div>

        <div class="form-group-light">
          <label for="lieuNaissance" class="form-label-light">Lieu de
naissance *</label>
          <input
            type="text"
            id="lieuNaissance"
            formControlName="lieuNaissance"
            class="form-input-light"
            placeholder="Ex: Cotonou, Bénin"
```

```
          />
        </div>

        <div class="md:col-span-2 form-group-light">
          <label for="adresse" class="form-label-light">Adresse
complète *</label>
          <textarea
            id="adresse"
            formControlName="adresse"
            class="form-input-light"
            rows="3"
            placeholder="Quartier, rue, ville..."
          ></textarea>
        </div>

        <div class="form-group-light">
          <label for="telephone" class="form-label-light">Numéro de
téléphone *</label>
          <input
            type="tel"
            id="telephone"
            formControlName="telephone"
            class="form-input-light"
            placeholder="+229 XX XX XX XX"
          />
        </div>
      </div>
    </fieldset>

    <div *ngIf="shouldSubmit()" class="form-actions-light">
      <button
        type="submit"
        [disabled]="!form.valid || loadingSubmission || isFormLocked"
        class="btn-action-light btn-primary">
        <ng-container *ngIf="loadingSubmission">
          <span class="spinner-border spinner-border-sm" role="status"
aria-hidden="true"></span>
          Enregistrement en cours...
        </ng-container>
        <ng-container *ngIf="!loadingSubmission">
          Enregistrer vos informations
        </ng-container>
      </button>
```

```
    </div>
  </form>
</div>
```

```
import { Component, EventEmitter, Input, OnInit, Output } from
'@angular/core';
import {
    FormBuilder,
    FormGroup,
    ReactiveFormsModule,
    Validators
} from '@angular/forms';
import { CommonModule } from '@angular/common';
import { Select2, Select2Data, Select2UpdateEvent } from
'ng-select2-component';
import { FullCandidature } from
'../../../../../../../core/models/entities/recrutement/candidature.mode
l';
import { PaginationOptions } from
'../../../../../../../core/models/others/http.models';
import { TypeHandicapService } from
'../../../../../../../core/services/referentiels/type-handicap.service'
;
import { StubPagination } from
'../../../../../../../core/models/builders/pagination.builder';

@Component({
    selector: 'app-informations-personnelles',
    standalone: true,
    imports: [CommonModule, ReactiveFormsModule, Select2],
    templateUrl: './informations-personnelles.component.html',
    styleUrl: './informations-personnelles.component.css'
})
export class InformationsPersonnellesComponent implements OnInit {
    @Input() candidature!: FullCandidature;
    @Input() isFormLocked: boolean = false;
    @Input() loadingSubmission: boolean = false;

    @Output() submitForm = new EventEmitter<any>();

    form!: FormGroup;
```

```
    selectData: { [key: string]: Select2Data } = {};
    pageOptions: PaginationOptions = new StubPagination().build();

    constructor(
        private fb: FormBuilder,
        private typeHandicapService: TypeHandicapService
    ) {}

    ngOnInit(): void {
        this.loadTypeHandicap();
    }

    private loadTypeHandicap(): void {
        this.typeHandicapService.all$(this.pageOptions).subscribe({
            next: (response) => {
                this.selectData['typeHandicap'] = response.content.map(
                    (item) => ({
                        value: item.id,
                        label: item.libelle
                    })
                );
                this.initializeForm();
            },
            error: (err) => {
                console.error(
                    'Erreur lors du chargement des types de handicap
:',
                    err
                );
                this.selectData['typeHandicap'] = [];
                this.initializeForm();
            }
        });
    }

    private initializeForm(): void {
        const typeHandicapValue =
            this.candidature?.candidat?.typeHandicap?.id || '';

        this.form = this.fb.group({
            nom: [
                this.candidature?.candidat?.lastName || '',
                Validators.required
```

```
            ],
            prenoms: [
                this.candidature?.candidat?.firstName || '',
                Validators.required
            ],
            sexe: [
                this.candidature?.candidat?.gender || 'M',
                Validators.required
            ],
            handicap: [
                this.candidature?.candidat?.handicap || 'non',
                Validators.required
            ],
            typeHandicap: [typeHandicapValue],
            dateNaissance: [
                this.candidature?.candidat?.birthDate || '',
                Validators.required
            ],
            lieuNaissance: [
                this.candidature?.candidat?.birthPlace || '',
                Validators.required
            ],
            adresse: [
                this.candidature?.candidat?.adresse || '',
                Validators.required
            ],
            telephone: [
                this.candidature?.candidat?.phone || '',
                Validators.required
            ]
        });

        // Forcer la mise à jour du Select2 pour typeHandicap
        if (typeHandicapValue) {
            setTimeout(() => {
this.form.get('typeHandicap')?.setValue(typeHandicapValue);
            }, 100);
        }

        // Gérer la logique conditionnelle du handicap
        this.form.get('handicap')?.valueChanges.subscribe((value) => {
            const typeHandicapControl = this.form.get('typeHandicap');
```

```
            if (value === 'oui') {
typeHandicapControl?.setValidators([Validators.required]);
            } else {
                typeHandicapControl?.clearValidators();
                typeHandicapControl?.setValue('');
            }
            typeHandicapControl?.updateValueAndValidity();
        });
    }

    onSelect(event: Select2UpdateEvent, formControl: string): void {
        this.form.get(formControl)?.setValue(event.value);
    }

    onSubmit(): void {
        if (this.form.valid && !this.isFormLocked) {
            const formData = new FormData();

            formData.append('nom', this.form.value.nom);
            formData.append('prenoms', this.form.value.prenoms);
            formData.append('sexe', this.form.value.sexe);
            formData.append('handicap', this.form.value.handicap);
            formData.append(
                'typeHandicap',
                this.form.value.typeHandicap &&
this.form.value.handicap === 'oui'
                    ? this.form.value.typeHandicap
                    : '99'
            );
            formData.append('dateNaissance',
this.form.value.dateNaissance);
            formData.append('lieuNaissance',
this.form.value.lieuNaissance);
            formData.append('adresse', this.form.value.adresse);
            formData.append('telephone', this.form.value.telephone);
            formData.append('naturePiece', '15');

            this.submitForm.emit(formData);
        } else {
            this.form.markAllAsTouched();
        }
    }
```

```
    shouldSubmit(): boolean {
        return ['CD1E', 'CDN', 'CPA'].includes(
            this.candidature?.etape?.code || ''
        );
    }

    get isHandicape(): boolean {
        return this.form?.get('handicap')?.value === 'oui';
    }
}
```