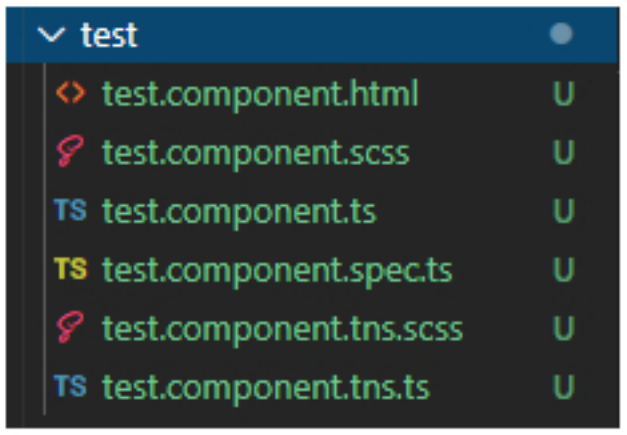
Création d'un nouveau component

Les vues de l'application s'appuie sur les mécanismes de `component` d'Angular.

1 - Générer un component vierge

- Se placer dans le répertoire `src/app/components/`
- Taper la commande `$ ng g c <nom_du_composant>`
  - La commande créer un dossier portant le nom du composant, ainsi que 4 fichiers.
  - Dupliquer et renommer, en ajoutant **.tns**, les fichiers **component.ts** et **.scss**.
  - EXEMPLE pour un composant appelé `test` :



2 - Implémentation des fichier .ts

Les fichiers component.ts font office de contrôleur de vue, voici un squelette de controleur :

```
@Component({
    selector: 'cma-login',
    templateUrl: './login.component.html',
    styleUrls: ['./login.component.scss']
})
export class LoginComponent implements OnInit {
    /* ICI LES ATTRIBUTS PRIVES POUR LE FONCTIONNEMENT INTERNET DU CONTRÔLEUR OU
       PUBLIC SI ON L'EXPOSER A LA VUE */

    constructor(
        private authService: AuthService, // QUELQUES EXEMPLE D'INJECTIONS DE DÉPENDANCE
        private router: Router,
        private _L: i18nService
    ) {
        this.loging = false;
        this.textUI = {};
    }

    ngOnInit() {
        /* ICI ON SOUSCRIT LE MODULE A UN SERVICE OU ON INITIALISE  */
    }
}
```

3 - Lien avec la couche service

Un service peut être utilisé par un autre service ou un contrôleur. Pour utiliser un service depuis un contrôleur il faut :

1. L'injecter dans le contrôleur

```
@Component({
    selector: 'cma-list-planning',
    templateUrl: './list-planning.component.html',
    styleUrls: ['./list-planning.component.scss']
})
export class ListPlanningComponent implements OnInit, OnDestroy {
    listPlanning: (ActivityLineModelAdapter | DaySeparatorLineModelAdapter)[];

    private planningSubscription: Subscription;
    private previousPlanningDate: Date;

    constructor( /* INJECTION DE DEUX SERVICE */
        private planningService: PlanningService,
        private planningModelTransformer: ListPlanningModelTransformer
    ) {
        this.listPlanning = new Array();
    }
    ...
}
```

2. Souscrire si besoins

```
@Component({
    selector: 'cma-list-planning',
    templateUrl: './list-planning.component.html',
    styleUrls: ['./list-planning.component.scss']
})
export class ListPlanningComponent implements OnInit, OnDestroy {
    ...

    ngOnInit() {
        /* SOUSCRIPTION DU CONTRÔLEUR A UN OU PLUSIEURS SERVICE */
        this.planningSubscription = this.planningService.planningSubject.subscribe({
            next: (planning: Planning) => {
                this.listPlanning = this.planningModelTransformer.createViewModelAdapterFromModel(
                    planning
                ).activityLines;
            }
        });
        this.planningService.emitPlanningSubject(); // <== UNE FOIS SOUSCRIT, IL FAUT PENSER A FAIRE ÉMETTRE LA
    }
}
```

3. Utiliser le service

```
@Component({
    selector: 'cma-list-planning',
    templateUrl: './list-planning.component.html',
    styleUrls: ['./list-planning.component.scss']
})
export class ListPlanningComponent implements OnInit, OnDestroy {
    ...
    private fetchPlanning() {
        let _this = this;
        this.isBusy = true;

        this.planningService.fetchPlanning(this.selectedDateTimeStamp).then(() => {
            _this.displayError = false;
            _this.isBusy = false;
            _this.isComponentReady = true;
        }).catch(() => {
            _this.displayError = true;
            let date: Date = new Date(_this.selectedDateTimeStamp);
            _this._L.translate("ui_planning.impossible_to_fetch_planning", (translatedMessage) => {
                _this.notificationService.pushAlert(translatedMessage, MessageType.ERROR, 5000);
            }, { month: date.getMonthName(), year: date.getFullYear() });
            _this.isBusy = false;
        });
    }
    ...
}
```

4. Penser à désinscrire le contrôleur avant sa destruction

```
@Component({
    selector: 'cma-list-planning',
    templateUrl: './list-planning.component.html',
    styleUrls: ['./list-planning.component.scss']
})
export class ListPlanningComponent implements OnInit, OnDestroy {

    ...

    ngOnDestroy(): void {
        if (this.planningSubscription) {
            this.planningSubscription.unsubscribe(); // <== DÉSINSCRIRE LE CONTRÔLEUR UNE FOIS DÉTRUIT
        }
    }
}
```