Création d'un service

Les services renferme la logique applicative, s'il y a du cache à effectuer sur une données, ce sera fait au niveau de la couche service. Pour le moment, il n'existe pas de cache générique, chaque service implémente lui même sont propre cache mais il serait judicieux à termes de développer quelque chose de plus générique ; un service `Cache.service.ts` par exemple.

1 - Création du fichier

Les fichiers services se trouvent dans le package `src/app/services/`, et doivent être nommés `<nomService>.service.ts`.
NOTE : Il peut être nécessaire de créer un service spéciale pour la version MOBILE :
`<nomService>.service.` **tns** `.ts`. (voir le principe de fonctionnement d'une application hybride).

2 - Implémentation du service :

Un service qui manipule une donnée de référence (ici le planning du PN) doit la mettre à disposition par le biais d'observables (rxjs). Le choix fait dans 'Crew Mobile Access' est de mettre à disposition de l'utilisateur du service un sujet (au sens de RXJS) permettant de souscrire à la donnée, et une méthode `emit<nom_donnée>Subject()`, permettant de demander au service de broadcaster la données à tous les souscrits.

NOTE : Lorsque le service broadcaste la donnée, il se doit de fournir une copie profonde, la donnée détenue par le service doit être modifiée uniquement par ce dernier, ou par un tiers uniquement en passant par des méthodes `mutateurs` proposées par le service.

```
@Injectable()
export class PlanningService {
    private planning: Planning;
    _planningSubject: Subject<Planning>;

    constructor(
        ...
    ) {
        this._planningSubject = new Subject();
    }
    get planningSubject() { // <== Expose le sujet
        return this._planningSubject;
    }

    emitPlanningSubject(): void { // <== Permet de demander au service de broadcaster la donnée (ici la donnée c'est le Planning)
        this._planningSubject.next(Object.create(this.planning)); // La donnée est dupliquée grâce à la méthode Object.create()
    }
}
```

3 - Lien avec la couche DAO

Les DAO sont responsables de la communication avec le backend. Les services utilisent les DAO pour accéder à la donnée ou pour la mettre à jour sur KEOPS (en passant par Keops Bridge). Dans `Crew Mobile Access`, les services proposent 2 type de méthodes permettant la récupération de donnée :

- les méthodes mettant en oeuvre le pattern observable
- et celle qui ne le font pas

Dans les deux cas, une méthode du service doit toujours retourner une promesse (Promise). Il existe un objet du model (ServiceResponse) permettant de formuler la réponse d'un service lorsque le contenu de la réponse doit être retournée même en cas d'échec.

```
@Injectable()
export class PlanningService {
    private planning: Planning;
    _planningSubject: Subject<Planning>;

    constructor(
    private activityDAO: ActivityDAO, // <== INJECTION DU DAO
) {
    ...
}

/* EXEMPLE D'UNE METHODE METTANT EN OEUVRE LE PATTERN OBSERVABLE */
fetchPlanning(timeStamp: number, forceRefresh: boolean = false): Promise<void> {
    let _this = this;
    let startDate = new Date(timeStamp);
    startDate.setDate(1);
    let endDate = new Date(startDate).setOnLastDayOfMonth();

    startDate.setOnWeekMonday();
    endDate.setOnWeekSunday();

    return new Promise((success, reject) => {
            /* INVOCATION DU DAO POUR RÉCUPÉRER UNE LISTE D'ACTIVITÉS */
            _this.activityDAO.getActivitiesBetweenTimestamps(startDate.getTime(), endDate.getTime()).then((activities: Activity[
                _this.planning = _this.createPlanningFromActivitiesListForTimeStamp(timeStamp, activities);
                _this.emitPlanningSubject(); // <== LA DONNEE A ETE MISE A JOUR, ON LA BROADCAST POUR EN AVERTIR LES SOUSCRIPTEU
                success();
            }).catch(() => {
                _this.logger.error("Error when fetching planning from server");
                reject();
            });
        });
    }
}
```