ORSYS formation

Formation ANGULAR

Chapitre 3

Animé par Michel BOCCIOLESI

Table des matières

Récupérer les sources du projet de formation ici

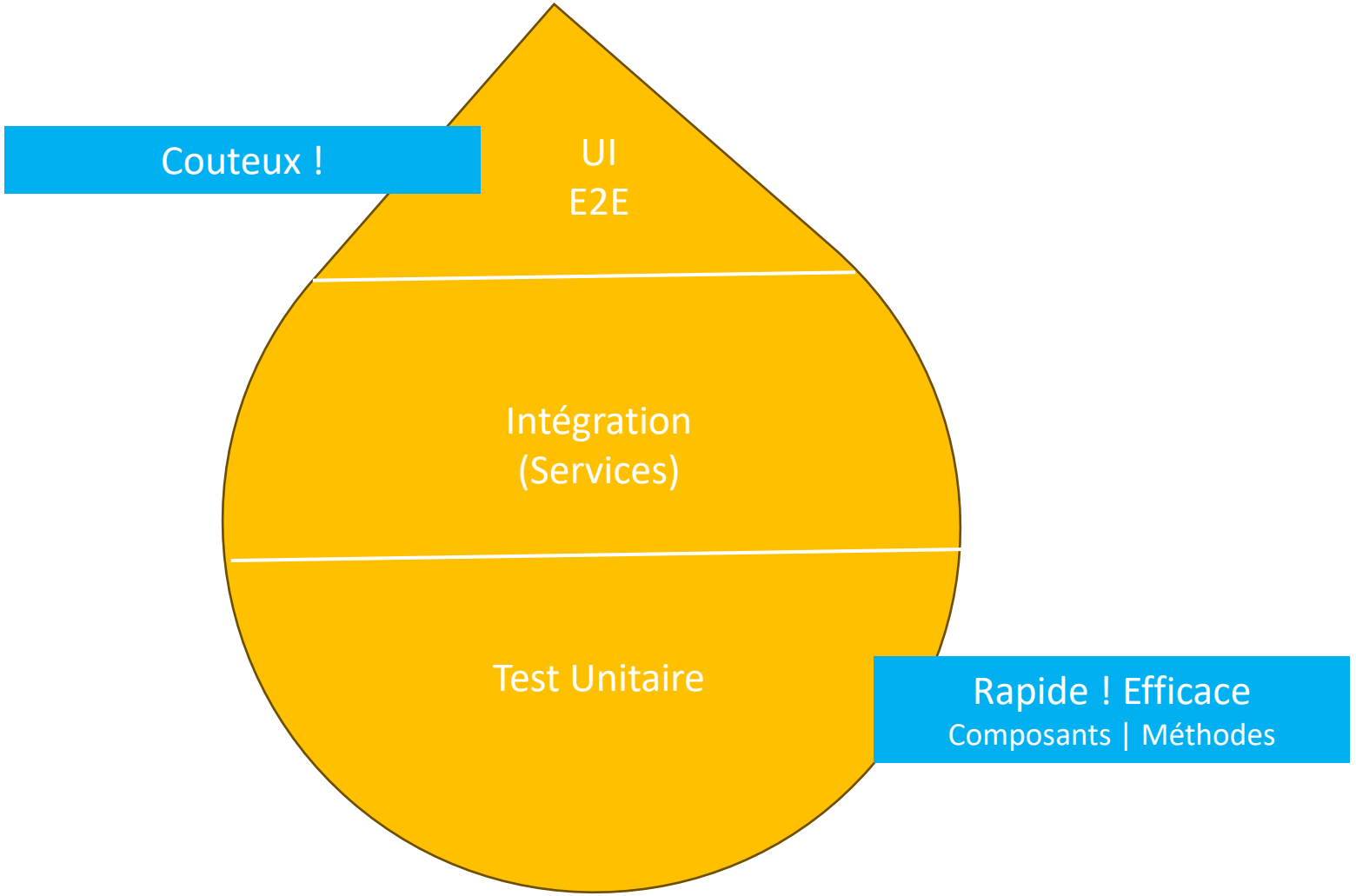
Les Tests Unitaires

Le Test

Le test (ou le Test Driven Development) permet de :

- Documenter le projet au fur et à mesure de sa création
- Faciliter les migrations entre versions majeures
- Simplifier la recherche d'erreurs
- Isoler l'objet de son contexte et de tester son intégration (Mock)

Le Test et la Documentation



1Er exemple : Apprendre de l'existant

app.component.ts

TP07-tests-unitaires > src > app > app.component.ts > AppComponent > titl

```ts
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  templateUrl: './app.component.html',
  styleUrls: ['./app.component.css']
})
export class AppComponent {
  title = 'TU';
}

```

app.component.html

TP07-tests-unitaires > src > app > app.component.html > h1

```html
<h1>{{title}}</h1>
```

app.component.spec.ts

TP07-tests-unitaires > src > app > app.component.spec.ts > describe('AppComponent') callback

```ts
import { TestBed } from '@angular/core/testing';
import { AppComponent } from './app.component';

// -----------------------------------------------------
// describe permet de créer un Objet de Test Unitaire
// -----------------------------------------------------
describe('AppComponent', () => {
  beforeEach(() => TestBed.configureTestingModule({
    declarations: [AppComponent],
  }));

  // it est une fonction qui crée une spécification
  it('should create the app', () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.componentInstance;
    // une spéc (it) attend un retour (expectation)
    expect(app).toBeTruthy(); // matchers Jasmine
  });


  it(`should have as title 'TP07-tests-unitaires'`, () => {
    const fixture = TestBed.createComponent(AppComponent);
    const composant = fixture.componentInstance;
    expect(composant.title).toEqual('TU');
  });


  it('should render title', () => {
    const fixture = TestBed.createComponent(AppComponent);
    fixture.detectChanges();
    const compiled = fixture.nativeElement as HTMLElement;
    expect(compiled.querySelector('h1')?.textContent).toContain('TU');
  });
});

```

2nd exemple : Injection de services

warn.service.ts

TP07-tests-unitaires > src > app > services > warn.service.ts > WarnService

```ts
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root'
})
export class WarnService {
  // ------------------------
  // méthodes
  // ------------------------
  public warnMessage = (msg: string) => {
    console.warn(`Le message est : ${msg}`);
  }
}
```

operations.service.ts

sts-unitaires > src > app > services > operations.service.ts > OperationsService > ajouter

```ts
import { Injectable } from '@angular/core';
import { WarnService } from './warn.service';

@Injectable({
  providedIn: 'root'
})
export class OperationsService {

  constructor(
    private _warn: WarnService
  ) { }

  // ------------------------
  // méthodes
  // ------------------------
  public ajouter = (nb1: number, nb2: number) => {
    this._warn.warnMessage('Ajout OK');
    // this._warn.warnMessage('Ajout OK');
    return nb1 + nb2;
  }

  public soustraire = (nb1: number, nb2: number) => {
    this._warn.warnMessage('Soustraction OK');
    return nb1 - nb2;
  }
}
```

operations.service.spec.ts

TP07-tests-unitaires > src > app > services > operations.service.spec.ts > describe('Test Injection de dépendances') ca

```ts
import { OperationsService } from "./operations.service";
import { WarnService } from "./warn.service";


// 1- Création de l'objet de Test
describe('Test Injection de dépendances', () => {

  // 2- liste des spécifications

  // it (xit - fit)
  it('Ajouter 2 nombres OK ?', () => {

    const operations = new OperationsService(new WarnService);
    const resultat = operations.ajouter(10, 5);
    // 3- expectation => ce que l'on attend
    expect(resultat).withContext('--- Erreur Expectation ---').toBe(15);

  });

  // 4- duplication du premier IT

  // it (xit - fit)
  it('Soustraire 2 nombres OK ?', () => {

    const operations = new OperationsService(new WarnService);
    const resultat = operations.soustraire(10, 5);
    // 3- expectation => ce que l'on attend
    expect(resultat).withContext('--- Erreur Expectation ---').toBe(5);

  });


})
```

2nd exemple : Injection de services

warn.service.ts

TP07-tests-unitaires > src > app > services > warn.service.ts > WarnService > logErreur

```ts
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root'
})
export class WarnService {
  // ------------------------
  // méthodes
  // ------------------------
  public warnMessage = (msg: string) => {
    console.warn(`Le message est : ${msg}`);
  }
  // ---
  public logErreur = (msg: string) => {
    console.error(`L'erreur' est : ${msg}`);
  }
}
```

operations.service.ts

sts-unitaires > src > app > services > operations.service.ts > OperationsService > ajouter

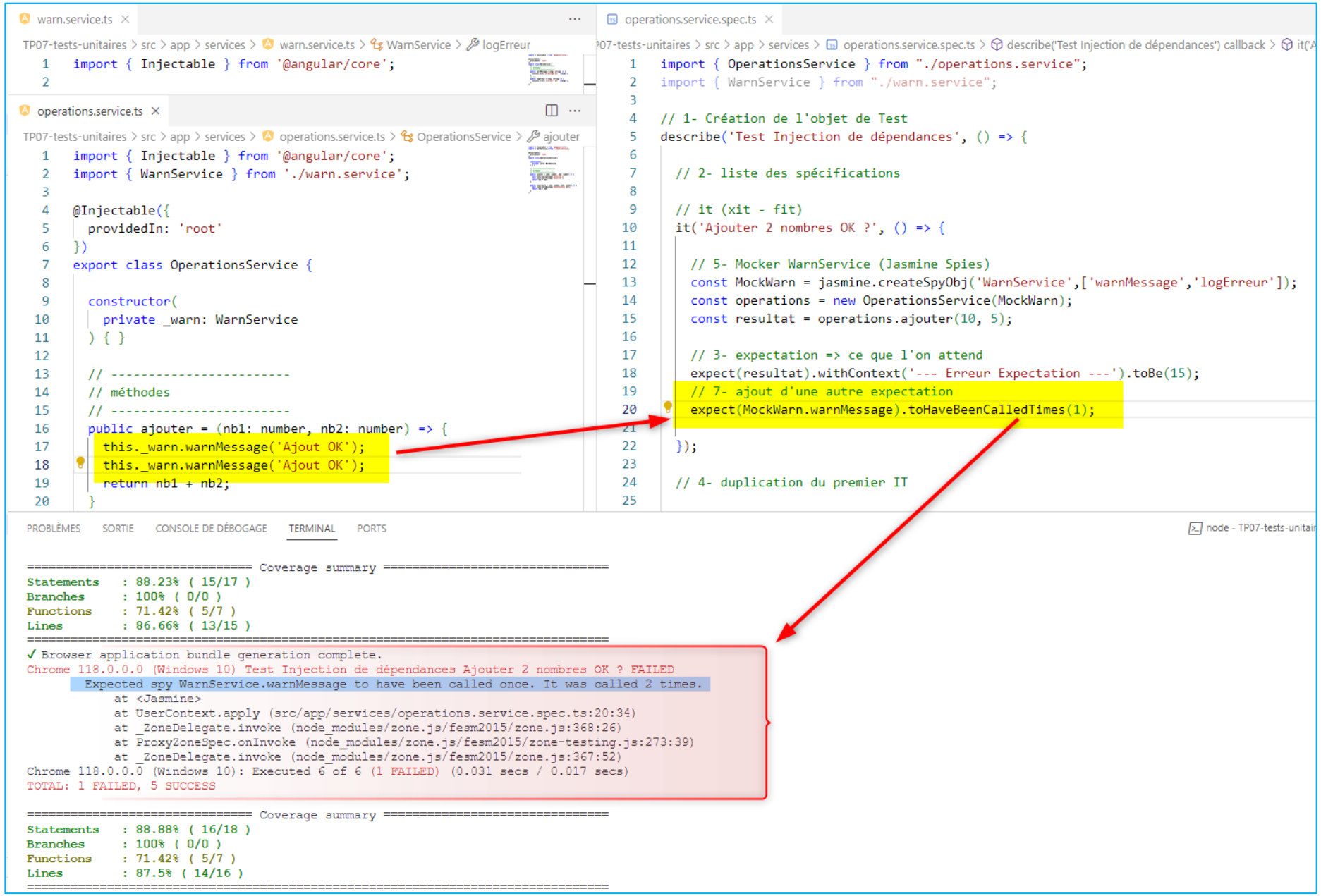
```ts
import { Injectable } from '@angular/core';
import { WarnService } from './warn.service';

@Injectable({
  providedIn: 'root'
})
export class OperationsService {

  constructor(
    private _warn: WarnService
  ) { }

  // ------------------------
  // méthodes
  // ------------------------
  public ajouter = (nb1: number, nb2: number) => {
    this._warn.warnMessage('Ajout OK');
    // this._warn.warnMessage('Ajout OK');
    return nb1 + nb2;
  }

  public soustraire = (nb1: number, nb2: number) =>
```

operations.service.spec.ts

TP07-tests-unitaires > src > app > services > operations.service.spec.ts > ...

```ts
import { OperationsService } from "./operations.service";
import { WarnService } from "./warn.service";

// 1- Création de l'objet de Test
describe('Test Injection de dépendances', () => {

  // 2- liste des spécifications

  // it (xit - fit)
  it('Ajouter 2 nombres OK ?', () => {

    // 5- Mocker WarnService (Jasmine Spies)
    const MockWarn = jasmine.createSpyObj('WarnService',['warnMessage','logErreur']);
    const operations = new OperationsService(MockWarn);
    const resultat = operations.ajouter(10, 5);

    // 3- expectation => ce que l'on attend
    expect(resultat).withContext('--- Erreur Expectation ---').toBe(15);

  });

  // 4- duplication du premier IT

  // it (xit - fit)
  it('Soustraire 2 nombres OK ?', () => {

    // 6- reproduire auusi ici le MockWarn
    const MockWarn = jasmine.createSpyObj('WarnService',['warnMessage','logErreur']);
    const operations = new OperationsService(MockWarn);
    const resultat = operations.soustraire(10, 5);
    // 3- expectation => ce que l'on attend
    expect(resultat).withContext('--- Erreur Expectation ---').toBe(5);

  });


})
```

2nd exemple : Injection de services



warn.service.ts

```
import { Injectable } from '@angular/core';
```

operations.service.ts

```
import { Injectable } from '@angular/core';
import { WarnService } from './warn.service';

@Injectable({
  providedIn: 'root'
})
export class OperationsService {

  constructor(
    private _warn: WarnService
  ) { }

  // ------------------------
  // méthodes
  // ------------------------
  public ajouter = (nb1: number, nb2: number) => {
    this._warn.warnMessage('Ajout OK');
    this._warn.warnMessage('Ajout OK');
    return nb1 + nb2;
  }
```

operations.service.spec.ts

```
import { OperationsService } from "./operations.service";
import { WarnService } from "./warn.service";

// 1- Création de l'objet de Test
describe('Test Injection de dépendances', () => {

  // 2- liste des spécifications

  // it (xit - fit)
  it('Ajouter 2 nombres OK ?', () => {

    // 5- Mocker WarnService (Jasmine Spies)
    const MockWarn = jasmine.createSpyObj('WarnService',['warnMessage','logErreur']);
    const operations = new OperationsService(MockWarn);
    const resultat = operations.ajouter(10, 5);

    // 3- expectation => ce que l'on attend
    expect(resultat).withContext('--- Erreur Expectation ---').toBe(15);
    // 7- ajout d'une autre expectation
    expect(MockWarn.warnMessage).toHaveBeenCalledTimes(1);

  });

  // 4- duplication du premier IT
```

PROBLÈMES SORTIE CONSOLE DE DÉBOGAGE TERMINAL PORTS

```
============================== Coverage summary ==============================
Statements   : 88.23% ( 15/17 )
Branches     : 100% ( 0/0 )
Functions    : 71.42% ( 5/7 )
Lines        : 86.66% ( 13/15 )
==============================================================================
✓ Browser application bundle generation complete.
Chrome 118.0.0.0 (Windows 10) Test Injection de dépendances Ajouter 2 nombres OK ? FAILED
        Expected spy WarnService.warnMessage to have been called once. It was called 2 times.
            at <Jasmine>
            at UserContext.apply (src/app/services/operations.service.spec.ts:20:34)
            at _ZoneDelegate.invoke (node_modules/zone.js/fesm2015/zone.js:368:26)
            at ProxyZoneSpec.onInvoke (node_modules/zone.js/fesm2015/zone-testing.js:273:39)
            at _ZoneDelegate.invoke (node_modules/zone.js/fesm2015/zone.js:367:52)
Chrome 118.0.0.0 (Windows 10): Executed 6 of 6 (1 FAILED) (0.031 secs / 0.017 secs)
TOTAL: 1 FAILED, 5 SUCCESS

============================== Coverage summary ==============================
Statements   : 88.88% ( 16/18 )
Branches     : 100% ( 0/0 )
Functions    : 71.42% ( 5/7 )
Lines        : 87.5% ( 14/16 )
==============================================================================
```

2nd exemple : Injection de services

warn.service.ts

TP07-tests-unitaires > src > app > services > warn.service.ts > WarnService > logErreur

```ts
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root'
})
export class WarnService {
  // ------------------------
  // méthodes
  // ------------------------
  public warnMessage = (msg: string) => {
    console.warn(`Le message est : ${msg}`);
  }
  // ---
  public logErreur = (msg: string) => {
    console.error(`L'erreur' est : ${msg}`);
  }
}
```

operations.service.ts

TP07-tests-unitaires > src > app > services > operations.service.ts > OperationsService > ajouter

```ts
import { Injectable } from '@angular/core';
import { WarnService } from './warn.service';

@Injectable({
  providedIn: 'root'
})
export class OperationsService {

  constructor(
    private _warn: WarnService
  ) { }

  // ------------------------
  // méthodes
  // ------------------------
  public ajouter = (nb1: number, nb2: number) => {
    this._warn.warnMessage('Ajout OK');
    // this._warn.warnMessage('Ajout OK');
    return nb1 + nb2;
  }

  public soustraire = (nb1: number, nb2: number) => {
    this._warn.warnMessage('Soustraction OK');
    return nb1 - nb2;
  }
}
```

operations.service.spec.ts

TP07-tests-unitaires > src > app > services > operations.service.spec.ts > describe('Test Injection de dépendances') callback

```ts
import { OperationsService } from "./operations.service";
import { WarnService } from "./warn.service";

// Création de l'objet de Test
describe('Test Injection de dépendances', () => {

  // ----------------------------------------------------
  // Zone de déclarations de variables - const - objets
  // ----------------------------------------------------
  let MockWarn:any;
  let operations:any;

  // ------------------------------------------------------------------------
  // traitement spécifiques communs (beforeEach - AfterEach - beforeAll)
  // ------------------------------------------------------------------------
  beforeEach(()=> {
    MockWarn = jasmine.createSpyObj('WarnService', ['warnMessage', 'logErreur']);
    operations = new OperationsService(MockWarn);
  });

  // ----------------------------
  //   liste des spécifications
  // ----------------------------

  it('Ajouter 2 nombres OK ?', () => {
    const resultat = operations.ajouter(10, 5);
    expect(resultat).withContext('--- Erreur Expectation ---').toBe(15);
    expect(MockWarn.warnMessage).toHaveBeenCalledTimes(1);
  });

  it('Soustraire 2 nombres OK ?', () => {
    const resultat = operations.soustraire(10, 5);
    expect(resultat).withContext('--- Erreur Expectation ---').toBe(5);
  });
})
```

2nd exemple : Injection de services

warn.service.ts

TP07-tests-unitaires > src > app > services > warn.service.ts > WarnService > warnMess

```ts
import { Injectable } from '@angular/core';

@Injectable({
  providedIn: 'root'
})
export class WarnService {
  // ------------------------
  // méthodes
  // ------------------------
  public warnMessage = (msg: string) => {
    console.warn(`Le message est : ${msg}`);
  }
  // ---
  public logErreur = (msg: string) => {
    console.error(`L'erreur' est : ${msg}`);
  }
}
```

operations.service.ts

TP07-tests-unitaires > src > app > services > operations.service.ts > OperationsService >

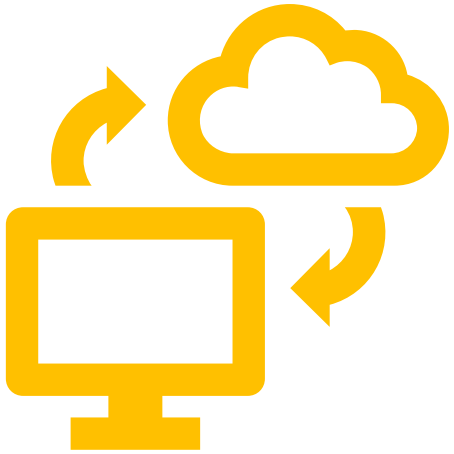
```ts
  // ------------------------
  // méthodes
  // ------------------------
  public ajouter = (nb1: number, nb2: number) => {
    this._warn.warnMessage('Ajout OK');
    // this._warn.warnMessage('Ajout OK');
    return nb1 + nb2;
  }

  public soustraire = (nb1: number, nb2: number) => {
    this._warn.warnMessage('Soustraction OK');
    return nb1 - nb2;
  }
}
```

operations.service.spec.ts

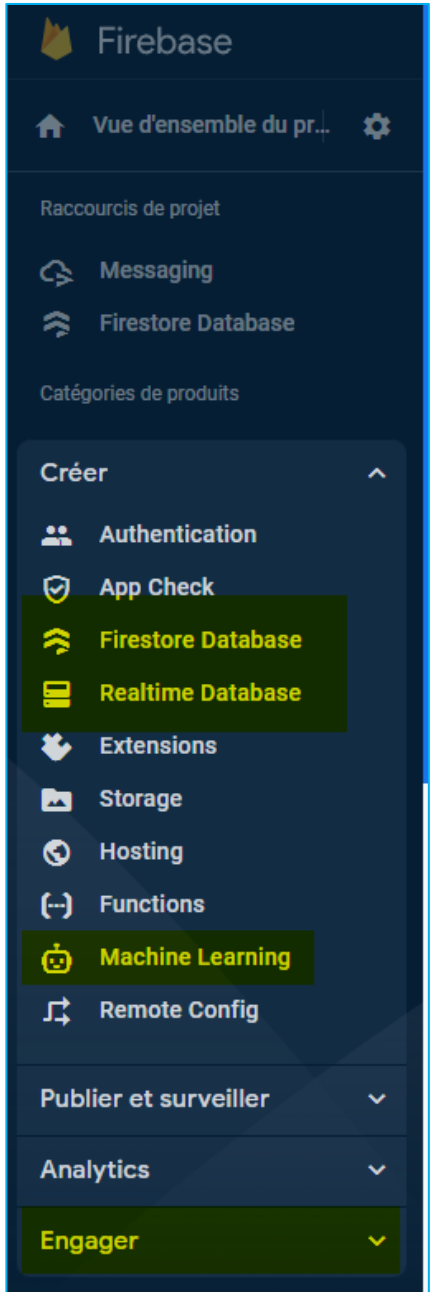
TP07-tests-unitaires > src > app > services > operations.service.spec.ts > describe('Test Injection de dépendances') callb

```ts
import { TestBed } from "@angular/core/testing";
import { OperationsService } from "./operations.service";
import { WarnService } from "./warn.service";

// Création de l'objet de Test
describe('Test Injection de dépendances', () => {
  // ------------------------------------------------
  // Zone de déclarations de variables - const - objets
  // ------------------------------------------------
  let MockWarn: any;
  let operations: any;
  // ------------------------------------------------------------------------
  // traitement spécifiques communs (beforeEach - AfterEach - beforeAll)
  // ------------------------------------------------------------------------
  beforeEach(() => {
    MockWarn = jasmine.createSpyObj('WarnService', ['warnMessage','logErreur']);
    // ------------------------------------------------------------------------
    // Création d'un Objet complet de test et de simulation : TESTBED
    // Créons une vraie injection de dépendances avec les providers
    // ------------------------------------------------------------------------
    TestBed.configureTestingModule({
      providers: [
        // injection de dépendances
        OperationsService,
        {
          provide: WarnService,
          // on n'utilise pas la méthode de WarnService
          // mais la méthode du Mock ('warnMessage', 'logErreur')
          useValue: MockWarn
        }
      ]
    });
    // ------------------------------------------
    operations = TestBed.inject(OperationsService);
    // ------------------------------------------
  });
  // ----------------------------
  //   liste des spécifications
  // ----------------------------
  it('Ajouter 2 nombres OK ?', () => {
    const resultat = operations.ajouter(10, 5);
    expect(resultat).withContext('--- Erreur Expectation ---').toBe(15);
    expect(MockWarn.warnMessage).toHaveBeenCalledTimes(1);
  });
```

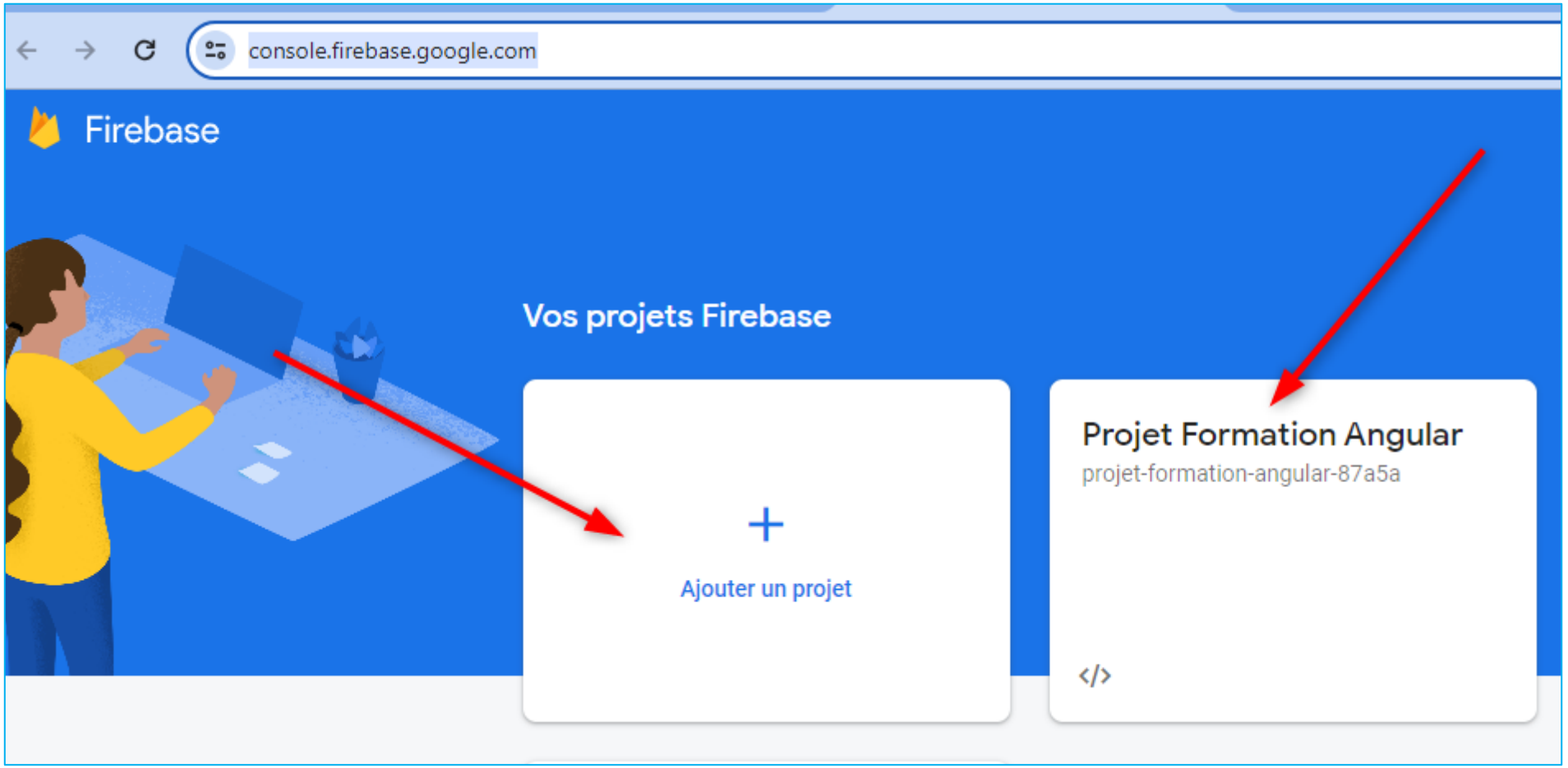
FireBase – Ecosystème BackEnd



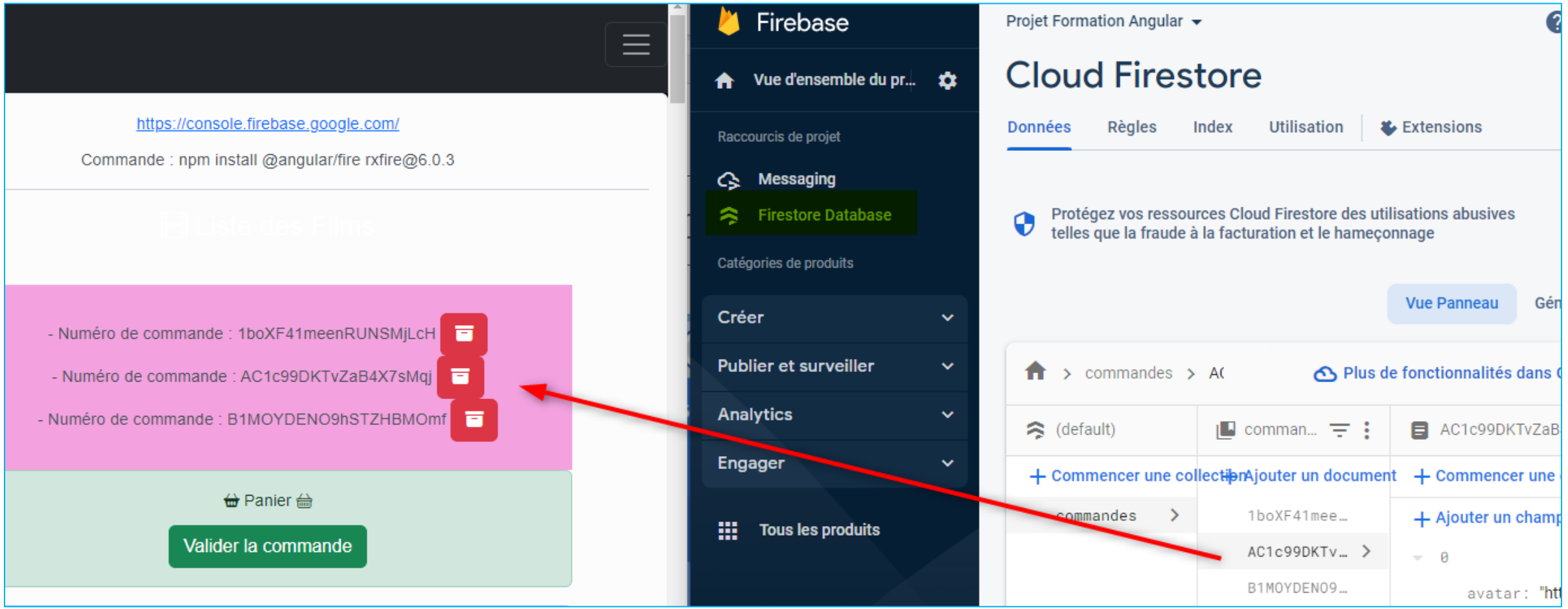
FireBase – Base de données en temps réel

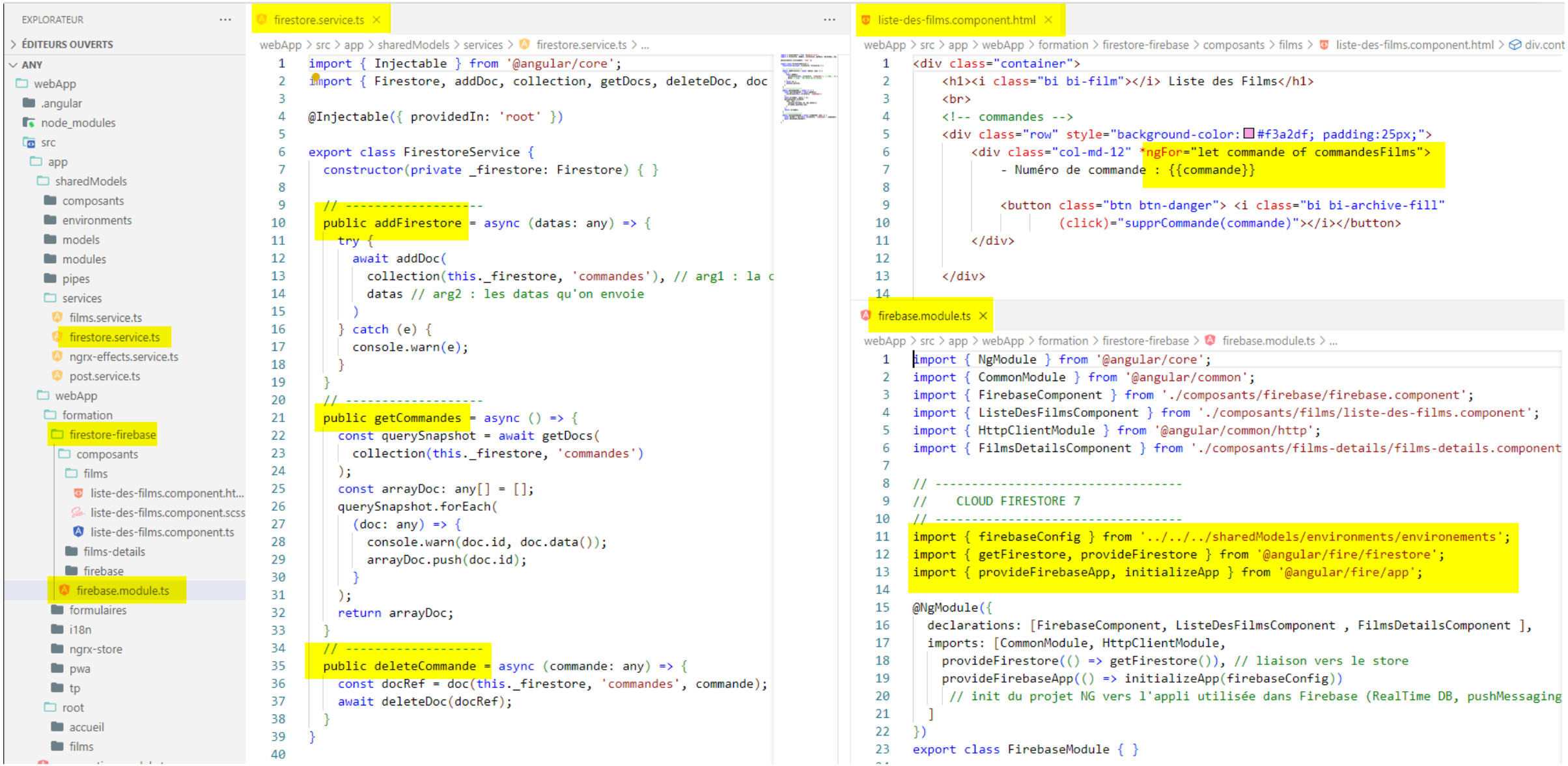


https://console.firebase.google.com



https://console.firebase.google.com



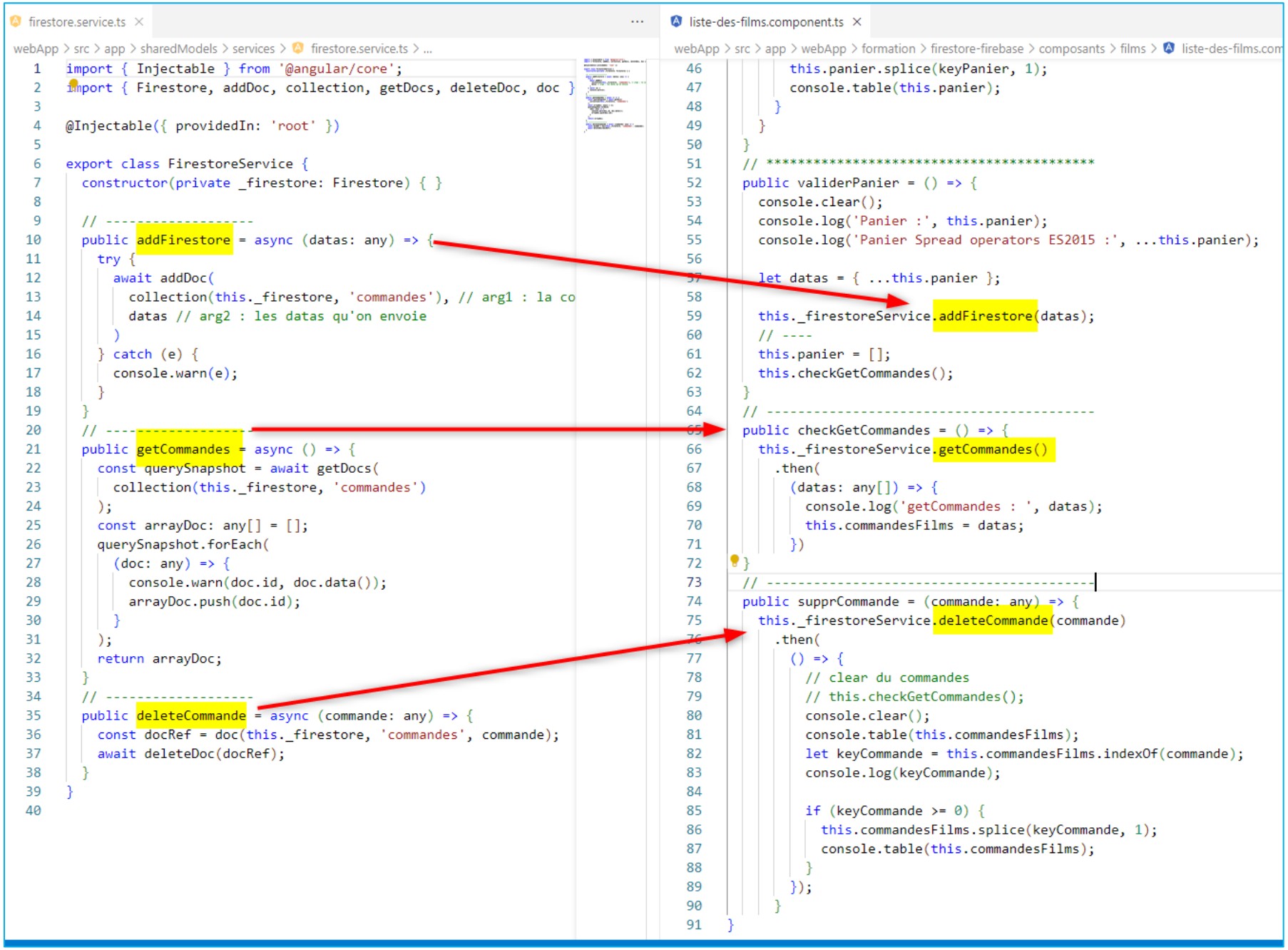

EXPLORATEUR
ÉDITEURS OUVERTS
ANY
webApp
.angular
node_modules
src
app
sharedModels
composants
environments
models
modules
pipes
services
films.service.ts
firestore.service.ts
ngrx-effects.service.ts
post.service.ts
webApp
formation
firestore-firebase
composants
films
liste-des-films.component.ht...
liste-des-films.component.scss
liste-des-films.component.ts
films-details
firebase
firebase.module.ts
formulaires
i18n
ngrx-store
pwa
tp
root
accueil
films
firestore.service.ts
webApp > src > app > sharedModels > services > firestore.service.ts > ...
import { Injectable } from '@angular/core';
import { Firestore, addDoc, collection, getDocs, deleteDoc, doc
@Injectable({ providedIn: 'root' })
export class FirestoreService {
constructor(private _firestore: Firestore) { }
// -------------------
public addFirestore = async (datas: any) => {
try {
await addDoc(
collection(this._firestore, 'commandes'), // arg1 : la c
datas // arg2 : les datas qu'on envoie
)
} catch (e) {
console.warn(e);
}
}
// -------------------
public getCommandes = async () => {
const querySnapshot = await getDocs(
collection(this._firestore, 'commandes')
);
const arrayDoc: any[] = [];
querySnapshot.forEach(
(doc: any) => {
console.warn(doc.id, doc.data());
arrayDoc.push(doc.id);
}
);
return arrayDoc;
}
// -------------------
public deleteCommande = async (commande: any) => {
const docRef = doc(this._firestore, 'commandes', commande);
await deleteDoc(docRef);
}
}
liste-des-films.component.html
webApp > src > app > webApp > formation > firestore-firebase > composants > films > liste-des-films.component.html > div.cont

<h1> Liste des Films</h1>

<!-- commandes -->

<div class="col-md-12" *ngFor="let commande of commandesFilms">
- Numéro de commande : {{commande}}
<button class="btn btn-danger"> <i class="bi bi-archive-fill"
(click)="supprCommande(commande)"></button>

firebase.module.ts
webApp > src > app > webApp > formation > firestore-firebase > firebase.module.ts > ...
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { FirebaseComponent } from './composants/firebase/firebase.component';
import { ListeDesFilmsComponent } from './composants/films/liste-des-films.component';
import { HttpClientModule } from '@angular/common/http';
import { FilmsDetailsComponent } from './composants/films-details/films-details.component
// ----------------------------------
// CLOUD FIRESTORE 7
// ----------------------------------
import { firebaseConfig } from '../../../sharedModels/environments/environements';
import { getFirestore, provideFirestore } from '@angular/fire/firestore';
import { provideFirebaseApp, initializeApp } from '@angular/fire/app';
@NgModule({
declarations: [FirebaseComponent, ListeDesFilmsComponent , FilmsDetailsComponent],
imports: [CommonModule, HttpClientModule,
provideFirestore(() => getFirestore()), // liaison vers le store
provideFirebaseApp(() => initializeApp(firebaseConfig))
// init du projet NG vers l'appli utilisée dans Firebase (RealTime DB, pushMessaging
]
})
export class FirebaseModule { }



firestore.service.ts

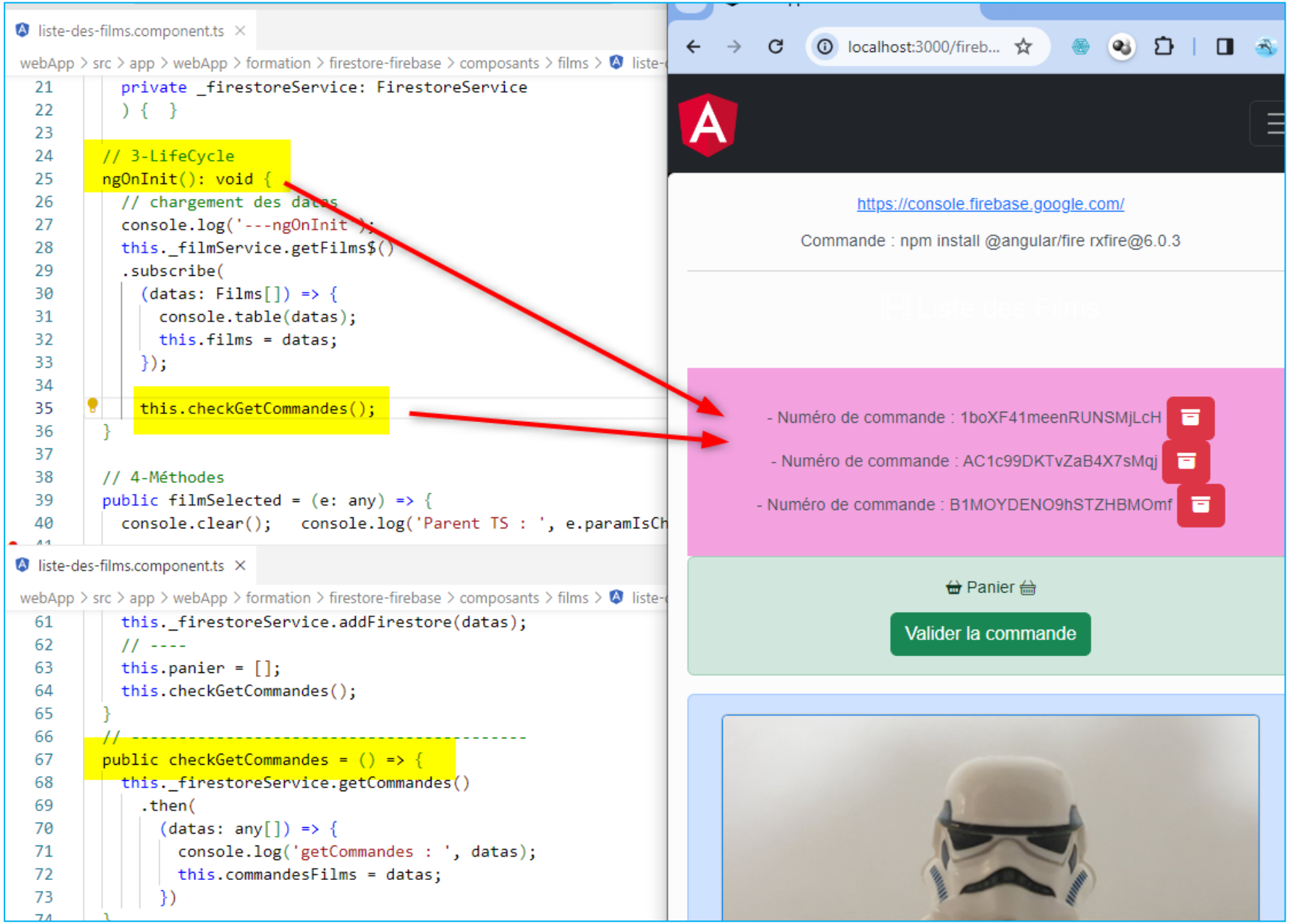
webApp > src > app > sharedModels > services > firestore.service.ts > ...

```ts
import { Injectable } from '@angular/core';
import { Firestore, addDoc, collection, getDocs, deleteDoc, doc }

@Injectable({ providedIn: 'root' })

export class FirestoreService {
  constructor(private _firestore: Firestore) { }

  // --------------------
  public addFirestore = async (datas: any) => {
    try {
      await addDoc(
        collection(this._firestore, 'commandes'), // arg1 : la co
        datas // arg2 : les datas qu'on envoie
      )
    } catch (e) {
      console.warn(e);
    }
  }
  // --------------------
  public getCommandes = async () => {
    const querySnapshot = await getDocs(
      collection(this._firestore, 'commandes')
    );
    const arrayDoc: any[] = [];
    querySnapshot.forEach(
      (doc: any) => {
        console.warn(doc.id, doc.data());
        arrayDoc.push(doc.id);
      }
    );
    return arrayDoc;
  }
  // --------------------
  public deleteCommande = async (commande: any) => {
    const docRef = doc(this._firestore, 'commandes', commande);
    await deleteDoc(docRef);
  }
}

```

liste-des-films.component.ts

webApp > src > app > webApp > formation > firestore-firebase > composants > films > liste-des-films.com

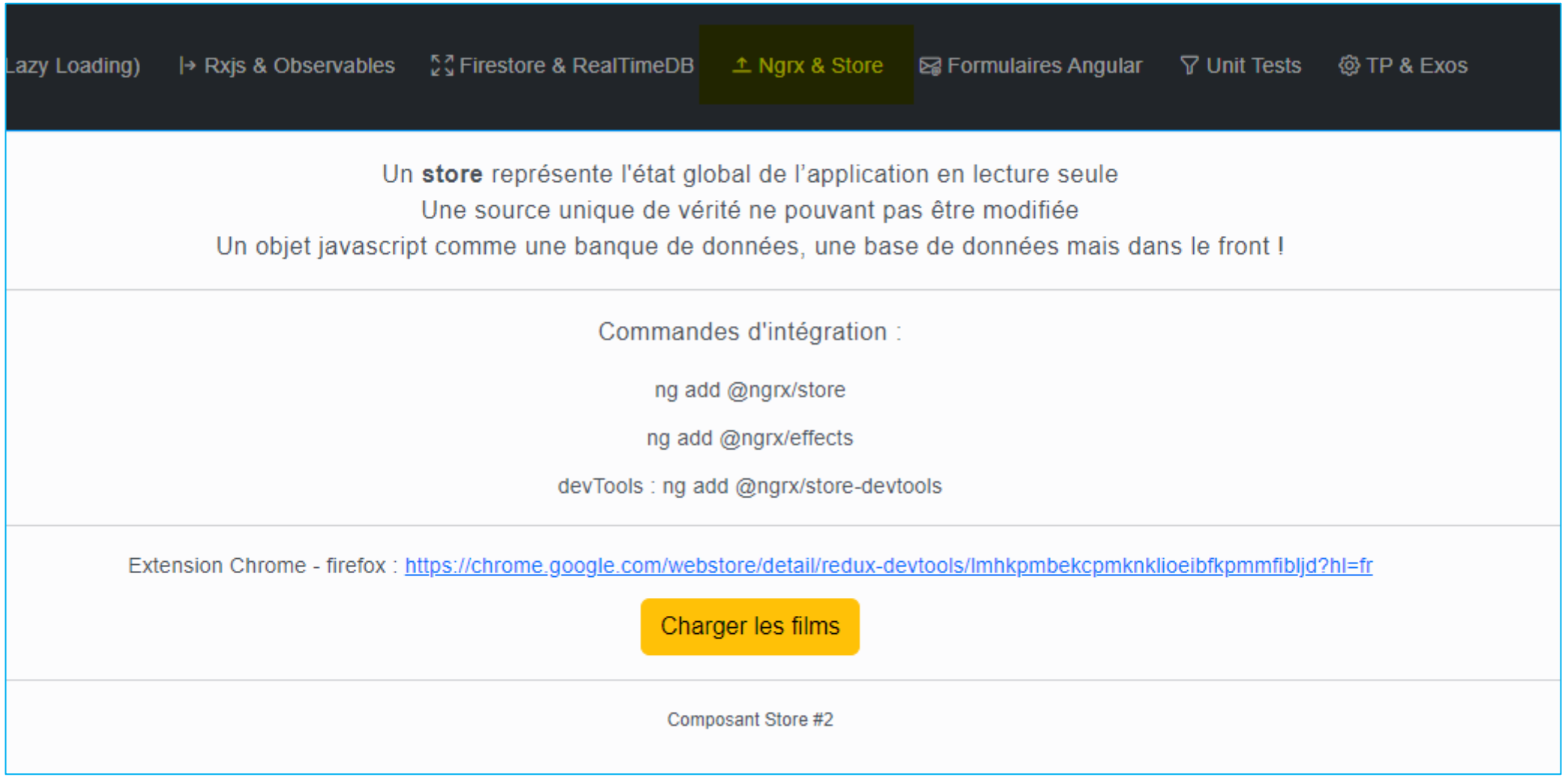
```ts
        this.panier.splice(keyPanier, 1);
        console.table(this.panier);
      }
    }
  }
  // ******************************************
  public validerPanier = () => {
    console.clear();
    console.log('Panier :', this.panier);
    console.log('Panier Spread operators ES2015 :', ...this.panier);

    let datas = { ...this.panier };

    this._firestoreService.addFirestore(datas);
    // ----
    this.panier = [];
    this.checkGetCommandes();
  }
  // ------------------------------------------
  public checkGetCommandes = () => {
    this._firestoreService.getCommandes()
      .then(
        (datas: any[]) => {
          console.log('getCommandes : ', datas);
          this.commandesFilms = datas;
        })
  }
  // ------------------------------------------
  public supprCommande = (commande: any) => {
    this._firestoreService.deleteCommande(commande)
      .then(
        () => {
          // clear du commandes
          // this.checkGetCommandes();
          console.clear();
          console.table(this.commandesFilms);
          let keyCommande = this.commandesFilms.indexOf(commande);
          console.log(keyCommande);

          if (keyCommande >= 0) {
            this.commandesFilms.splice(keyCommande, 1);
            console.table(this.commandesFilms);
          }
        });
  }
}
```


liste-des-films.component.ts
webApp > src > app > webApp > formation > firestore-firebase > composants > films > liste-
21 private _firestoreService: FirestoreService
22) { }
23
24 // 3-LifeCycle
25 ngOnInit(): void {
26 // chargement des datas
27 console.log('---ngOnInit');
28 this._filmService.getFilms$()
29 .subscribe(
30 (datas: Films[]) => {
31 console.table(datas);
32 this.films = datas;
33 });
34
35 this.checkGetCommandes();
36 }
37
38 // 4-Méthodes
39 public filmSelected = (e: any) => {
40 console.clear(); console.log('Parent TS : ', e.paramIsCh
liste-des-films.component.ts
webApp > src > app > webApp > formation > firestore-firebase > composants > films > liste-
61 this._firestoreService.addFirestore(datas);
62 // ----
63 this.panier = [];
64 this.checkGetCommandes();
65 }
66 // --
67 public checkGetCommandes = () => {
68 this._firestoreService.getCommandes()
69 .then(
70 (datas: any[]) => {
71 console.log('getCommandes : ', datas);
72 this.commandesFilms = datas;
73 })
localhost:3000/fireb...
https://console.firebase.google.com/
Commande : npm install @angular/fire rxfire@6.0.3
Liste des films
- Numéro de commande : 1boXF41meenRUNSMjLcH
- Numéro de commande : AC1c99DKTvZaB4X7sMqj
- Numéro de commande : B1MOYDENO9hSTZHBMOmf
Panier
Valider la commande

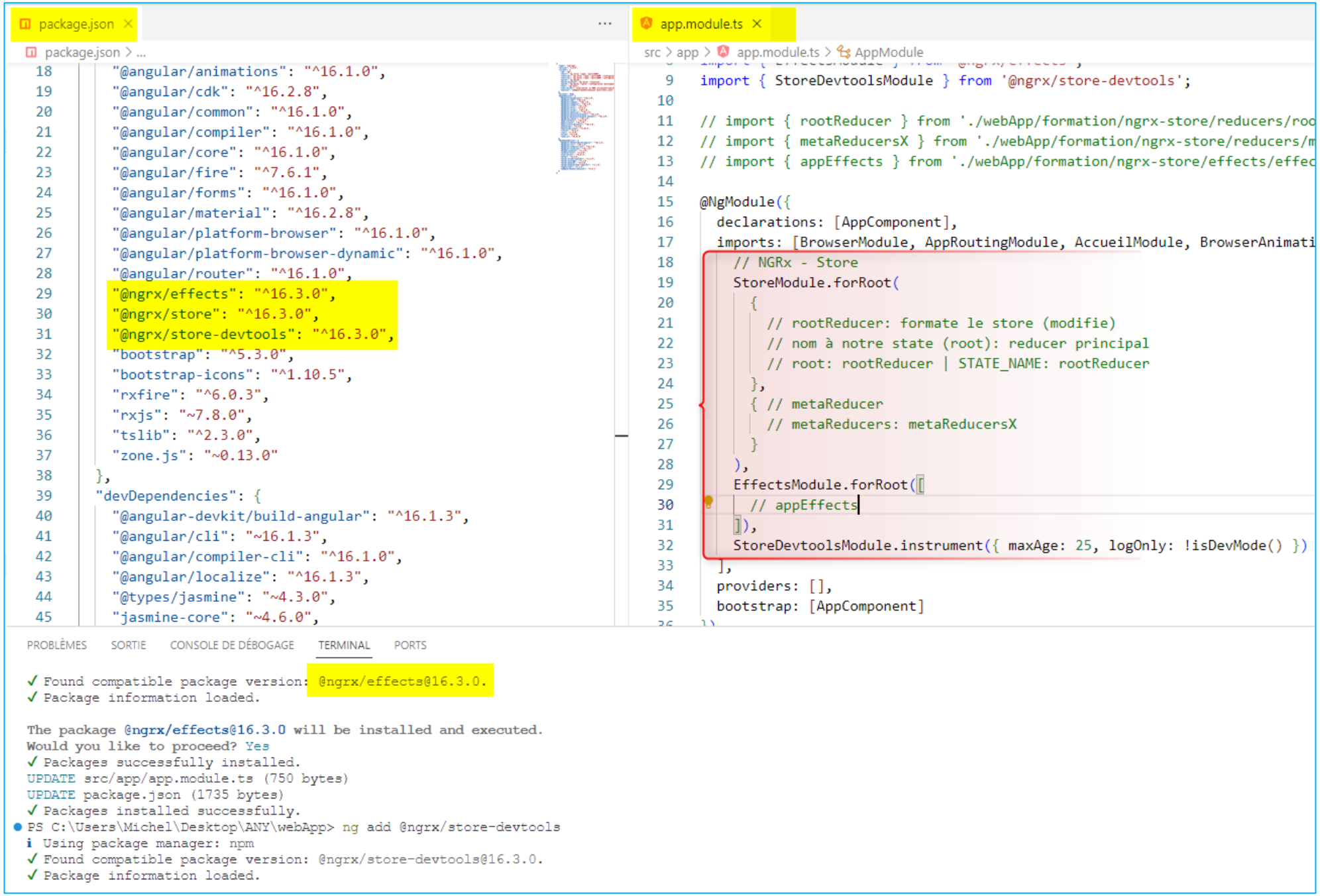
NGRX – La notion de STORE

aNGular – Redux - rXjs

NGRX


Lazy Loading)
Rxjs & Observables
Firestore & RealTimeDB
Ngrx & Store
Formulaires Angular
Unit Tests
TP & Exos
Un **store** représente l'état global de l'application en lecture seule
Une source unique de vérité ne pouvant pas être modifiée
Un objet javascript comme une banque de données, une base de données mais dans le front !
Commandes d'intégration :
ng add @ngrx/store
ng add @ngrx/effects
devTools : ng add @ngrx/store-devtools
Extension Chrome - firefox : https://chrome.google.com/webstore/detail/redux-devtools/lmhkpmbekcpmknklioeibfkpmmfibljd?hl=fr
Charger les films
Composant Store #2

Installation



NGRX pourquoi ?

Lorsqu'il y a beaucoup d'interactions entre composants et composants-enfants,
Lorsque le projet devient très conséquent en termes de développement,
Lorsque le projet grandit et se construit au fur et à mesure …

Le projet est de plus en plus difficile à maintenir.
On peut rencontrer des effets de bords non désirés.
La structure « complexe » du projet rend de plus en plus difficile les opérations d'amélioration et d'enrichissement.

BEST NOUVELLE EDITION

Formation : Angular, maîtriser le Framework Front-End de Google

concepts de développement

4,5 / 5

Angular est le framework javascript de référence de Google. Il utilise tous les standards du Web. Il offre des performances accrues avec une conception modulaire adaptée à la mobilité ainsi qu'une amélioration de la productivité de vos équipes de développement. Angular bénéficie immédiatement d'un écosystème riche et d'une communauté toujours plus grande.

Objectifs pédagogiques

À l'issue de la formation, le participant sera en mesure de :

- Organiser, modulariser et tester ses développements JavaScript

Inter Intra Sur mesure

Cours pratique en présentiel ou en classe à distance

Réf. AGU

4j - 28h00

Pauses-café et déjeuners offerts

Dates, lieux et Inscription

Nous contacter

> Formations > Technologies numériques > Technologies Web > Développement Front-End > Formation Angular, développement avancé

BEST

Formation : Angular, développement avancé

4,3 / 5

Vous découvrirez en profondeur les bonnes pratiques de développement des applications Angular avec les dernières version du framework Angular et le moteur de rendu optimisé Ivy. Vous apprendrez à maitriser le FormBuilder pour des formulaires réactifs ainsi que la génération de tests unitaires.

Objectifs pédagogiques

À l'issue de la formation, le participant sera en mesure de :

- Savoir utiliser les décorateurs Angular
- Architecturer les applications web complexes
- Intégrer les outils de documentation et les tests unitaires
- Développer et intégrer des librairies de composants

Inter Intra Sur mesure

Cours pratique en présentiel ou en classe à distance

Réf. ANY

3j - 21

Pauses-café et déjeuners offerts

Dates, lieux et Inscription

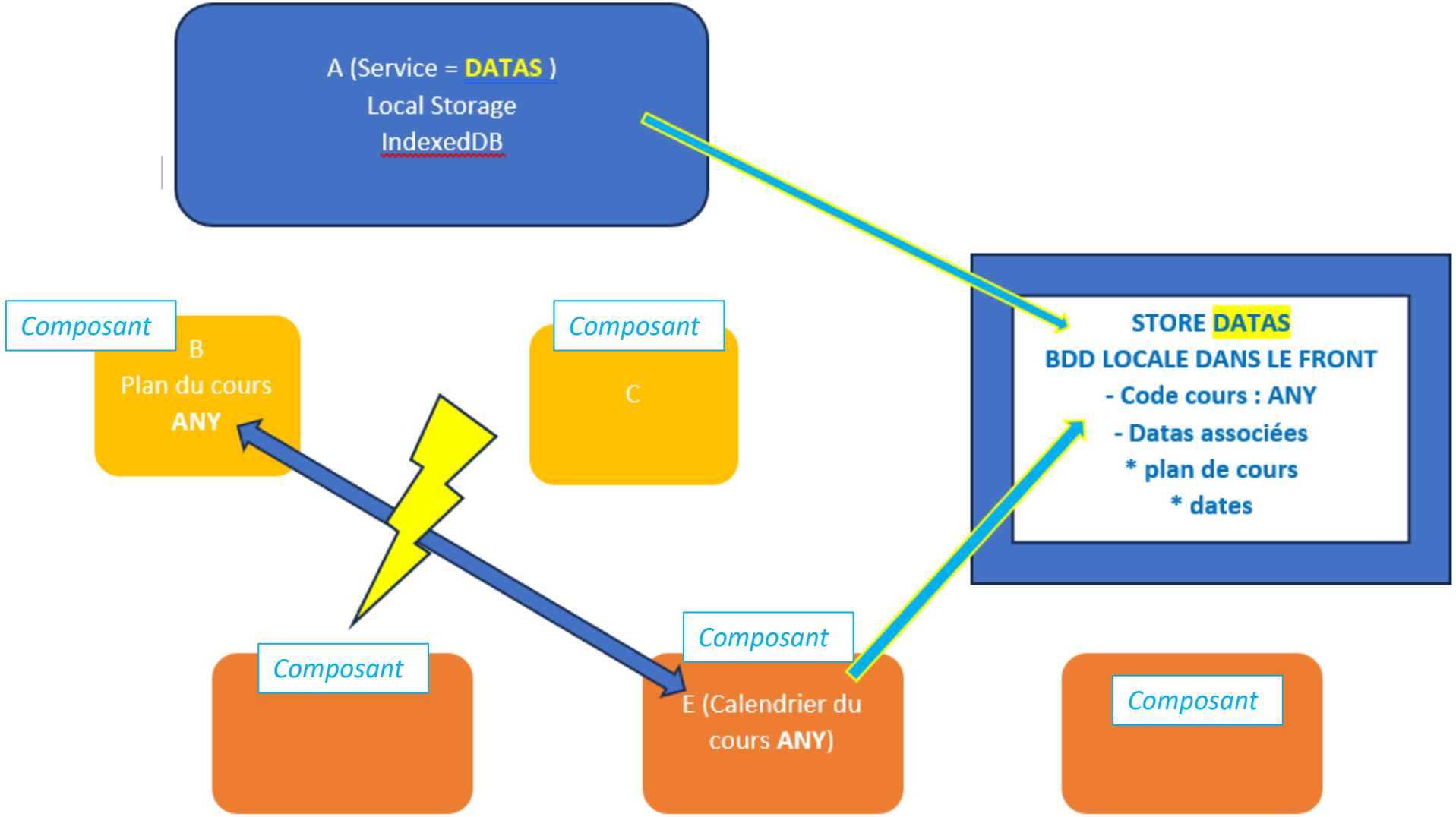
Nous contacter

NGRX la solution

Au lieu de traverser tout l'héritage des Input et des Output des composants parents-enfants, le composant peut s'inscrire auprès du Store et dispatcher une Action.
C'est-à-dire demander quelque chose ou demander à faire quelque chose.

L'accès est direct sans passer par la structure pargois « trop rigide » des web-components imbriqués.

NGRX

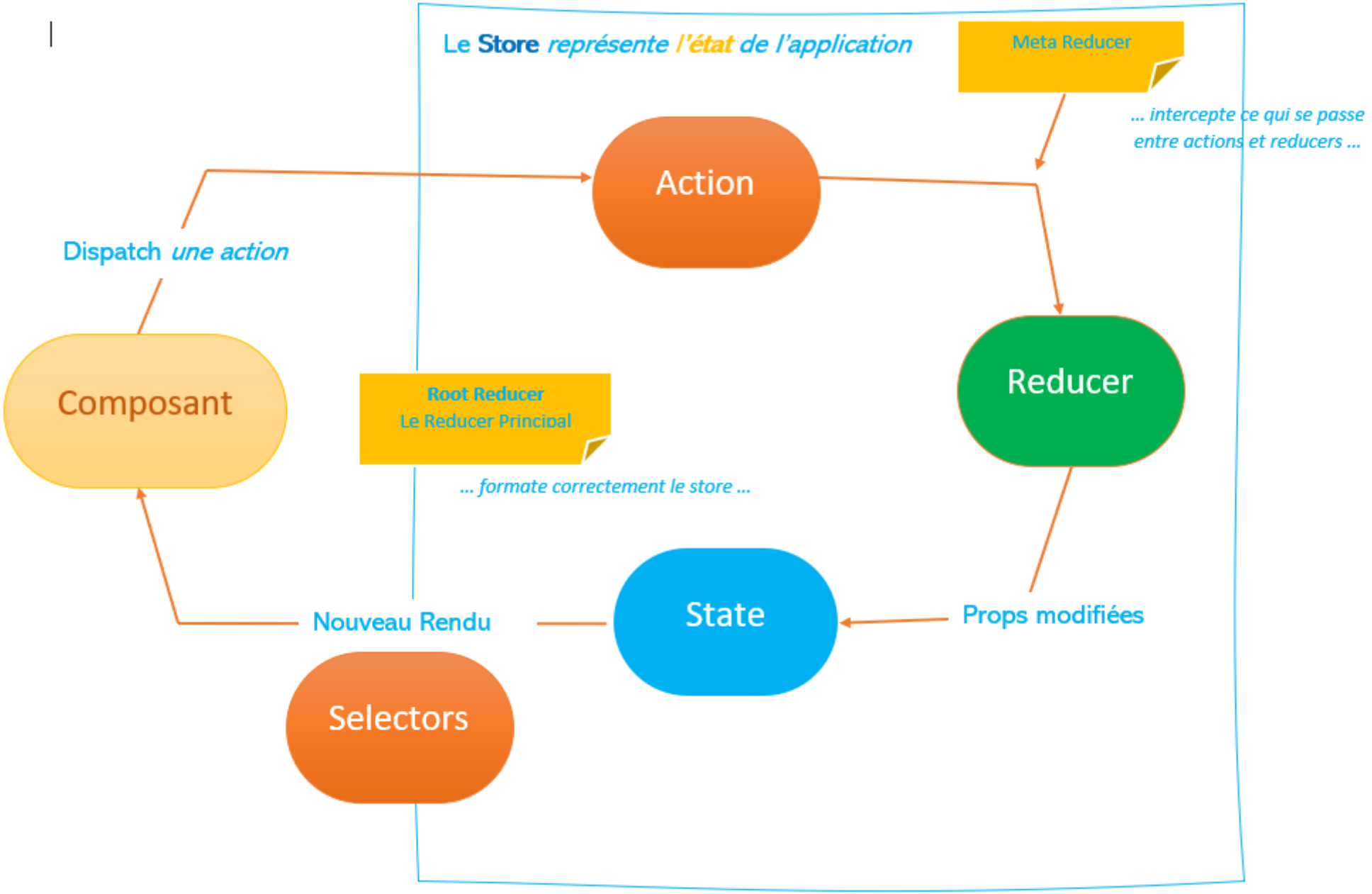


1 Composant s'inscrit auprès du STORE !
Dispatcher des actions (demande à faire qq chose avec les datas enregistrées dans le store)

NGRX Principe de fonctionnement

1- Le composant dispatche une action
2- Celle-ci doit être listée dans le store
3- Un reducer réceptionne l'action « dispatchée » et se charge de son éxécution
4- Le reducer modifie le state de l'application
5- Le composant dispatchant l'action demande à accéder à une partie du store grâce aux selectors
6- Un nouveau rendu HTML est proposé à tous les composants ayant accès à ce selector !

NGRX

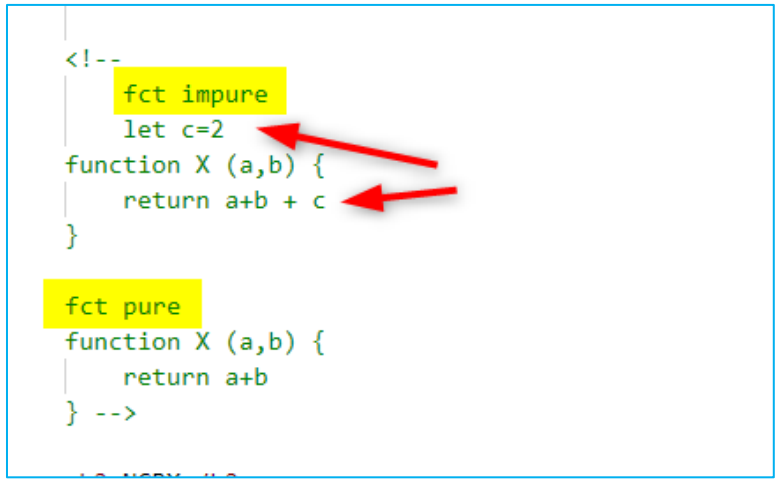

Le Store représente l'état de l'application
Meta Reducer
... intercepte ce qui se passe entre actions et reducers ...
Action
Dispatch une action
Composant
Root Reducer
Le Reducer Principal
Reducer
... formate correctement le store ...
State
Nouveau Rendu
Props modifiées
Selectors

NGRX SideEffect - BackEnd

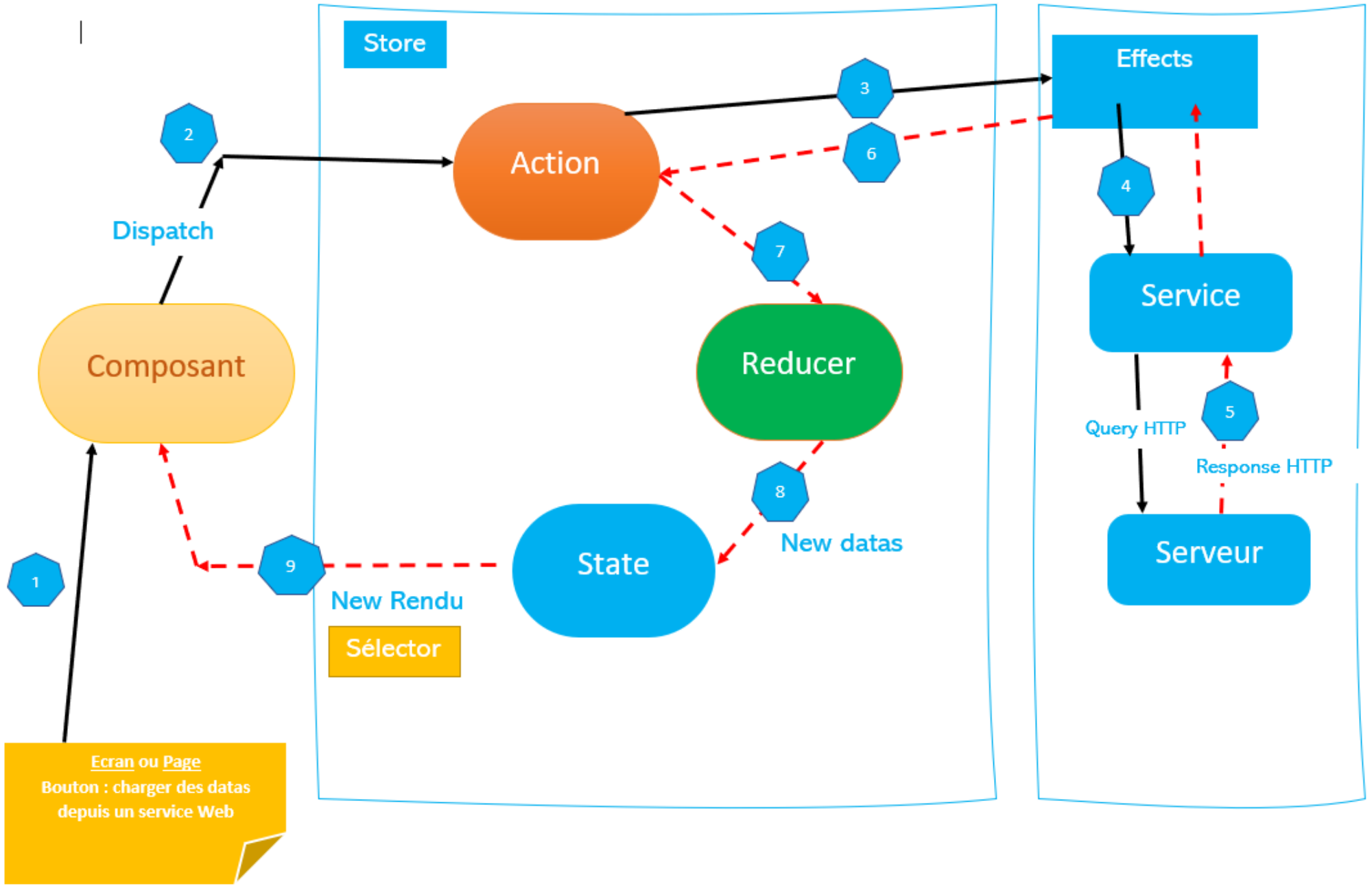
Les sideEffects permettent de connecter le back via des services web à notre application Angular-STORE.

Le schéma peut « paraître » complexe au 1er abord mais il est logique dans la conception de la « fonction pure » du Reducer !

Pour rappel, une fonction pure ne peut retourner que les paramètres reçus en appel (en entrée)



NGRX


Store
Effects
Action
Dispatch
Composant
Reducer
Service
Query HTTP
Response HTTP
Serveur
State
New datas
New Rendu
Sélector
Ecran ou Page
Bouton : charger des datas
depuis un service Web
1
2
3
4
5
6
7
8
9

Root Reducer

root-reducers.ts

Formation-ANY-07-2023 > TP07-NGRX > src > app > webApp > formation > ngrx-store > reducers > root-reducers.ts

```ts
import { createReducer, on } from "@ngrx/store";
import { changenameAction, initAction, loadFilmsAction, loadFilmsNotOkActi
import { Films } from "src/app/sharedModels/models/class/films";

// const INITIAL_STATE = {}; // à l'init => en prod
// ------------------- OPTIMISATION ------------------
// export const STATE_NAME = 'appli';
export interface rootState {
    appName: string;
    actor: {
        name: string;
        isJedi: boolean;
    }
    film?: Films;
    films?: Films[],
    loaded?: boolean
}
//-----------------------------------------------------
const INITIAL_STATE: rootState = {
    appName: 'Formation Angular NGRX',
    actor: {
        name: '',
        isJedi: false
    }
};
export const rootReducer = createReducer(
    // state initial
    INITIAL_STATE,
    // ons => 'on' sur une action
    // définir les actions invoquées
    on(
        initAction,
        // tâche ou le job que va faire le reducer ...
        // sur les props du store représenté par un state
        (state) => {
            // le reducer est capable de prendre qu'une partie du store(st
            return {
                ...state, // ... spred operators ES2015 (déstructure notre
                actor: {
                    ...state.actor, // copie de la valeur de actor
                    isJedi: true // param modifié
                }
            }
        }
    ),
```

root-reducers.ts

TP07-NGRX > src > app > webApp > formation > ngrx-store > reducers > root-reducers.ts

```ts
    // autre on
    on(
        changenameAction,
        (state, props) => {
            return {
                ...state,
                actor: {
                    ...state.actor,
                    name: props.paramNameActionCOMP
                }
            }
        }
    ),
    // autre on
    on(loadFilmsAction,
        (state) => {
            return {
                ...state,
                loaded: false
            }
        }
    ),
    // autre on
    on(loadFilmsOkAction,
        (state, props) => {
            return {
                ...state,
                films:props.films,
                loaded: true
            }
        }
    ),
    // autre on
    on(loadFilmsNotOkAction,
        (state, props) => {
            return {
                ...state,
                loaded: true,
                KO:true,
                erreurDesc:props.err
            }
        }
    )
);
```

Meta Reducer

root-reducers.ts | selectors.ts | actions.ts | effects.ts | meta-reducers.ts

Formation-ANY-07-2023 > TP07-NGRX > src > app > webApp > formation > ngrx-store > reducers > meta-reducers.ts > log > <fun

```ts
// affiche en console les différentes étapes
// qui se déroulent dans le store entre les actions et les reducers…

import { ActionReducer, MetaReducer } from "@ngrx/store"

// actionReducer représente la combinaison de l'action et du reducer
// nous donne le résultat de ce qui se passe entre l'action et le reducer

const log = (paramReducer:ActionReducer<any>) => {

    return(state:any, action:any) => {

        const currentState = paramReducer(state, action);

        console.groupCollapsed(action.type);

            console.log('Etat précédent : ', state);
            console.warn('Action : ', action);
            console.log('Etat suivant : ', currentState);

        console.groupEnd();

        return currentState;
    }
}
// ------------ export le metaReducer ---
export const metaReducersX: MetaReducer[] = [log];
```

selectors

root-reducers.ts | selectors.ts

tion-ANY-07-2023 > TP07-NGRX > src > app > webApp > formation > ngrx-store > selectors > sel

```ts
// on va créer des parties du store qu'on veut select
// pour le smettre à dispo des composants

import { createSelector } from "@ngrx/store";

const selectRootState =  (state:any) => {
    return state.root;
}

export const selectorGetActor =  createSelector(
    selectRootState,
    (state:any) => state.actor
)

export const selectorGetFilms = createSelector(
    // on prend une partie du state
    selectRootState, // a partir de quoi on va filtrer ce que l
    (state:any) => state.films
);
export const selectorGetLoaded = createSelector(
    selectRootState,
    (state:any) => state.loaded
);
export const selectorGetKO = createSelector(
    selectRootState,
    (state: any) => state.KO
  );
```

selectors

root-reducers.ts | selectors.ts

tion-ANY-07-2023 > TP07-NGRX > src > app > webApp > formation > ngrx-store > selectors > sel

```ts
// on va créer des parties du store qu'on veut select
// pour le smettre à dispo des composants

import { createSelector } from "@ngrx/store";

const selectRootState =  (state:any) => {
    return state.root;
}

export const selectorGetActor =  createSelector(
    selectRootState,
    (state:any) => state.actor
)

export const selectorGetFilms = createSelector(
    // on prend une partie du state
    selectRootState, // a partir de quoi on va filtrer ce que l
    (state:any) => state.films
);
export const selectorGetLoaded = createSelector(
    selectRootState,
    (state:any) => state.loaded
);
export const selectorGetKO = createSelector(
    selectRootState,
    (state: any) => state.KO
  );
```

actions

root-reducers.ts | selectors.ts | actions.ts

Formation-ANY-07-2023 > TP07-NGRX > src > app > webApp > formation > ngrx-store > actions > act

```ts
import { createAction, props } from "@ngrx/store";
import { Films } from "src/app/sharedModels/models/class/films";

export const initAction =  createAction(
    // nom de l'action
    '[ROOT] Init Action'
);

export const changenameAction =  createAction(
    // nom de l'action
    '[ROOT] Change Name Actor Action',
    // props à faire ajouter par le reducer dans le state
    props<{paramNameActionCOMP:string}>()
);

export const loadFilmsAction =  createAction(
    // nom de l'action
    '[FILMS] Ask Load Films Action'
);

export const loadFilmsOkAction = createAction(
    '[FILMS DATAS] Load films OK',
    props<{films:Films[]}>()
);

export const loadFilmsNotOkAction = createAction(
    '[FILMS DATAS] Load films Not OK',
    props<{err:any}>()
);
```

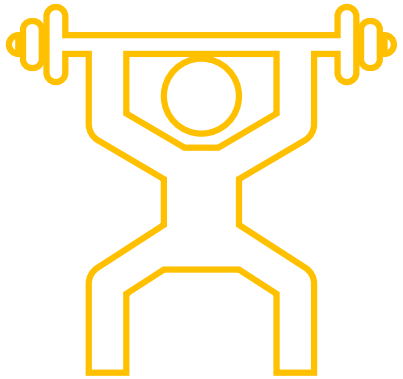
effects

root-reducers.ts | selectors.ts | actions.ts | effects.ts

Formation-ANY-07-2023 > TP07-NGRX > src > app > webApp > formation > ngrx-store > effects > effects.ts > appEffects > loadFilms$

```
6   import * as compActions from '../actions/actions';
7   import { Films } from "src/app/sharedModels/models/class/films";
8
9   @Injectable()
10
11  export class appEffects {
12
13
14      constructor(private _serviceEffects: NgrxEffectsService, private _actions: Actions) { }
15
16      // création d'un Observable
17
18      loadFilms$: Observable<any> = createEffect(
19          () => {
20              // détection des actions invoquées
21              return this._actions.pipe(
22                  tap(
23                      (action) => {
24                          console.log('******** Actions : ', action);
25                      }
26                  ),
27                  ofType(
28                      // ofType peut filtrer la liste de ce qu'on lui passe
29                      compActions.loadFilmsAction
30                  ),
31                  mergeMap(
32                      action => this._serviceEffects.getFilms$()
33                      // on récupère les datas : films
34                      .pipe(
35                          map(
36                              (films:Films[]) => {
37                                  return compActions.loadFilmsOkAction({films})
38                                  // c'est l'effect qui passe des datas à l'action
39                              }
40                          ),
41                          catchError(
42                              (err) => {
43                                  // console.log(err);
44                                  return of(compActions.loadFilmsNotOkAction({err}));
45
46                              }
47                          )
48                      )
49                  )
50              )
51          }
```

Les versions marquantes !



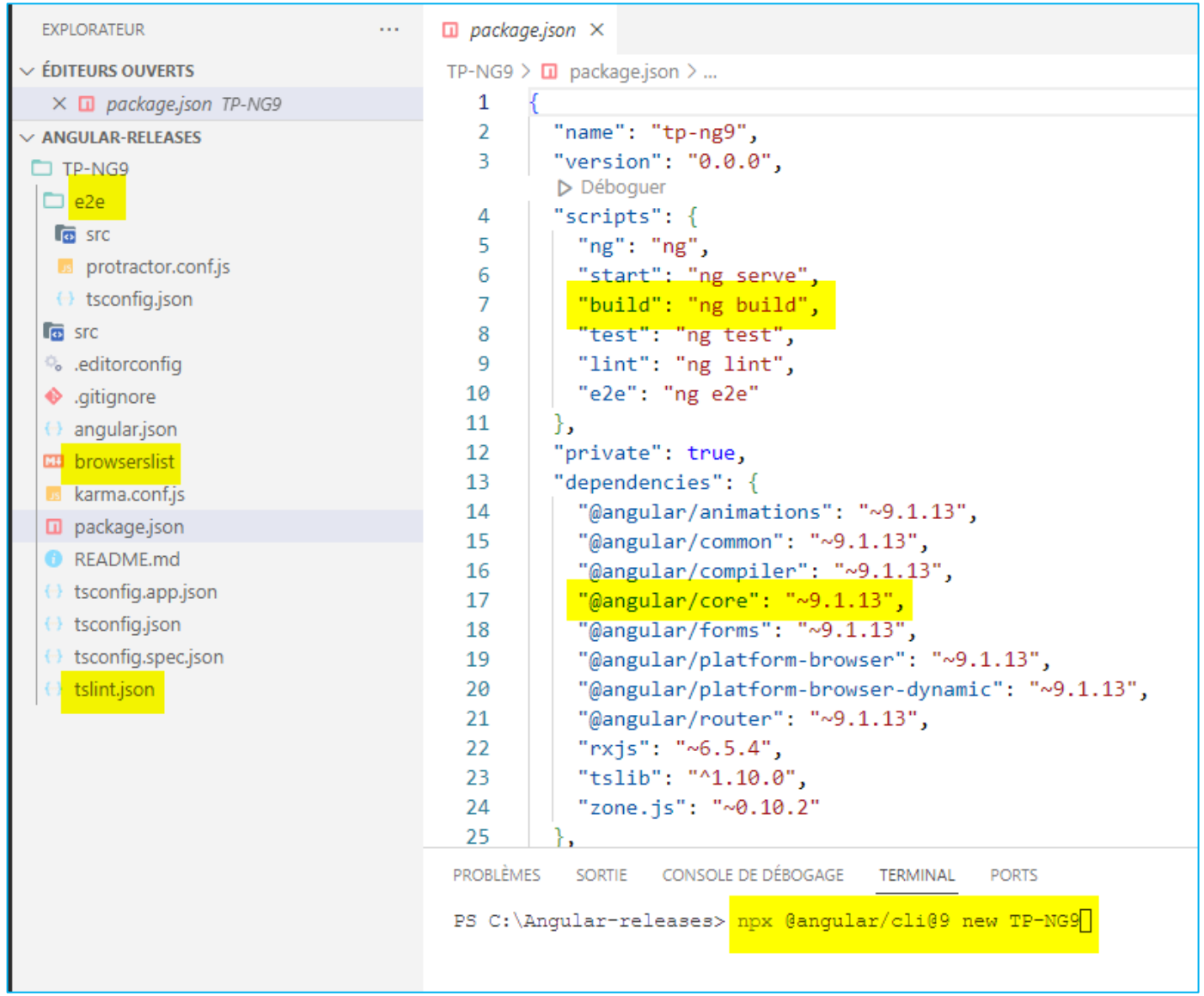
Les versions marquantes Angular

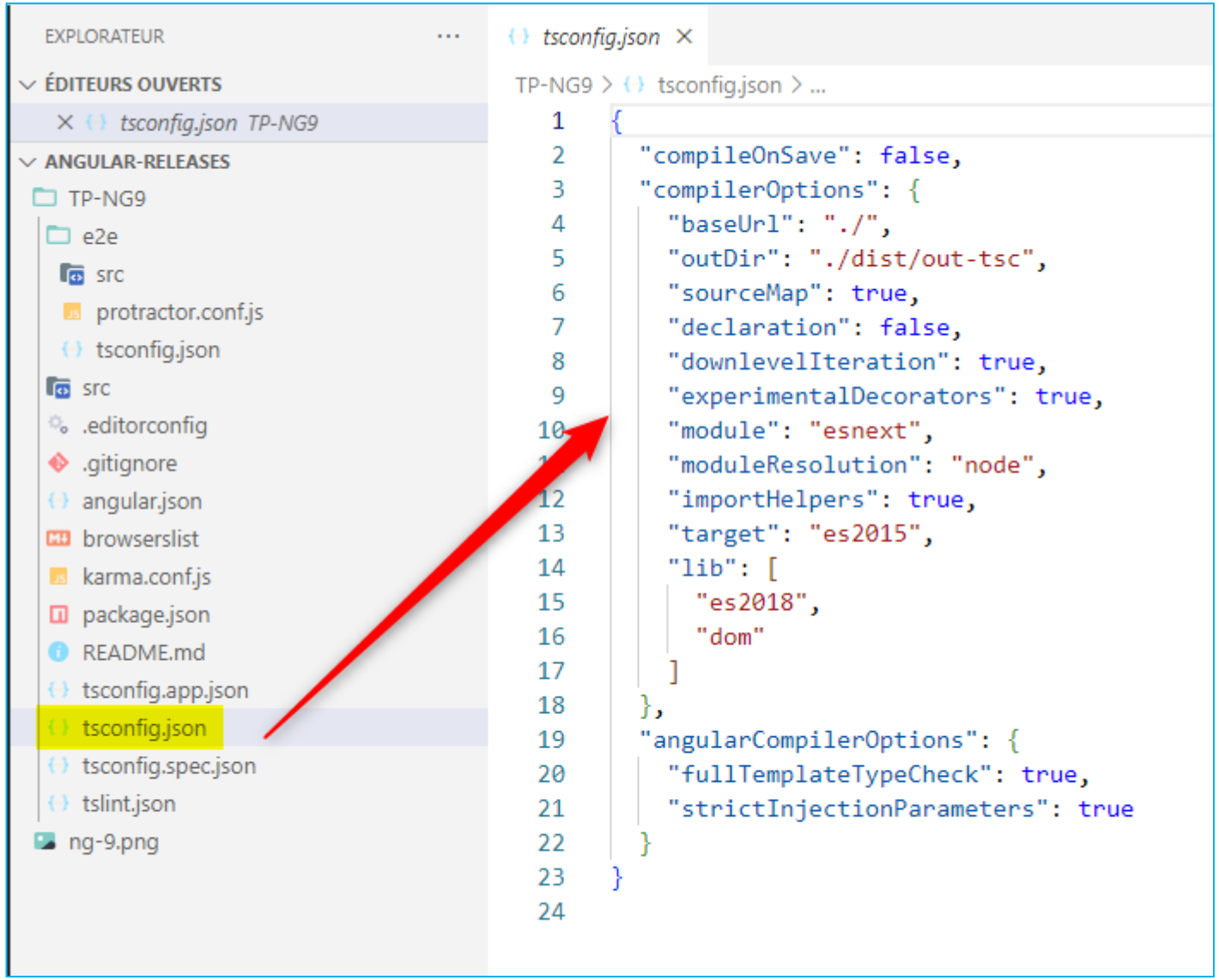
- Version 9 (6 février 2020)

  - Nouveau compiler et nouveau moteur de rendu Ivy
  - Réduction notable de la taille des bundles (build)
  - Optimisation de la compilation (temps, débug) AOT

  activé par défaut avec ng serve => remontée des erreurs en mode dév …

  - composant YouTube
  https://github.com/angular/components/tree/main/src/youtube-player
  - composant Google Maps
  https://github.com/angular/components/blob/main/src/google-maps/README.md


EXPLORATEUR
ÉDITEURS OUVERTS
package.json TP-NG9
ANGULAR-RELEASES
TP-NG9
e2e
src
protractor.conf.js
tsconfig.json
src
.editorconfig
.gitignore
angular.json
browserslist
karma.conf.js
package.json
README.md
tsconfig.app.json
tsconfig.json
tsconfig.spec.json
tslint.json
package.json
TP-NG9 > package.json > ...
{
"name": "tp-ng9",
"version": "0.0.0",
Déboguer
"scripts": {
"ng": "ng",
"start": "ng serve",
"build": "ng build",
"test": "ng test",
"lint": "ng lint",
"e2e": "ng e2e"
},
"private": true,
"dependencies": {
"@angular/animations": "~9.1.13",
"@angular/common": "~9.1.13",
"@angular/compiler": "~9.1.13",
"@angular/core": "~9.1.13",
"@angular/forms": "~9.1.13",
"@angular/platform-browser": "~9.1.13",
"@angular/platform-browser-dynamic": "~9.1.13",
"@angular/router": "~9.1.13",
"rxjs": "~6.5.4",
"tslib": "^1.10.0",
"zone.js": "~0.10.2"
},
PROBLÈMES
SORTIE
CONSOLE DE DÉBOGAGE
TERMINAL
PORTS
PS C:\Angular-releases> npx @angular/cli@9 new TP-NG9


EXPLORATEUR
ÉDITEURS OUVERTS
tsconfig.json TP-NG9
ANGULAR-RELEASES
TP-NG9
e2e
src
protractor.conf.js
tsconfig.json
src
.editorconfig
.gitignore
angular.json
browserslist
karma.conf.js
package.json
README.md
tsconfig.app.json
tsconfig.json
tsconfig.spec.json
tslint.json
ng-9.png
tsconfig.json
TP-NG9 > tsconfig.json > ...
{
"compileOnSave": false,
"compilerOptions": {
"baseUrl": "./",
"outDir": "./dist/out-tsc",
"sourceMap": true,
"declaration": false,
"downlevelIteration": true,
"experimentalDecorators": true,
"module": "esnext",
"moduleResolution": "node",
"importHelpers": true,
"target": "es2015",
"lib": [
"es2018",
"dom"
]
},
"angularCompilerOptions": {
"fullTemplateTypeCheck": true,
"strictInjectionParameters": true
}
}

Les versions marquantes Angular

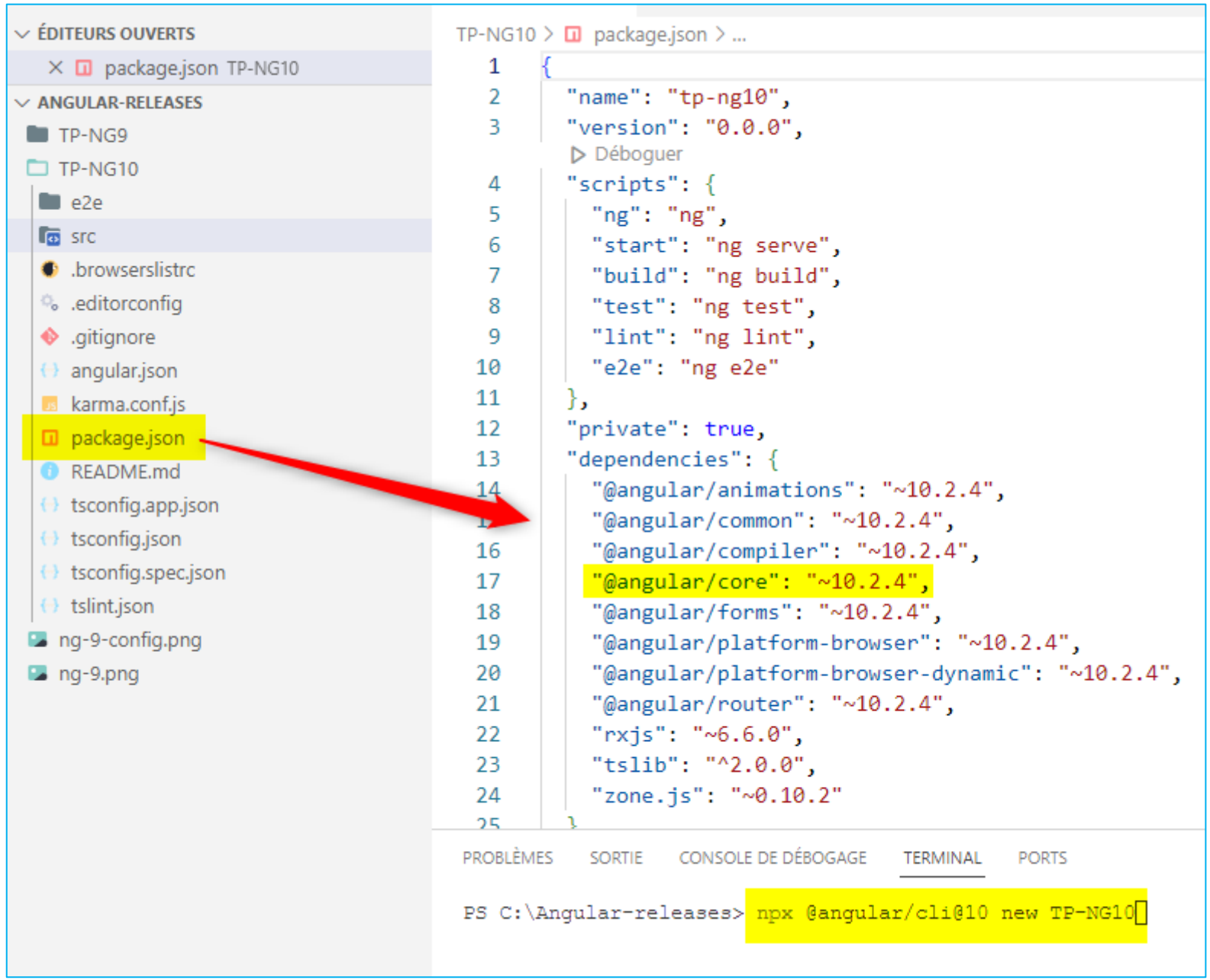
•Version 10 (24 juin 2020)

- Mode strict (typage obligatoire)
  ng new TP01 --strict

Rappels :

- basic : pas de vérification de types
- full : les props utilisées dans la Vue doivent être correctement définies dans le TS

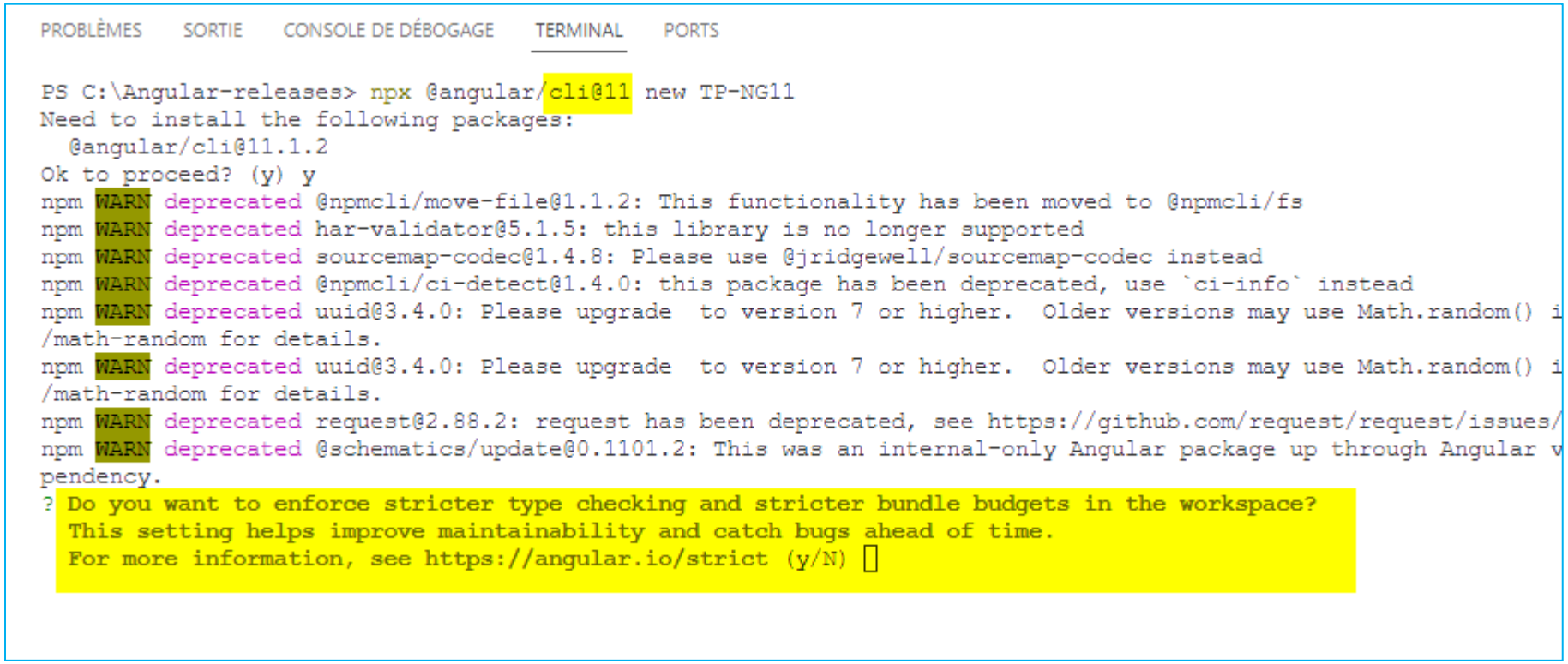
- nouveau datePicker dans Angular Matérial


ÉDITEURS OUVERTS
package.json TP-NG10
ANGULAR-RELEASES
TP-NG9
TP-NG10
e2e
src
.browserslistrc
.editorconfig
.gitignore
angular.json
karma.conf.js
package.json
README.md
tsconfig.app.json
tsconfig.json
tsconfig.spec.json
tslint.json
ng-9-config.png
ng-9.png
TP-NG10 > package.json > ...
{
"name": "tp-ng10",
"version": "0.0.0",
Déboguer
"scripts": {
"ng": "ng",
"start": "ng serve",
"build": "ng build",
"test": "ng test",
"lint": "ng lint",
"e2e": "ng e2e"
},
"private": true,
"dependencies": {
"@angular/animations": "~10.2.4",
"@angular/common": "~10.2.4",
"@angular/compiler": "~10.2.4",
"@angular/core": "~10.2.4",
"@angular/forms": "~10.2.4",
"@angular/platform-browser": "~10.2.4",
"@angular/platform-browser-dynamic": "~10.2.4",
"@angular/router": "~10.2.4",
"rxjs": "~6.6.0",
"tslib": "^2.0.0",
"zone.js": "~0.10.2"
}
PROBLÈMES
SORTIE
CONSOLE DE DÉBOGAGE
TERMINAL
PORTS
PS C:\Angular-releases> npx @angular/cli@10 new TP-NG10

Les versions marquantes Angular

•Version 11 (17 novembre 2020)

- Mode strict est proposé lors de la création de l'appli



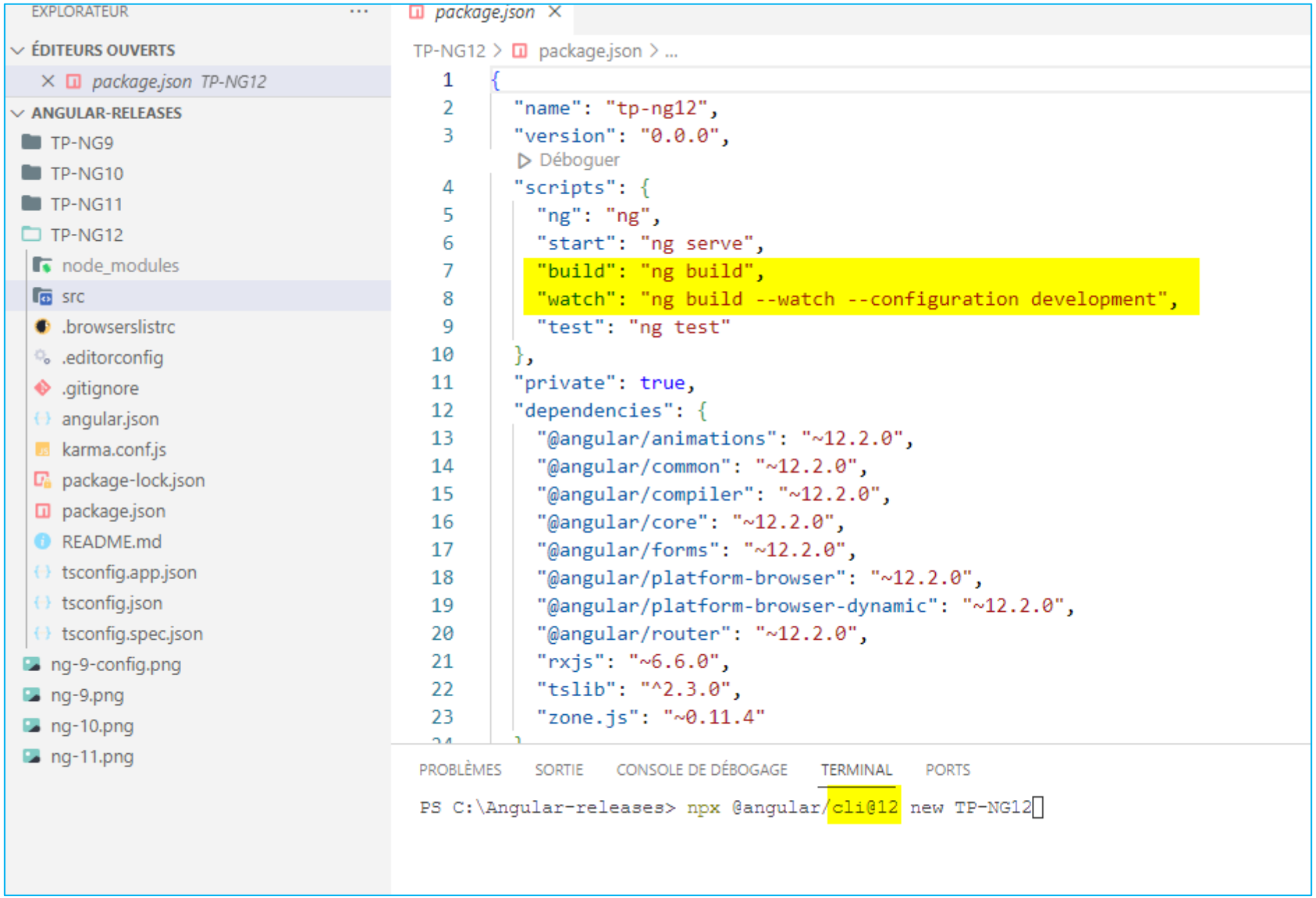
Les versions marquantes Angular

•Version 12 (19 mai 2021)

- Abandon progressif de la compatibilité et support IE

validé avec NG13

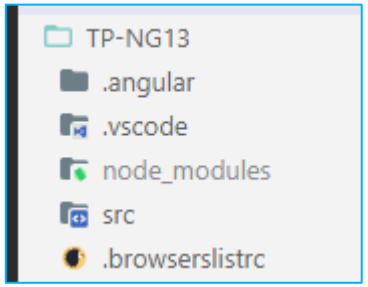
- Fin du support de e2e, protractor..
- TSlint déprécié depuis 2019 est remplacé par ESlint
- script watch qui remplace et améliore build
- build est désormais le mode prod


EXPLORATEUR
ÉDITEURS OUVERTS
package.json TP-NG12
ANGULAR-RELEASES
TP-NG9
TP-NG10
TP-NG11
TP-NG12
node_modules
src
.browserslistrc
.editorconfig
.gitignore
angular.json
karma.conf.js
package-lock.json
package.json
README.md
tsconfig.app.json
tsconfig.json
tsconfig.spec.json
ng-9-config.png
ng-9.png
ng-10.png
ng-11.png
package.json
TP-NG12 > package.json > ...
{
"name": "tp-ng12",
"version": "0.0.0",
Déboguer
"scripts": {
"ng": "ng",
"start": "ng serve",
"build": "ng build",
"watch": "ng build --watch --configuration development",
"test": "ng test"
},
"private": true,
"dependencies": {
"@angular/animations": "~12.2.0",
"@angular/common": "~12.2.0",
"@angular/compiler": "~12.2.0",
"@angular/core": "~12.2.0",
"@angular/forms": "~12.2.0",
"@angular/platform-browser": "~12.2.0",
"@angular/platform-browser-dynamic": "~12.2.0",
"@angular/router": "~12.2.0",
"rxjs": "~6.6.0",
"tslib": "^2.3.0",
"zone.js": "~0.11.4"
PROBLÈMES
SORTIE
CONSOLE DE DÉBOGAGE
TERMINAL
PORTS
PS C:\Angular-releases> npx @angular/cli@12 new TP-NG12

Les versions marquantes Angular

•Version 13 (3 novembre 2021)

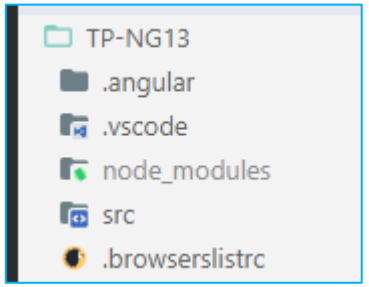
- Arrivée du cache .angular => accélération des temps de compilation
- démarrage en mode dév plus rapide
- Travail continuel d'optimisation de IVY et ViewEngine



Les versions marquantes Angular

•Version 14 (11 juin 2022)

- FormControl enfin typé
- composants standalone (*validé avec NG15*)
- optimisation des messages d'erreurs
- erreur banana in a box [()] ou ([])



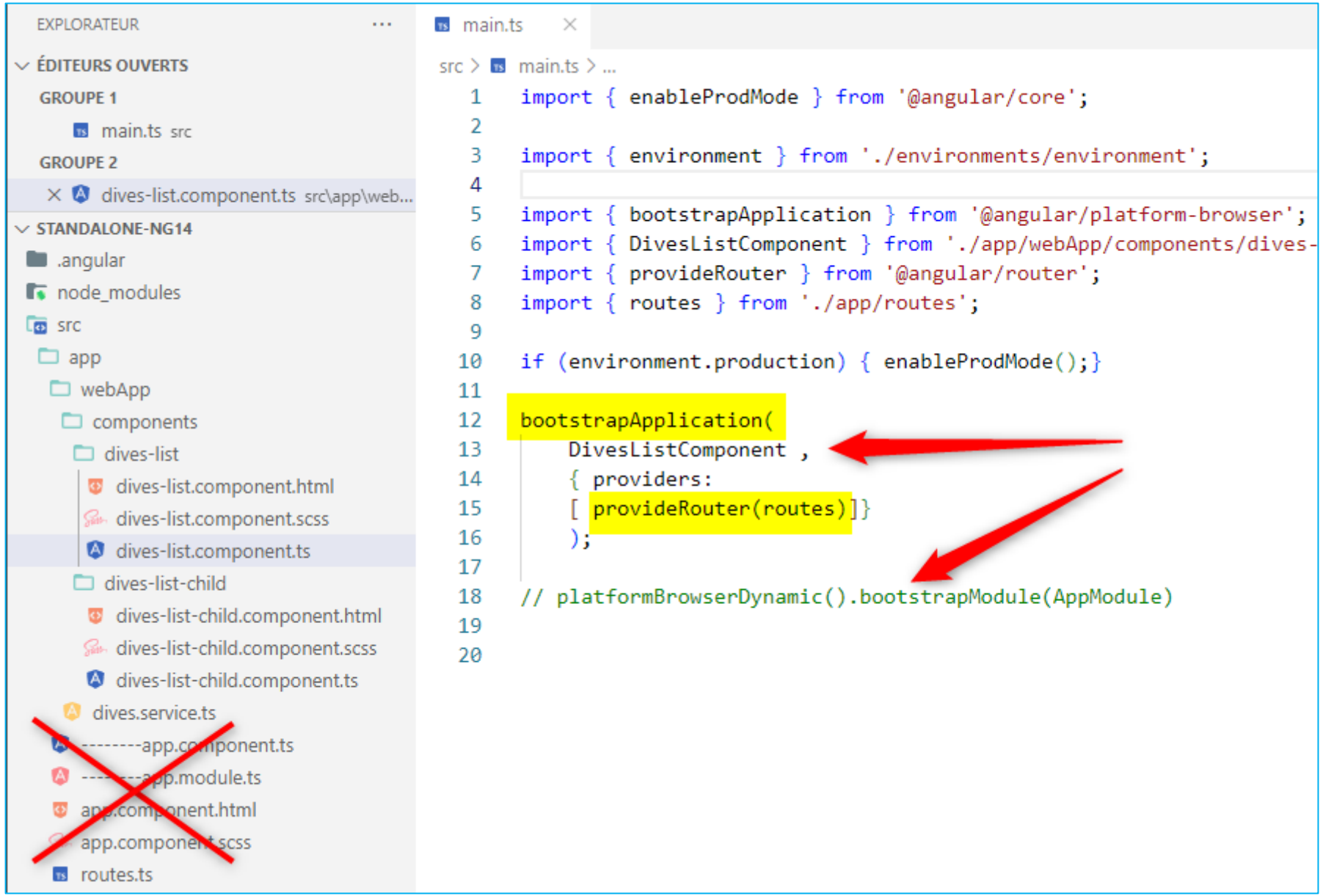
Les versions marquantes Angular

•Version 15 (16 novembre 2022)

- composants standalone

  Prop : standalone:true
- Omission des ngModules ?
- API autonomes ?

```
@Component({
  selector: 'app-root',
  // **********************
  standalone: true,
  templateUrl: './dives-list.component.html',
  styleUrls: ['./dives-list.component.scss'],
  // **********************
  imports: [
    // les autres composants doivent être également en component standalone !!
    DivesListChildComponent,
    // les pipes - les directives
    FilterPipe,
    // les modules
    CommonModule, HttpClientModule, RouterModule, FormsModule
  ],
  providers:[
    DivesService
  ]
})
```

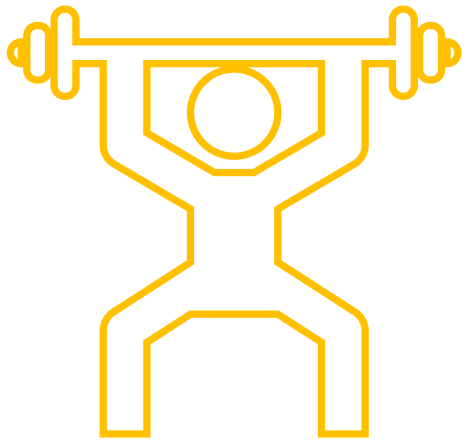

EXPLORATEUR
ÉDITEURS OUVERTS
GROUPE 1
main.ts src
GROUPE 2
dives-list.component.ts src\app\web...
STANDALONE-NG14
.angular
node_modules
src
app
webApp
components
dives-list
dives-list.component.html
dives-list.component.scss
dives-list.component.ts
dives-list-child
dives-list-child.component.html
dives-list-child.component.scss
dives-list-child.component.ts
dives.service.ts
app.component.ts
app.module.ts
app.component.html
app.component.scss
routes.ts
main.ts
src > main.ts > ...
1 import { enableProdMode } from '@angular/core';
2
3 import { environment } from './environments/environment';
4
5 import { bootstrapApplication } from '@angular/platform-browser';
6 import { DivesListComponent } from './app/webApp/components/dives-
7 import { provideRouter } from '@angular/router';
8 import { routes } from './app/routes';
9
10 if (environment.production) { enableProdMode();}
11
12 bootstrapApplication(
13 DivesListComponent ,
14 { providers:
15 [provideRouter(routes)]}
16);
17
18 // platformBrowserDynamic().bootstrapModule(AppModule)
19
20

Les versions marquantes Angular

- Version 16 (12 mai 2023)

- Les Signals
- Change Detection Change
- Encore besoin de Zone.JS ?
- Avenir de RXJS ?

Le Detection Change Mode
Nouveautés NG16 : Signal



Le Detection Change Mode Angular

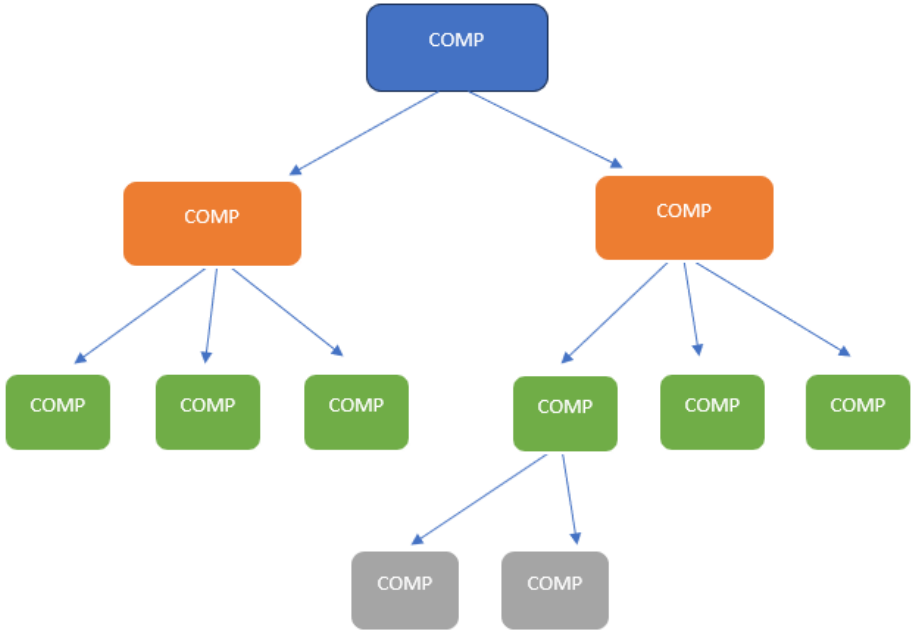
Lorsque des propriétés d'entrée (Input) ou des événements d'entrée sont déclenchés, Angular met à jour l'arbre des composants pour détecter d'éventuels changements à apporter aux affichages des Vues.

Modifier le comportement par défaut du « Change Detection Mode » d'Angular peut être utile lorsque des composants « rendent » (affichent) des données qui ne changent pas souvent.

Le nombre de cycles de change detection sera réduit et les performances de l'application en seront augmentées.

Le Detection Change Mode Angular

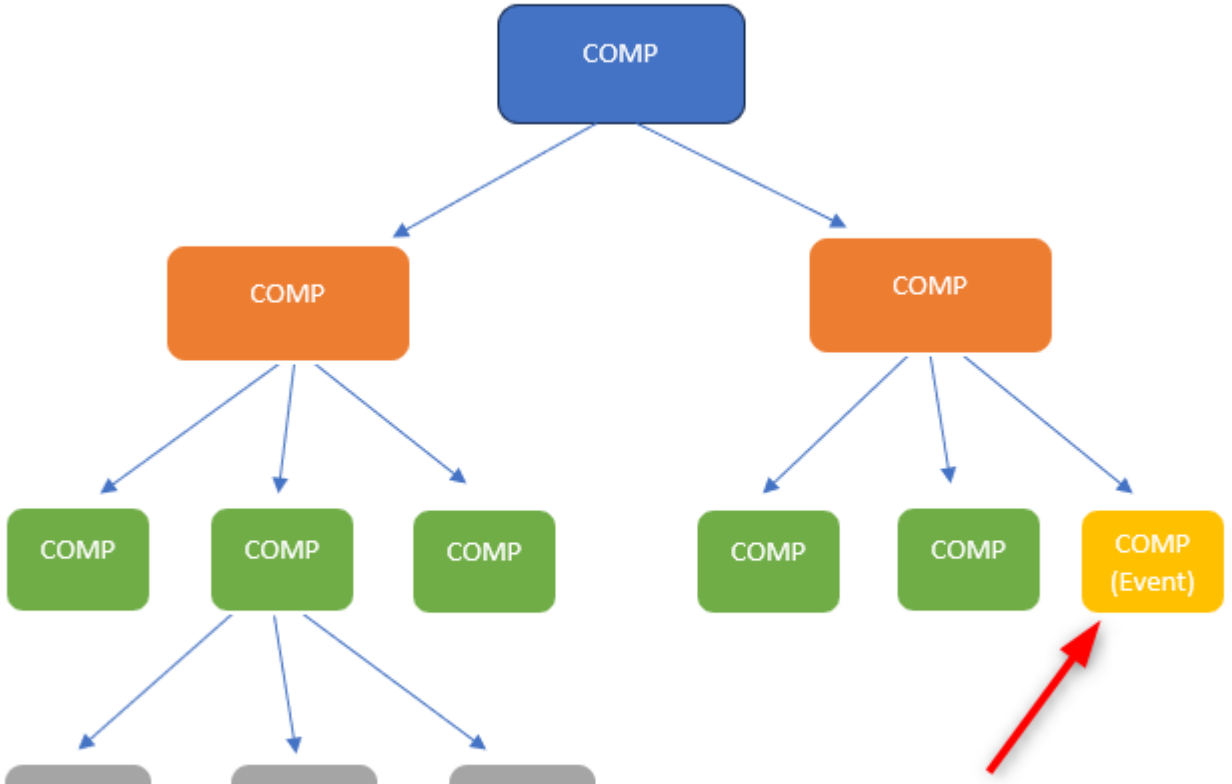
Comment Angular met à jour l'affichage des composants imbriqués par des Inputs,
dans une stratégie composants parents-enfants



Le Detection Change Mode par défaut

```
@Component({
  selector: 'app-liste-des-films',
  templateUrl: './liste-des-films.component.html',
  styleUrls: ['./liste-des-films.component.scss'],
  changeDetection: ChangeDetectionStrategy.Default,
})
```

2-Angular parcourt tout l'arbre des composants pour afficher un changement de valeur sur les composants



1-Une interaction survient sous la forme d'un Event sur un composant enfant

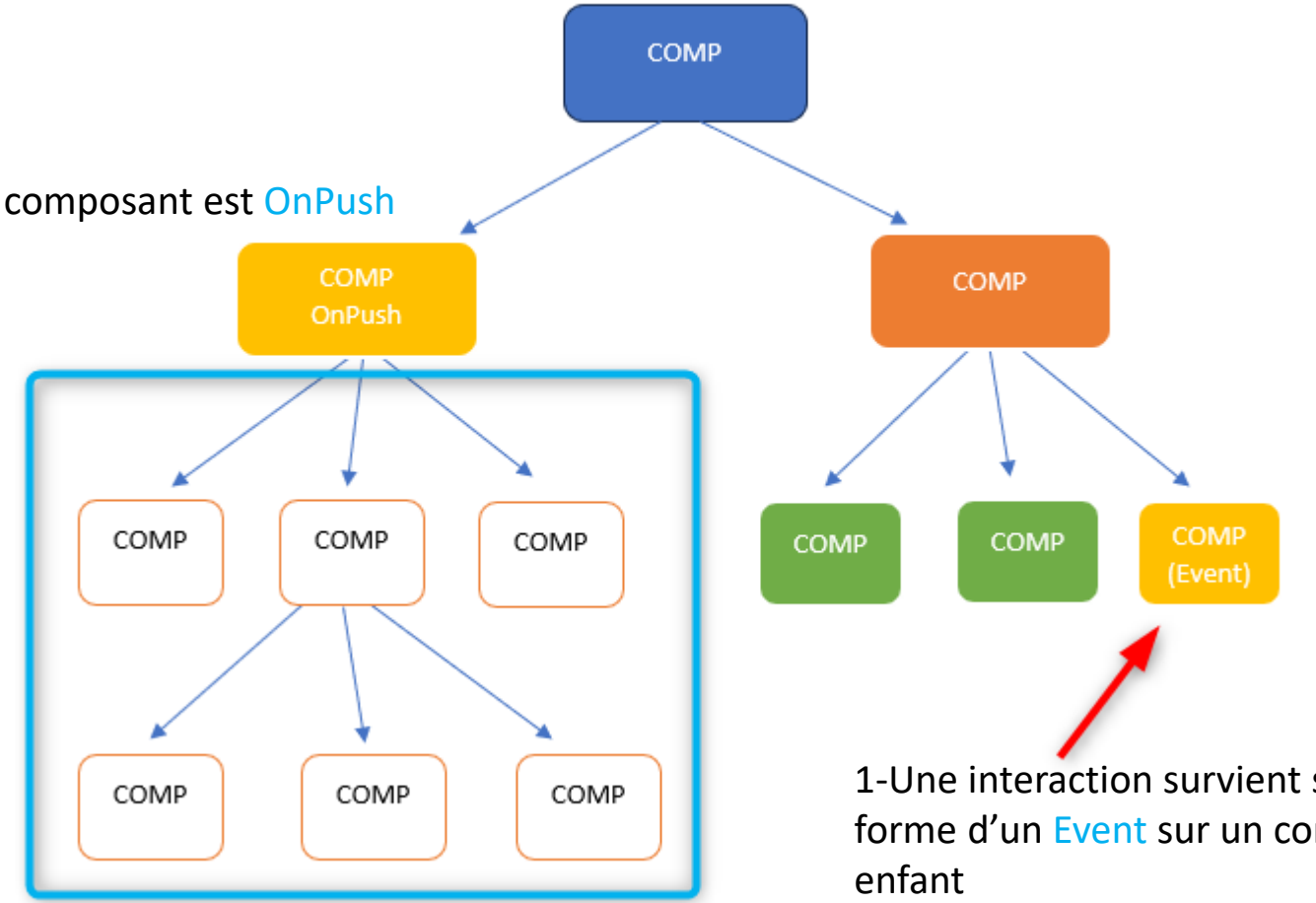
Le Detection Change Mode onPush

Angular limite le nombre de vérifications sur les enfants

Si la référence de l'objet (valeur…) passée en Input ne change pas, le composant enfant ne doit pas être modifié.

Angular arrête son parcours…

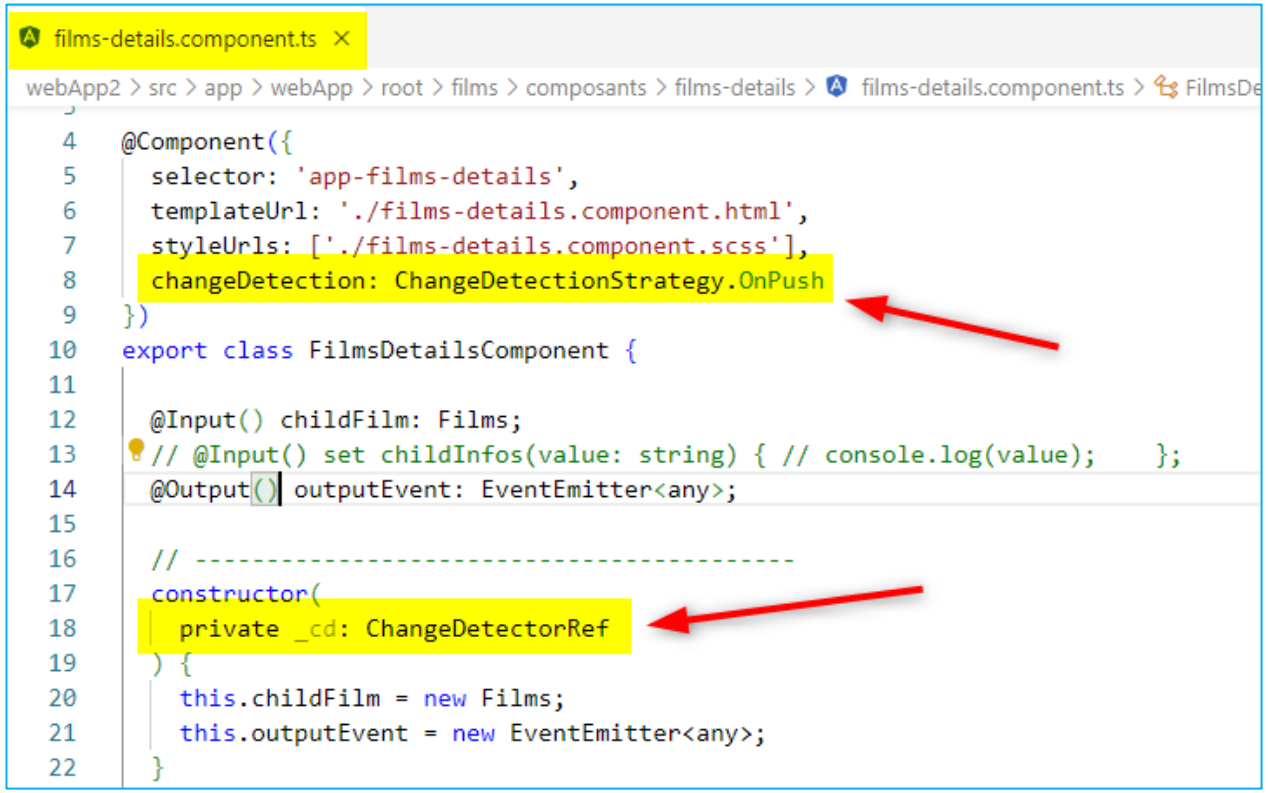
Le composant est OnPush



1-Une interaction survient sous la forme d'un Event sur un composant enfant

La class ChangeDetetctorRef

On peut forcer la mise à jour de l'affichage de la Vue d'un composant de manière explicite, sans attendre (ou dépendre) du déclenchement des valeurs d'entrées du composant.



La class ChangeDetetctorRef

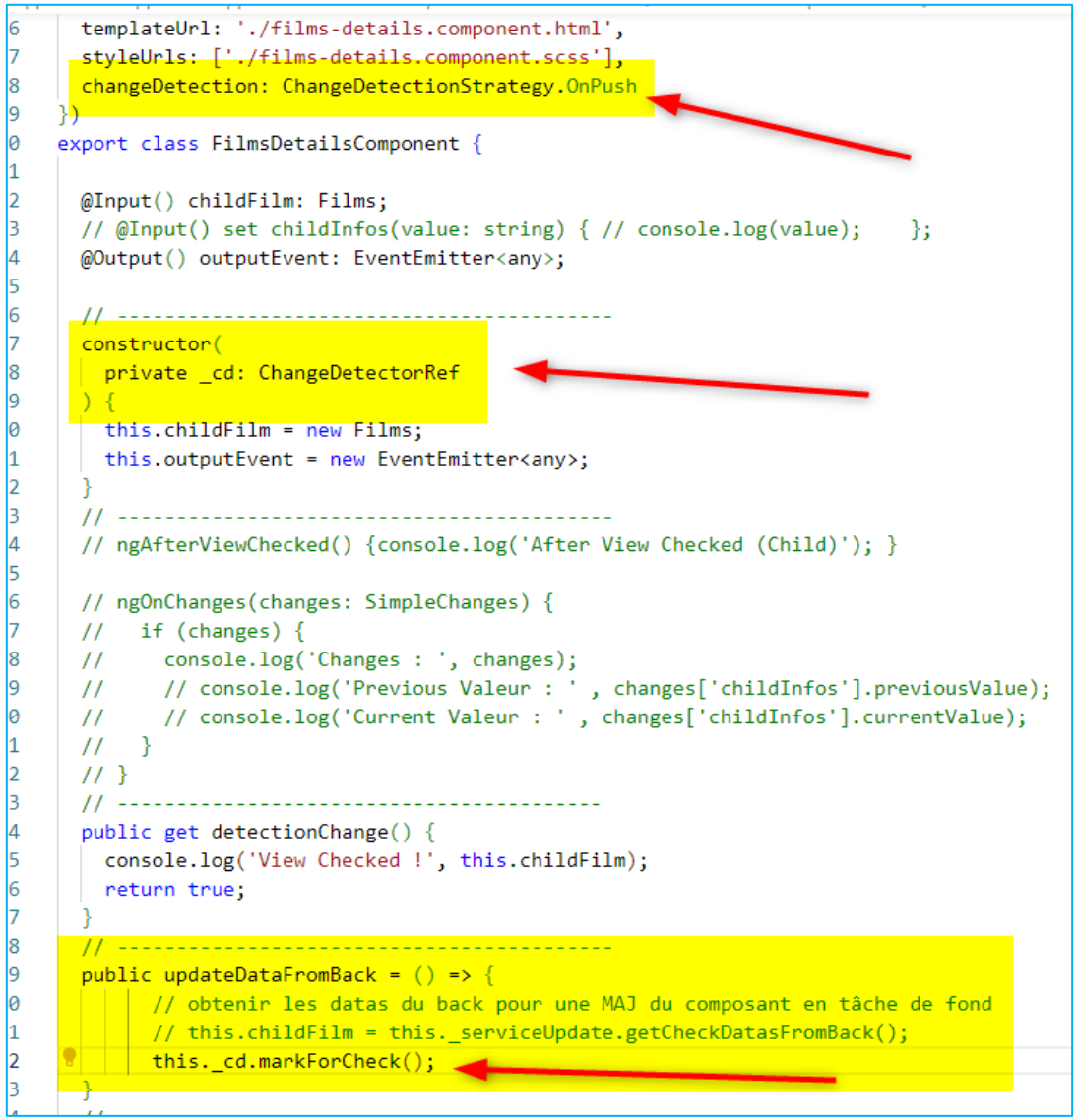
« markForCheck » indique que le composant doit être vérifié lors du prochain parcours de l'arbre des composants, même si la stratégie de mise à jour de « change detection » est OnPush

```
// ----------------------------------------
public updateDataFromBack = () => {
    // obtenir les datas du back pour ur
    // this.childFilm = this._serviceUpd
    this._cd.markForCheck();
}
```

```
// ----------------------------------------
constructor(
  private _cd: ChangeDetectorRef
) {
  this.childFilm = new Films;
  this.outputEvent = new EventEmitter<any>;
}
// ----------------------------------------
// ngAfterViewChecked() {console.log('After View Checked (Child)'); }

// ngOnChanges(changes: SimpleChanges) {
//   if (changes) {
//     console.log('Changes : ', changes);
//     // console.log('Previous Valeur : ' , changes['childInfos'].previousValu
//     // console.log('Current Valeur : ' , changes['childInfos'].currentValue
//   }
// }
// ----------------------------------------
public get detectionChange() {
  console.log('View Checked !', this.childFilm);
  return true;
}
// ----------------------------------------
public updateDataFromBack = () => {
    // obtenir les datas du back pour une MAJ du composant en tâche de fond
    // this.childFilm = this._serviceUpdate.getCheckDatasFromBack();
    this._cd.
}
// ----------
public choixFil
```

checkNoChanges
detach
detectChanges
markForCheck
reattach

La class ChangeDetetctorRef



```
  templateUrl: './films-details.component.html',
  styleUrls: ['./films-details.component.scss'],
  changeDetection: ChangeDetectionStrategy.OnPush
})
export class FilmsDetailsComponent {

  @Input() childFilm: Films;
  // @Input() set childInfos(value: string) { // console.log(value);    };
  @Output() outputEvent: EventEmitter<any>;

  // ----------------------------------------
  constructor(
    private _cd: ChangeDetectorRef
  ) {
    this.childFilm = new Films;
    this.outputEvent = new EventEmitter<any>;
  }
  // ----------------------------------------
  // ngAfterViewChecked() {console.log('After View Checked (Child)'); }

  // ngOnChanges(changes: SimpleChanges) {
  //   if (changes) {
  //     console.log('Changes : ', changes);
  //     // console.log('Previous Valeur : ' , changes['childInfos'].previousValue);
  //     // console.log('Current Valeur : ' , changes['childInfos'].currentValue);
  //   }
  // }
  // ----------------------------------------
  public get detectionChange() {
    console.log('View Checked !', this.childFilm);
    return true;
  }
  // ----------------------------------------
  public updateDataFromBack = () => {
      // obtenir les datas du back pour une MAJ du composant en tâche de fond
      // this.childFilm = this._serviceUpdate.getCheckDatasFromBack();
      this._cd.markForCheck();
  }
```

Le Signal

Meilleures performances en termes d'éxécution !
Le signal réduit la complexité du parcours de l'arbre des composants parents-enfants

La vérification du changement d'affichage dans le composant, devient de plus en plus fine et étroitement liée au composant concerné. La granularité du « Change Detection Mode » est nettement améliorée.

Zone.js utilisé depuis le début d'Angular pour détecter les changements dans les composants va être progressivement arrêté ! (pour rappel, le principe d'une Zone permet d'éxécuter du code dans le contexte Angular et hors contexte Angular) 😯

Le signaux peuvent interagir avec RXJS et apporter de la complémentarité !
On peut passer facilement d'un Observable à un signal et vice –versa

toSignal() toObservable()

Le Signal

Le signal représente l'état d'un composant, d’une propriété , d’une valeur spécifique.

Lorsque le signal est mis à jour, les parties de l'application qui dépendent de ce signal peuvent être informées et modifier leurs propres données. (mécanisme d’optimisation que l’on reconnaît dans le Store)

Lorsque une valeur change, le signal émet une information et l’application peut se mettre à jour de manière sélective.