miam | Web

Janvier–avril 2023

JUNIA
Grande école d'ingénieurs
HEI·ISEN·ISA

Rappels

Typescript :

Projets mieux structurés, plus grands, plus scalables.
Typé, meilleur linter, introspection de code etc.

Front : angular, react, vuejs

Back : node

Framework

Librairie, boîte à outils.
Outils déjà (bien) faits, on ne réinvente pas la roue.
Influence notre architecture.

NestJs :
Convention over configuration
Pattern MVC

Vous avez un projet de base, voici l’architecture

```
alexis@alexis-laptop-ubuntu:~/junia$ cd nestjs-project/
alexis@alexis-laptop-ubuntu:~/junia/nestjs-project$ ll
total 380
drwxrwxr-x   6 alexis alexis   4096 mars   5 22:31 ./
drwxrwxr-x  11 alexis alexis   4096 mars   5 22:31 ../
-rw-rw-r--   1 alexis alexis    663 mars   5 22:31 .eslintrc.js
drwxrwxr-x   7 alexis alexis   4096 mars   5 22:31 .git/
-rw-rw-r--   1 alexis alexis    391 mars   5 22:31 .gitignore
-rw-rw-r--   1 alexis alexis    171 mars   5 22:31 nest-cli.json
drwxrwxr-x 450 alexis alexis  20480 mars   5 22:31 node_modules/
-rw-rw-r--   1 alexis alexis   1945 mars   5 22:31 package.json
-rw-rw-r--   1 alexis alexis 310964 mars   5 22:31 package-lock.json
-rw-rw-r--   1 alexis alexis     51 mars   5 22:31 .prettierrc
-rw-rw-r--   1 alexis alexis   3340 mars   5 22:31 README.md
drwxrwxr-x   2 alexis alexis   4096 mars   5 22:31 src/
drwxrwxr-x   2 alexis alexis   4096 mars   5 22:31 test/
-rw-rw-r--   1 alexis alexis     97 mars   5 22:31 tsconfig.build.json
-rw-rw-r--   1 alexis alexis    546 mars   5 22:31 tsconfig.json
alexis@alexis-laptop-ubuntu:~/junia/nestjs-project$
```

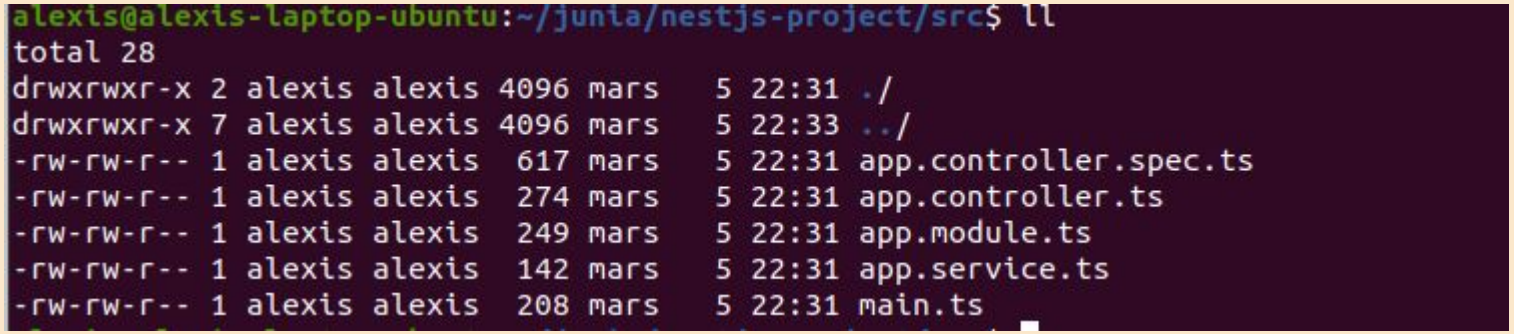
Rest :

Un standard qui donne des règles pour créer votre api.

C'est-à-dire que votre serveur va offrir des routes qui vont respecter certaines contraintes qui vont garantir une facilité d’utilisation, une scalabilité, une extensibilité.

NestJs les modules

- Au moins un module dans l'appli: root module = point de départ.
- Sert à organiser le code en fonctionnalités
- bonne pratique : un dossier par module
- un module est un singleton, il peut donc être importé par plusieurs autres module.



```
alexis@alexis-laptop-ubuntu:~/junia/nestjs-project/src$ cat app.module.ts
import { Module } from '@nestjs/common';
import { AppController } from './app.controller';
import { AppService } from './app.service';

@Module({
  imports: [],
  controllers: [AppController],
  providers: [AppService],
})
export class AppModule {}
```

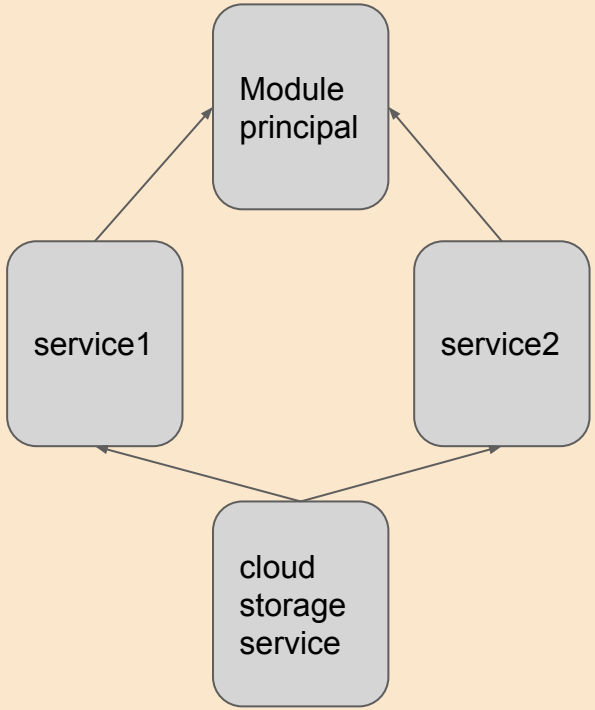
Un module est défini à l'aide du decorator @Module.

Donne des metadata utilisés par nestjs pour organiser la structure de l'appli

Les propriétés:

- providers: Array de providers utilisés dans le module via un injection de dependence (on verra par la suite les providers)
- controllers Array des controllers instanciés dans le module
- export : Array des providers à exporter aux autres modules
- imports: Array des modules requis pour celui ci

Exemple d’architecture modulaire :
J’ai deux services qui ont besoin d’un module qui permet d’envoyer des fichier sur cloudStorage



```
alexis@alexis-laptop-ubuntu:~/junia/nestjs-project$ nest g module leboncoin
CREATE src/leboncoin/leboncoin.module.ts (86 bytes)
UPDATE src/app.module.ts (328 bytes)
alexis@alexis-laptop-ubuntu:~/junia/nestjs-project$ ll
total 376
drwxrwxr-x   7 alexis alexis   4096 mars  14 20:26 ./
drwxrwxr-x  12 alexis alexis   4096 mars  10 19:59 ../
drwxrwxr-x   2 alexis alexis   4096 mars   5 22:33 dist/
-rw-rw-r--   1 alexis alexis    663 mars   5 22:31 .eslintrc.js
drwxrwxr-x   7 alexis alexis   4096 mars   5 22:31 .git/
-rw-rw-r--   1 alexis alexis    171 mars   5 22:31 nest-cli.json
drwxrwxr-x 450 alexis alexis  20480 mars   5 22:33 node_modules/
-rw-rw-r--   1 alexis alexis   1945 mars   5 22:31 package.json
-rw-rw-r--   1 alexis alexis 310964 mars   5 22:33 package-lock.json
-rw-rw-r--   1 alexis alexis     51 mars   5 22:31 .prettierrc
-rw-rw-r--   1 alexis alexis   3340 mars   5 22:31 README.md
drwxrwxr-x   3 alexis alexis   4096 mars  14 20:32 src/
drwxrwxr-x   2 alexis alexis   4096 mars   5 22:31 test/
-rw-rw-r--   1 alexis alexis     97 mars   5 22:31 tsconfig.build.json
-rw-rw-r--   1 alexis alexis    546 mars   5 22:31 tsconfig.json
alexis@alexis-laptop-ubuntu:~/junia/nestjs-project$ ll src/
total 32
drwxrwxr-x 3 alexis alexis 4096 mars  14 20:32 ./
drwxrwxr-x 7 alexis alexis 4096 mars  14 20:26 ../
-rw-rw-r-- 1 alexis alexis  617 mars   5 22:31 app.controller.spec.ts
-rw-rw-r-- 1 alexis alexis  274 mars   5 22:31 app.controller.ts
-rw-rw-r-- 1 alexis alexis  328 mars  14 20:32 app.module.ts
-rw-rw-r-- 1 alexis alexis  142 mars   5 22:31 app.service.ts
drwxrwxr-x 2 alexis alexis 4096 mars  14 20:32 leboncoin/
-rw-rw-r-- 1 alexis alexis  208 mars   5 22:31 main.ts
alexis@alexis-laptop-ubuntu:~/junia/nestjs-project$ ll src/leboncoin/
total 12
drwxrwxr-x 2 alexis alexis 4096 mars  14 20:32 ./
drwxrwxr-x 3 alexis alexis 4096 mars  14 20:32 ../
-rw-rw-r-- 1 alexis alexis   86 mars  14 20:32 leboncoin.module.ts
```

```
alexis@alexis-laptop-ubuntu:~/junia/nestjs-project$ cat src/app.module.ts
import { Module } from '@nestjs/common';
import { AppController } from './app.controller';
import { AppService } from './app.service';
import { LeboncoinModule } from './leboncoin/leboncoin.module';

@Module({
  imports: [LeboncoinModule],
  controllers: [AppController],
  providers: [AppService],
})
export class AppModule {}
```

```
alexis@alexis-laptop-ubuntu:~/junia/nestjs-project$ cat src/leboncoin/leboncoin.module.ts
import { Module } from '@nestjs/common';

@Module({})
export class LeboncoinModule {}
```

Les controllers

- gère les requêtes entrantes et les réponses
- lié à une route spécifique (ex: /annonces)
- Possède des handlers pour gérer les endpoints et les methodes (GET, POST, DELETE, etc.)
- Se sert les dépendances pour utiliser les providers.

Un Controller est défini à l'aide du decorator @Controller.
Une string qui définit le chemin vers la route contrôlée.

```
@Controller('/annonces')
export class AnnoncesController {


}
```

Les handlers sont simplement des méthodes liées au controller, décorées avec les decorators commes @Get, @Post, @Delete etc. pour définir les méthodes correspondantes.

```
@Controller('/annonces')
export class AnnoncesController {
 @Get()
 getAllAnnonces() {
   console.log('get all');
 }

 @Post()
 createAnnonce() {
   console.log('create');
 }

 @Delete()
 deleteAnnonce() {
   console.log('delete');
 }
}
```


HTTP request incoming
Request routed to Controller, handler is called with arguments
NestJS will parse the relevant request data and it will be available in the handler.
Handler handles the request
Perform operations such as communication with a service. For example, retrieving an item from the database.
Handler returns response value
Response can be of any type and even an exception. Nest will wrap the returned value as an HTTP response and return it to the client.

```
alexis@alexis-laptop-ubuntu:~/junia/nestjs-project$ nest g controller annonces --no-spec
CREATE src/annonces/annonces.controller.ts (105 bytes)
UPDATE src/app.module.ts (417 bytes)
alexis@alexis-laptop-ubuntu:~/junia/nestjs-project$ ll src/
total 36
drwxrwxr-x 4 alexis alexis 4096 mars  14 22:08 ./
drwxrwxr-x 7 alexis alexis 4096 mars  14 20:26 ../
drwxrwxr-x 2 alexis alexis 4096 mars  14 22:08 annonces/
-rw-rw-r-- 1 alexis alexis  617 mars   5 22:31 app.controller.spec.ts
-rw-rw-r-- 1 alexis alexis  274 mars   5 22:31 app.controller.ts
-rw-rw-r-- 1 alexis alexis  417 mars  14 22:08 app.module.ts
-rw-rw-r-- 1 alexis alexis  142 mars   5 22:31 app.service.ts
drwxrwxr-x 2 alexis alexis 4096 mars  14 20:32 leboncoin/
-rw-rw-r-- 1 alexis alexis  208 mars   5 22:31 main.ts
alexis@alexis-laptop-ubuntu:~/junia/nestjs-project$ ll src/annonces/
total 12
drwxrwxr-x 2 alexis alexis 4096 mars  14 22:08 ./
drwxrwxr-x 4 alexis alexis 4096 mars  14 22:08 ../
-rw-rw-r-- 1 alexis alexis  105 mars  14 22:08 annonces.controller.ts
```

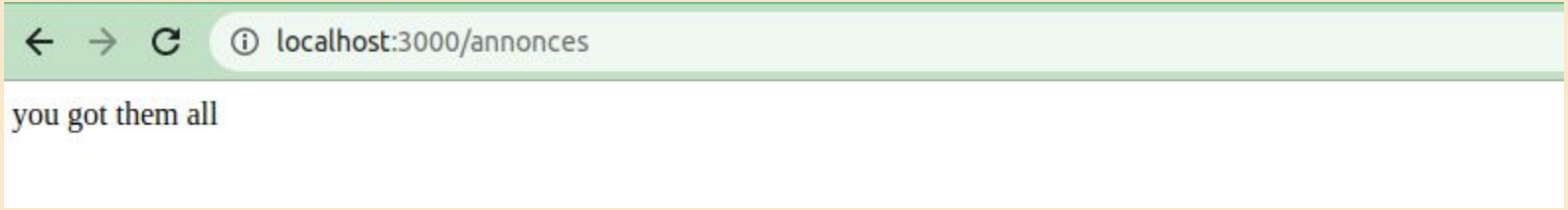
Si générés dans l’ordre, le controller est automatiquement ajouté au module

```
alexis@alexis-laptop-ubuntu:~/junia/nestjs-project$ cat src/annonces/annonces.module.ts
import { Module } from '@nestjs/common';
import { AnnoncesController } from './annonces.controller';

@Module({
  controllers: [AnnoncesController]
})
export class AnnoncesModule {}
```

Exemple : ma première route complète

```
import { Controller, Get } from
'@nestjs/common';

@Controller('annonces')
export class AnnoncesController {
 @Get()
 getAllAnnonces() {
   console.log('get all annonces');
   return 'you got them all';
 }
}
```

```
alexis@alexis-laptop-ubuntu:~/junia/nestjs-project$ npm run start

> nestjs-project@0.0.1 start
> nest start

[Nest] 98570  - 14/03/2023 22:19:33     LOG [NestFactory] Starting Nest application...
[Nest] 98570  - 14/03/2023 22:19:33     LOG [InstanceLoader] LeboncoinModule dependencies initialized +25ms
[Nest] 98570  - 14/03/2023 22:19:33     LOG [InstanceLoader] AnnoncesModule dependencies initialized +0ms
[Nest] 98570  - 14/03/2023 22:19:33     LOG [InstanceLoader] AppModule dependencies initialized +0ms
[Nest] 98570  - 14/03/2023 22:19:33     LOG [RoutesResolver] AppController {/}: +4ms
[Nest] 98570  - 14/03/2023 22:19:33     LOG [RouterExplorer] Mapped {/, GET} route +2ms
[Nest] 98570  - 14/03/2023 22:19:33     LOG [RoutesResolver] AnnoncesController {/annonces}: +0ms
[Nest] 98570  - 14/03/2023 22:19:33     LOG [RouterExplorer] Mapped {/annonces, GET} route +0ms
[Nest] 98570  - 14/03/2023 22:19:33     LOG [RoutesResolver] AnnoncesController {/annonces}: +0ms
[Nest] 98570  - 14/03/2023 22:19:33     LOG [RouterExplorer] Mapped {/annonces, GET} route +0ms
[Nest] 98570  - 14/03/2023 22:19:33     LOG [NestApplication] Nest application successfully started +2ms
get all annonces
```



Providers et services

Providers

- peut être injecté dans les constructors avec le decorator @Injectable
- Doit être povidé dans les modules pour être utilisable
- Peut être exporté pour être disponible dans les autres modules

Sevices

- Défini comme un provider. Tous les providers ne sont pas des services
- Ce qu'on a déjà vu, concept général pour offrire un service, utilisable a différents endroits de l'appli
- Automatiquement des singletons quand wrapped avec le decorator @Injectable() et provided dans un module.
- La source principale de la logique buisness. Elle est ici et non dans les controllers.

Dependency injection

Any component within the NestJS ecosystem can inject a provider that is decorated with the **@Injectable**.

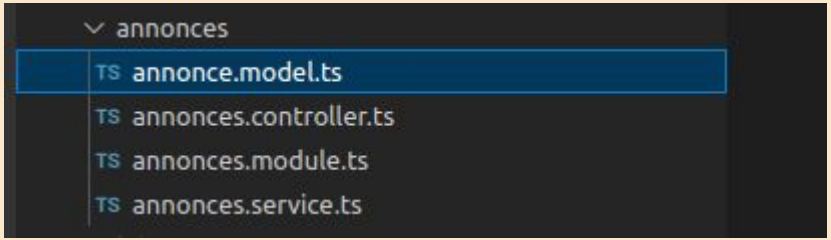
We define the dependencies in the constructor of the class. NestJS will take care of the injection for us, and it will then be available as a class property.

```
alexis@alexis-laptop-ubuntu:~/junia/nestjs-project$ nest g service annonces --no-spec
CREATE src/annonces/annonces.service.ts (92 bytes)
UPDATE src/annonces/annonces.module.ts (269 bytes)
alexis@alexis-laptop-ubuntu:~/junia/nestjs-project$ ll src/annonces/
total 20
drwxrwxr-x 2 alexis alexis 4096 mars  14 22:33 ./
drwxrwxr-x 4 alexis alexis 4096 mars  14 22:08 ../
-rw-rw-r-- 1 alexis alexis  213 mars  14 22:19 annonces.controller.ts
-rw-rw-r-- 1 alexis alexis  269 mars  14 22:33 annonces.module.ts
-rw-rw-r-- 1 alexis alexis   92 mars  14 22:33 annonces.service.ts
alexis@alexis-laptop-ubuntu:~/junia/nestjs-project$ cat src/annonces/annonces.module.ts
import { Module } from '@nestjs/common';
import { AnnoncesController } from './annonces.controller';
import { AnnoncesService } from './annonces.service';

@Module({
  controllers: [AnnoncesController],
  providers: [AnnoncesService],
})
export class AnnoncesModule {}
alexis@alexis-laptop-ubuntu:~/junia/nestjs-project$
```

```
import { Controller, Get } from '@nestjs/common';
import { AnnoncesService } from './annonces.service';

@Controller('annonces')
export class AnnoncesController {
 constructor(private annoncesService: AnnoncesService) {}

 @Get()
 getAllAnnonces() {
   console.log('get all annonces');
   return 'you got them all';
 }
}
```

Création du model

Attention au naming



interface ou classe selon le besoin

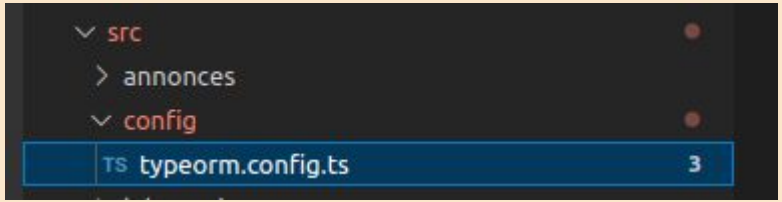
```
export interface Annonce {
 id: string;
 title: string;
 price: number;
}
```

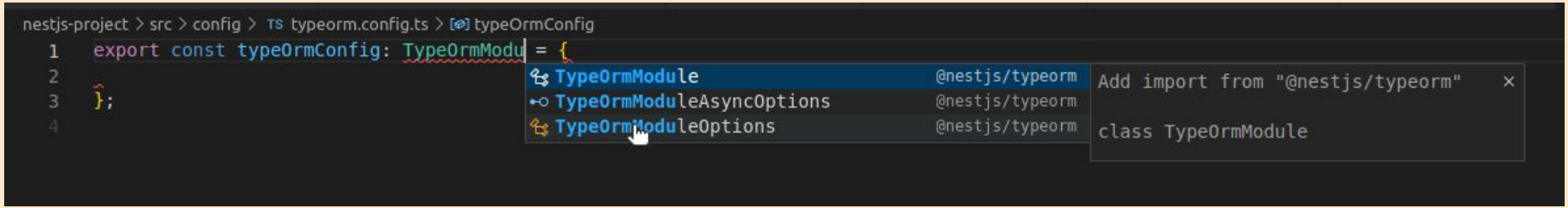
Liaison avec notre base et un orm

1. Installation de l'orm

```
alexis@alexis-laptop-ubuntu:~/junia/nestjs-project$ npm install --save @nestjs/typeorm typeorm mysql
```

2. config de l'orm



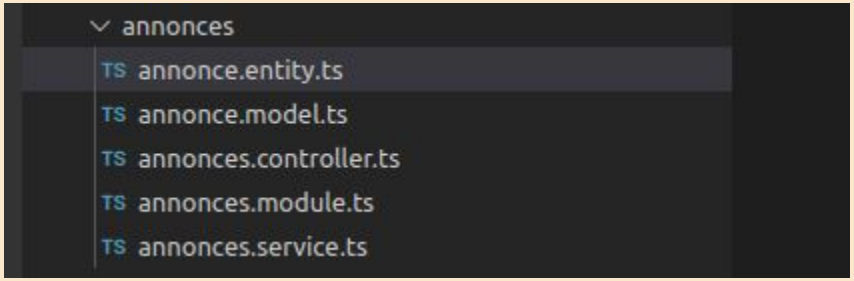


```ts
import { TypeOrmModuleOptions } from '@nestjs/typeorm';

export const typeOrmConfig: TypeOrmModuleOptions = {
 type: 'mysql',
 host: 'localhost',
 port: 3306,
 username: 'junia_user',
 password: 'junia_user',
 database: 'junia',
 entities: [Annonce],
};
```

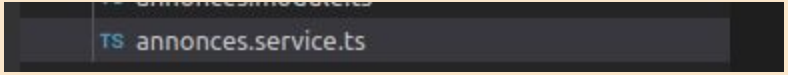
TS app.module.ts U

```ts
import { Module } from '@nestjs/common';
import { AppController } from './app.controller';
import { AppService } from './app.service';
import { AnnoncesController } from './annonces/annonces.controller';
import { AnnoncesModule } from './annonces/annonces.module';
import { TypeOrmModule } from '@nestjs/typeorm';
import { AnnoncesService } from './annonces/annonces.service';
import { typeOrmConfig } from './config/typeorm.config';

@Module({
 imports: [TypeOrmModule.forRoot(typeOrmConfig), AnnoncesModule],
 controllers: [AppController, AnnoncesController],
 providers: [AppService, AnnoncesService],
})
export class AppModule {}
```

entities



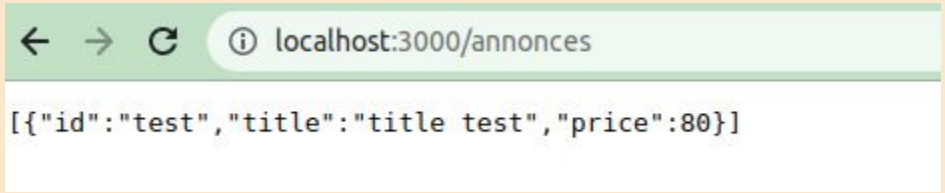

```
import { BaseEntity, Column, Entity } from 'typeorm';

@Entity('annonces')
export class Annonce extends BaseEntity {
 @PrimaryColumn()
 id: string;
 @Column()
 title: string;
 @Column()
 price: number;
}
```




```ts
import { Injectable } from '@nestjs/common';
import { InjectRepository } from '@nestjs/typeorm';
import { Repository } from 'typeorm';
import { Annonce } from './annonce.entity';

@Injectable()
export class AnnoncesService {
 constructor(
   @InjectRepository(Annonce)
   private annoncesRepository: Repository<Annonce>,
 ) {}

 async getAll() {
   const res = await this.annoncesRepository.find();
   console.log('res is :', res);
   return res;
 }
```

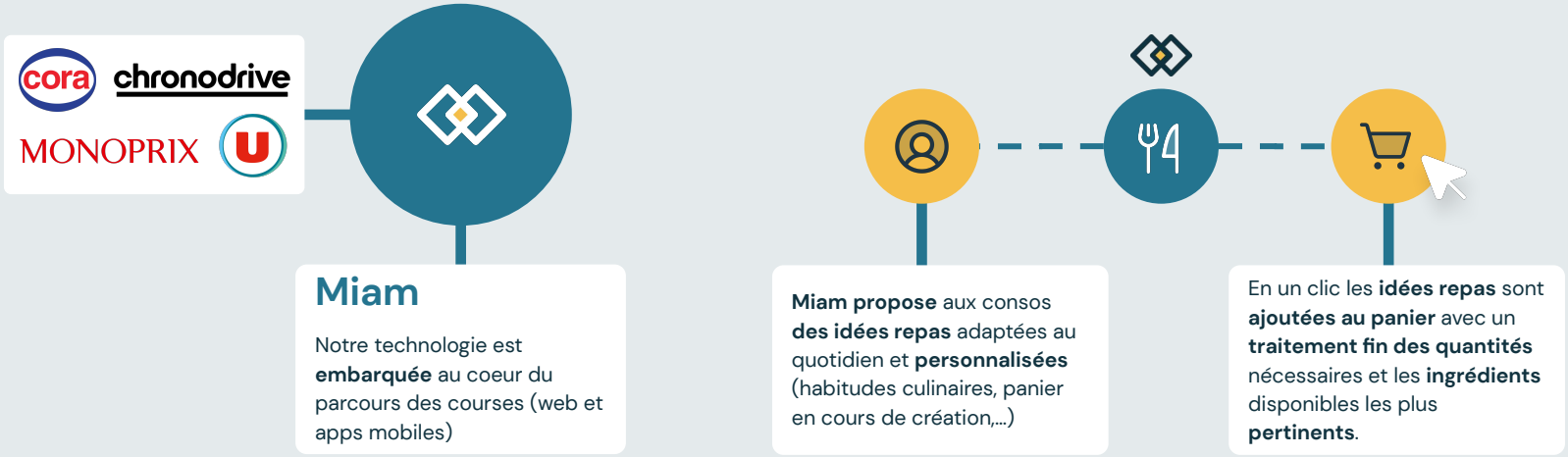
TS annonces.controller.ts

```ts
import { Controller, Get } from '@nestjs/common';
import { AnnoncesService } from './annonces.service';

@Controller('annonces')
export class AnnoncesController {
 constructor(private annoncesService: AnnoncesService) {}

 @Get()
 getAllAnnonces() {
   console.log('get all annonces');
   return this.annoncesService.getAll();
 }
}
```



A propos de Miam

Miam réinvente et simplifie les courses alimentaires.



Miam

Notre technologie est **embarquée** au coeur du parcours des courses (web et apps mobiles)

Miam propose aux consos **des idées repas** adaptées au quotidien et **personnalisées** (habitudes culinaires, panier en cours de création,...)

En un clic les **idées repas** sont **ajoutées au panier** avec un **traitement fin des quantités** nécessaires et les **ingrédients** disponibles les plus **pertinents**.

Qui sommes-nous ? Une startup tech lilloise, qui développe depuis 2019 des solutions d'intelligence artificielle à destination de la grande distribution.

Plus de détails sur https://miam.tech