COMPTE-RENDU TP2 NESTJS

RÉALISATRICE:

FATMA LARIBI, GL3 GROUPE 2

PLAN

I. Exercice 1: Queries sans et avec QueryBuilder
II. Exercice 2 : Pagination et CRUD générique
III. Exercice 3 : Stats
IV. Exercie 4 : Relations
V. Exercice 5 : Standalone application pour seed de base de données avec Faker

LES RÉPOS

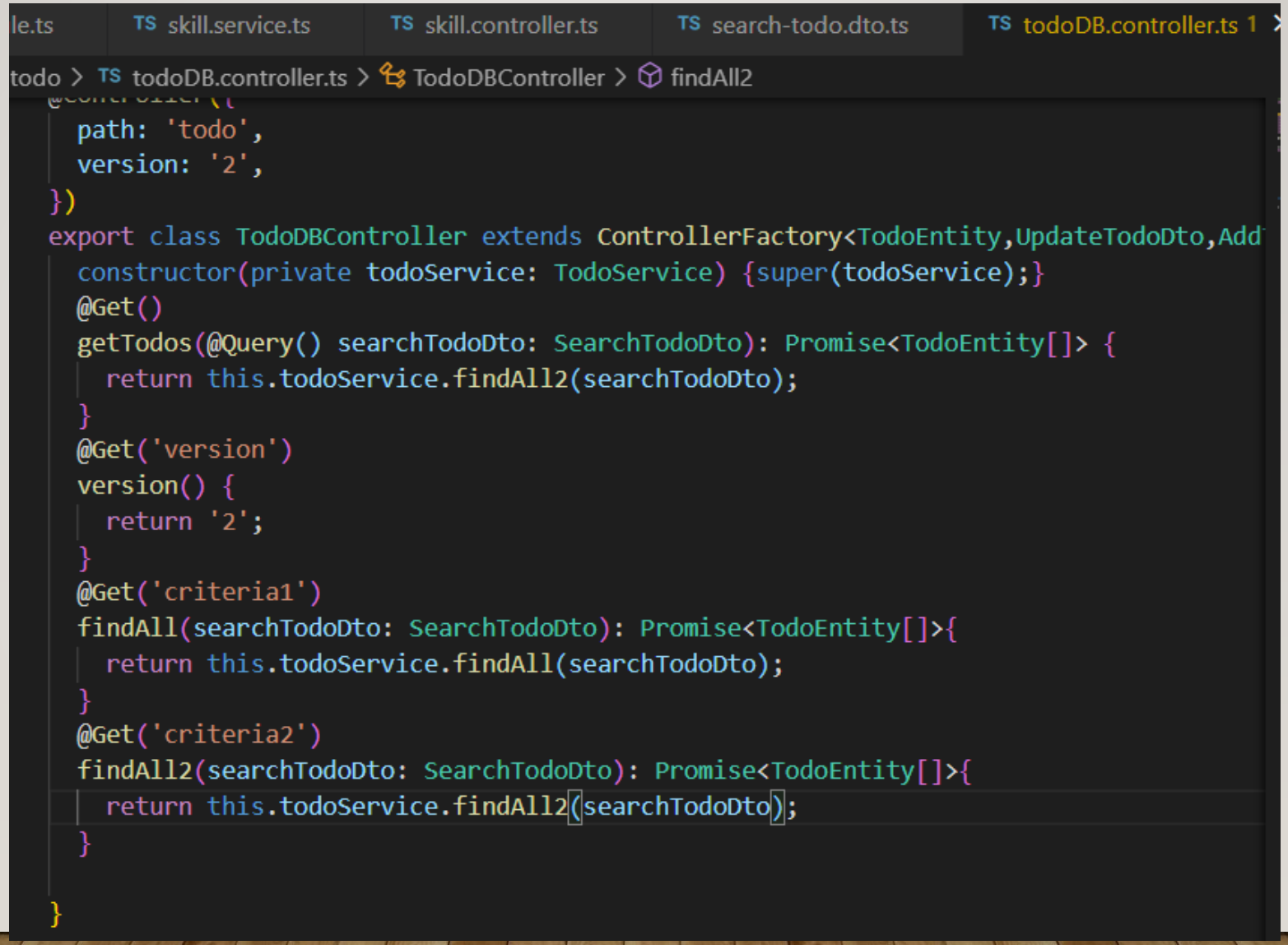
Les 4 premiers exercices: https://github.com/fatma-laribi/NestJsTp2.git

Standalone app: https://github.com/fatma-laribi/standaloneDBSeed.git

EXERCICE 1:

- Modifier l'api get de sorte que maintenant on puisse avoir les Todo dont le name **ou** description contiennent la chaine passée en paramètre **et** ayant le statut passé en paramètre.
- Les deux critères de recherche restent optionnels.

Ajout des routes (findAll et findAll2) dans le controleur todoDBController



Modification du service

findAll: sans utilisation de QueryBuilder
findAll2 : avec QueryBuilder

```
findAll(searchTodoDto: SearchTodoDto): Promise<TodoEntity[]> {
  const criterias = [];
  if (searchTodoDto.status) {
    criterias.push({ status: searchTodoDto.status });
  }
  if (searchTodoDto.criteria) {
    criterias.push({ name: Like(`%${searchTodoDto.criteria}%`) });
    criterias.push({ description: Like(`%${searchTodoDto.criteria}%`) });
  }
  if (criterias.length) {
    return this.todoRepository.find({ withDeleted: true, where: criterias });
  }
  return this.todoRepository.find({ withDeleted: true});
}
```

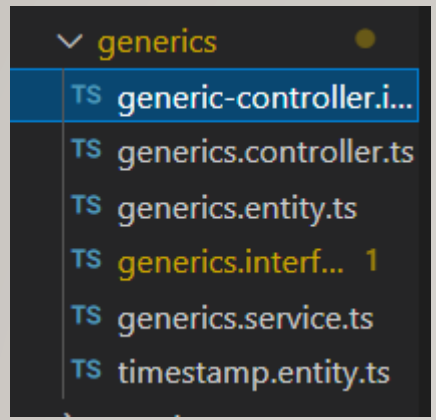
```
findAll2(searchTodoDto: SearchTodoDto) : Promise<TodoEntity[]>{
  const queryBuilder = this.todoRepository.createQueryBuilder("todo");

  if(searchTodoDto.status){
    queryBuilder.where("todo.status = :status",{status:searchTodoDto.status}) ;
    if(searchTodoDto.criteria){
      queryBuilder.andWhere(new Brackets(
        qb => qb.where("todo.name= :name",{name:searchTodoDto.criteria})
        .orWhere("todo.description= :description",{description:searchTodoDto.criteria})
      ));
      }
  }
  else if(searchTodoDto.criteria){
    queryBuilder.andWhere(new Brackets(
      qb => qb.where("todo.name= :name",{name:searchTodoDto.criteria})
      .orWhere("todo.description= :description",{description:searchTodoDto.criteria})
    ));
  }
  return queryBuilder.getMany();

}
```

EXERCICE 2:

- Ajouter le traitement nécessaire pour paginer votre fonction getAll.



Les détails des fichiers génériques ajoutés:
https://github.com/fatma-laribi/NestJsTp2/tree/master/src/generics

Pour utiliser les fonctionnalités dans les autres modules:
Dans le Service (par exemple celui de Todo):
export class TodoService extends GenericService<TodoEntity,UpdateTodoDto,AddTodoDto>
Dans le controleur:
export class TodoDBController extends
ControllerFactory<TodoEntity,UpdateTodoDto,AddTodoDto>(UpdateTodoDto,AddTodoDto)

La raison pour laquelle on utilise ControllerFactory c'est pour que le class-validator fonctionne et pour qu'on puisse passer les dto (pour le cas générique, la validation ne fonctionne plus sauf avec cette méthode)

- ✓ Il suffit d'étendre le controleur générique et le service générique en passant les dto correspondants pour qu'on ait toutes les fonctionnalités du crud
- ✓ La méthode getAll:

```
async getAll(take = 10, skip = 0){


  const [data, total] = await this.repository.findAndCount({ take, skip });
  return { data, total };
}
async addEntity(entity: createDto):Promise<T> {
  return await this.repository.save(entity as any);
}
```

EXERCICE 3 : STATS

```
@Injectable()
export class StatsService {
    constructor(
    @InjectRepository(TodoEntity)
    private todoRepository){}
   async getTodoNumber(){
        const todos=[];
        for (const key in TodoStatusEnum) {
            const [result, total] = await this.todoRepository.findAndCount({status: TodoStatusEnum[key]});
          todos.push(`status: ${TodoStatusEnum[key]}`);
            todos.push(total);
          }
        return todos;
    }
    getTodosByDates(dateDebut:Date,dateFin:Date){
        const todos= this.todoRepository.find({createdAt: Between(dateDebut,dateFin)});
        return todos;
    }
}
```

```
@Controller('stats')
export class StatsController {
    constructor(private statsService: StatsService) {}
    @Get('status')
    getTodoNumber(){
        return this.statsService.getTodoNumber();
    }
    @Get('date')
    getTodosByDates(@Query("dateDebut") dateDebut:Date,@Query("dateFin") dateFin:Date){
        return this.statsService.getTodosByDates(dateDebut,dateFin);
    }
}
```

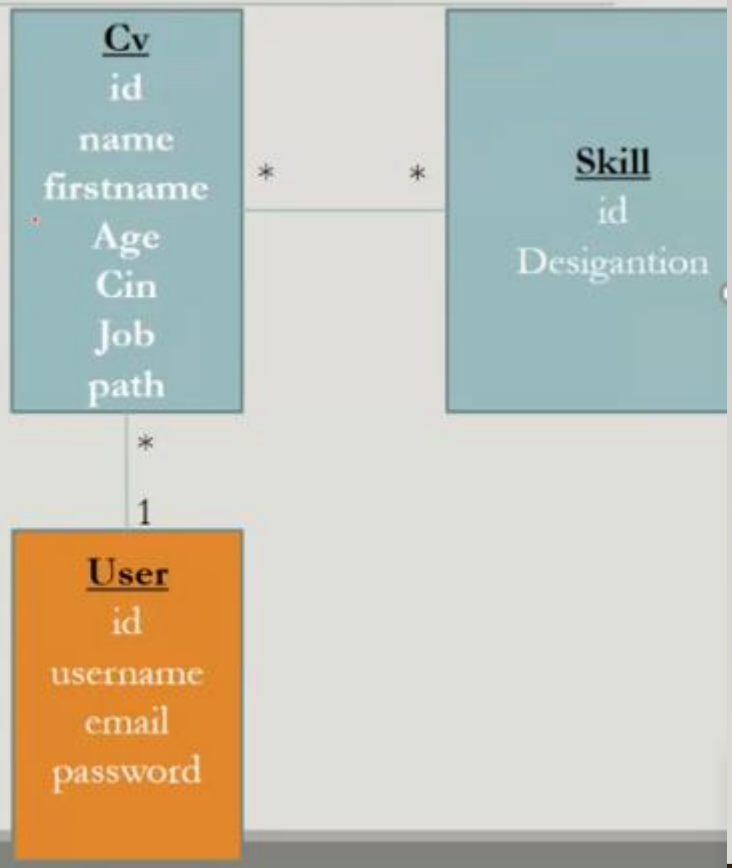
EXERCICE 4: RELATIONS

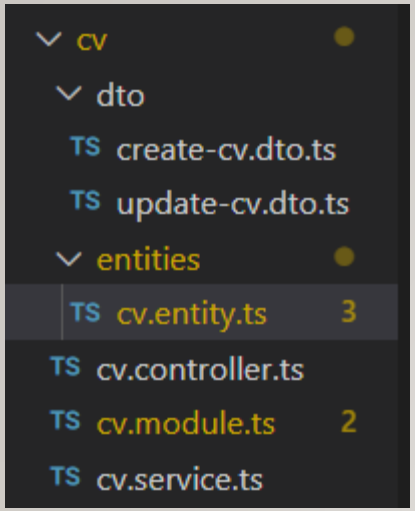
- Nous voulons reproduire le schéma relatif à un petit gestionnaire de cvs.
- Créer les modules, contrôleurs, et services et entités relatives à ce schéma ainsi que les relations qui y sont associées.

Astuce : vous pouvez utiliser la commande :

nest generate resource

- Cette commande génère non seulement tous les blocs de construction NestJS (module, service, classes de contrôleur), mais également une classe d'entité, des classes DTO ainsi que les fichiers de test (.spec).





L'entité Cv

```ts
/* eslint-disable prettier/prettier */
import { Type } from "class-transformer";
import { IsNotEmpty, IsNumber } from "class-validator";
import { identity } from "rxjs";
import { TimestampEntity } from "src/generics/timestamp.entity";
import { Skill } from "src/skill/entities/skill.entity";
import { User } from "src/user/entities/user.entity";
import { Column, Entity, ManyToMany, ManyToOne, PrimaryGeneratedColumn } from "typeorm";
@Entity("cv")
export class Cv extends TimestampEntity{

@Column()
name:string;

@Column()
firstName:string;

@Column()
@Type(()=>Number)
@IsNumber()
age:number;

@Column()
@Type(()=>Number)
@IsNumber()
cin:number;

@Column()
job:string;

@Column()
path:string;

@ManyToMany(
    ()=>Skill
)
skills:Skill[];

@ManyToOne(

    ()=>User
)
user:User;
}
```

Grâce à nest generate resource, on a généré les dto, entities, module, controller, et service
Pour le controller et le service, on n'a qu'à étendre du crud générique (de même pour Skill et User)

```typescript
/* eslint-disable prettier/prettier */
import { Controller} from '@nestjs/common';
import { CvService } from './cv.service';
import { ControllerFactory} from 'src/generics/generics.controller';
import { Cv } from './entities/cv.entity';
import { UpdateCvDto } from './dto/update-cv.dto';
import { CreateCvDto } from './dto/create-cv.dto';

@Controller('cv')
export class CvController  extends ControllerFactory<Cv,UpdateCvDto,CreateCvDto>(UpdateCvDto,CreateCvDto){

    constructor(private readonly cvService: CvService) {
        super(cvService);}

}
```

```ts
/* eslint-disable prettier/prettier */
import { Injectable } from '@nestjs/common';
import { InjectRepository } from '@nestjs/typeorm';
import { GenericService } from 'src/generics/generics.service';
import { Repository } from 'typeorm';
import { CreateCvDto } from './dto/create-cv.dto';
import { UpdateCvDto } from './dto/update-cv.dto';
import { Cv } from './entities/cv.entity';

@Injectable()
export class CvService  extends GenericService<Cv,UpdateCvDto,CreateCvDto>{
    constructor(
        @InjectRepository(Cv)
        private readonly cvRepository: Repository<Cv>) {
            super(cvRepository);
    }
  }
```

```ts
/* eslint-disable prettier/prettier */
import { Module } from '@nestjs/common';
import { CvService } from './cv.service';
import { CvController } from './cv.controller';
import { Cv } from './entities/cv.entity';
import { TypeOrmModule } from '@nestjs/typeorm';
import { TodoModule } from 'src/todo/todo.module';
import { EntityRepository } from 'typeorm';


@Module({
  imports:[TypeOrmModule.forFeature([Cv])],//
  controllers: [CvController],
  providers: [CvService]
})
export class CvModule {}
```

L’entité Skill

```ts
/* eslint-disable prettier/prettier */
import { Cv } from "src/cv/entities/cv.entity";
import { TimestampEntity } from "src/generics/timestamp.entity";
import { Column, Entity, ManyToMany, PrimaryGeneratedColumn } from "typeorm";
@Entity("skill")
export class Skill extends TimestampEntity{

    @Column()
    designation:string;
    @ManyToMany(
        ()=>Cv
    )
    cvs:Cv[];

}
```

RQ: La propriété id existe dans TimeStampEntity

L'entité User

```
/* eslint-disable prettier/prettier */
import { IsEmail } from "class-validator";
import { CvService } from "src/cv/cv.service";
import { Cv } from "src/cv/entities/cv.entity";
import { TimestampEntity } from "src/generics/timestamp.entity";
import { Column, Entity, OneToMany, PrimaryGeneratedColumn } from "typeorm";
@Entity("user")
export class User extends TimestampEntity{

    @Column()
    username:string;
    @Column()
    @IsEmail()
    email:string;

    @Column()
    password:string;
   @OneToMany(
        ()=>Cv,
        (cv:Cv)=>cv.id
    )
    cvs:Cv[];

}
```

EXERCICE 5: STANDALONE APP + FAKER

- Créer une standalone application permettant le seed de votre Base de données.

1- Execution de nmp i faker et npm install -D @types/faker

2- main.ts

src > TS main.ts > ...

```ts
/* eslint-disable prettier/prettier */
import { NestFactory } from '@nestjs/core';
import { Connection, createConnection } from 'typeorm';
import { AppModule } from './app.module';
import { AppService } from './app.service';
import { createUsers } from './createUsers.crud';
import { UserEntity } from './user.entity';

async function bootstrap() {
  const app = await NestFactory.createApplicationContext(AppModule);
  const appService = app.get(AppService);
  console.log(appService.getHello());
}
bootstrap();
const app = async () => {
  const connection: Connection = await createConnection({
    type: 'mysql',
    entities: [
      UserEntity,
    ],
    host: 'localhost',
    port: 3306,
    username: 'root',
    password: '',
    database: 'testseed',
    synchronize:true,
  })

  await connection.synchronize(false).catch(console.error)
  await createUsers(connection)
}

app()
```

3- création de UserEntity

user.entity.ts > ...

```ts
/* eslint-disable prettier/prettier */
import { Column, Entity, PrimaryGeneratedColumn } from "typeorm";

@Entity('user')
export class UserEntity{
    @PrimaryGeneratedColumn()
    id:number;

    @Column()
    firstName:string;

    @Column()
    lasttName:string;

    @Column()
    birthDate:Date;

    @Column()
    email:string;

    @Column()
    password:string;
}
```

4- création de createUsers

createUsers.crud.ts > ...

```ts
/* eslint-disable prettier/prettier */
import { Connection } from "typeorm";
import { UserEntity } from "./user.entity";
import { name, internet, date} from 'faker';
const createUsers = async (con: Connection) => {
    const users: Array<UserEntity> = [];
    for (const _ of Array.from({ length: 10 })) {
      const firstName = name.firstName();
      const lastName = name.lastName();
      const email = internet.email();
      const password = internet.password();
      const birthDate = date.past();
      const user: Partial<UserEntity> = new UserEntity(
      );
      user.birthDate=birthDate;
      user.email=email;
      user.firstName=firstName;
      user.lasttName=lastName;
      user.password=password;
      users.push((await con.manager.save(user)) as UserEntity);
    }}
    export {createUsers}
```

5- Execution

```
SELECT * FROM `user`
```

Profilage [Éditer en ligne] [Éditer] [Expliquer SQL] [Créer le code source PHP] [Actualiser]

Tout afficher | Nombre de lignes : 25 Filtrer les lignes: Chercher dans cette table Trier par clé : Aucun(e)

+ Options

	id	firstName	lasttName	birthDate	email	password
Éditer Copier Supprimer	1	Florine	Ortiz	2021-07-04 07:26:35	Jensen.Hahn@yahoo.com	VvDtoDmA2AyhbJK
Éditer Copier Supprimer	2	Macey	Batz	2021-07-07 16:22:49	Toy.Nienow@gmail.com	nuMqCcQqXGM2iiM
Éditer Copier Supprimer	3	Paige	Stanton	2021-06-25 21:41:13	Imelda_Towne89@yahoo.com	u1WiT_I_Mmz5NoD
Éditer Copier Supprimer	4	Filomena	Lehner	2022-02-21 07:59:41	Simone_Murphy@yahoo.com	a0Pz87iozmT4fD6
Éditer Copier Supprimer	5	Petra	Lubowitz	2022-04-07 06:52:49	Maci0@gmail.com	FJ6wjbg83Luue0l
Éditer Copier Supprimer	6	Flo	Gusikowski	2021-11-17 15:37:02	Cleta.Rosenbaum@gmail.com	LQtU8sHudyq8rzl
Éditer Copier Supprimer	7	Vivien	Schulist	2021-06-26 22:06:44	Derek.Jaskolski@hotmail.com	HRywm1baAq3aaju
Éditer Copier Supprimer	8	Tony	Keeling	2021-05-09 10:53:48	Dannie.Kunde@yahoo.com	yTqCvnXylD8SsFA
Éditer Copier Supprimer	9	Trace	Hamill	2021-04-19 18:31:37	Erna.Mann15@hotmail.com	VlzUEPPVS_0oJli
Éditer Copier Supprimer	10	Isom	Toy	2021-08-26 05:08:17	Earline_Kutch16@yahoo.com	3717DzJCZSf6CyK

Merci pour votre attention!