

NestJS in a nutshell

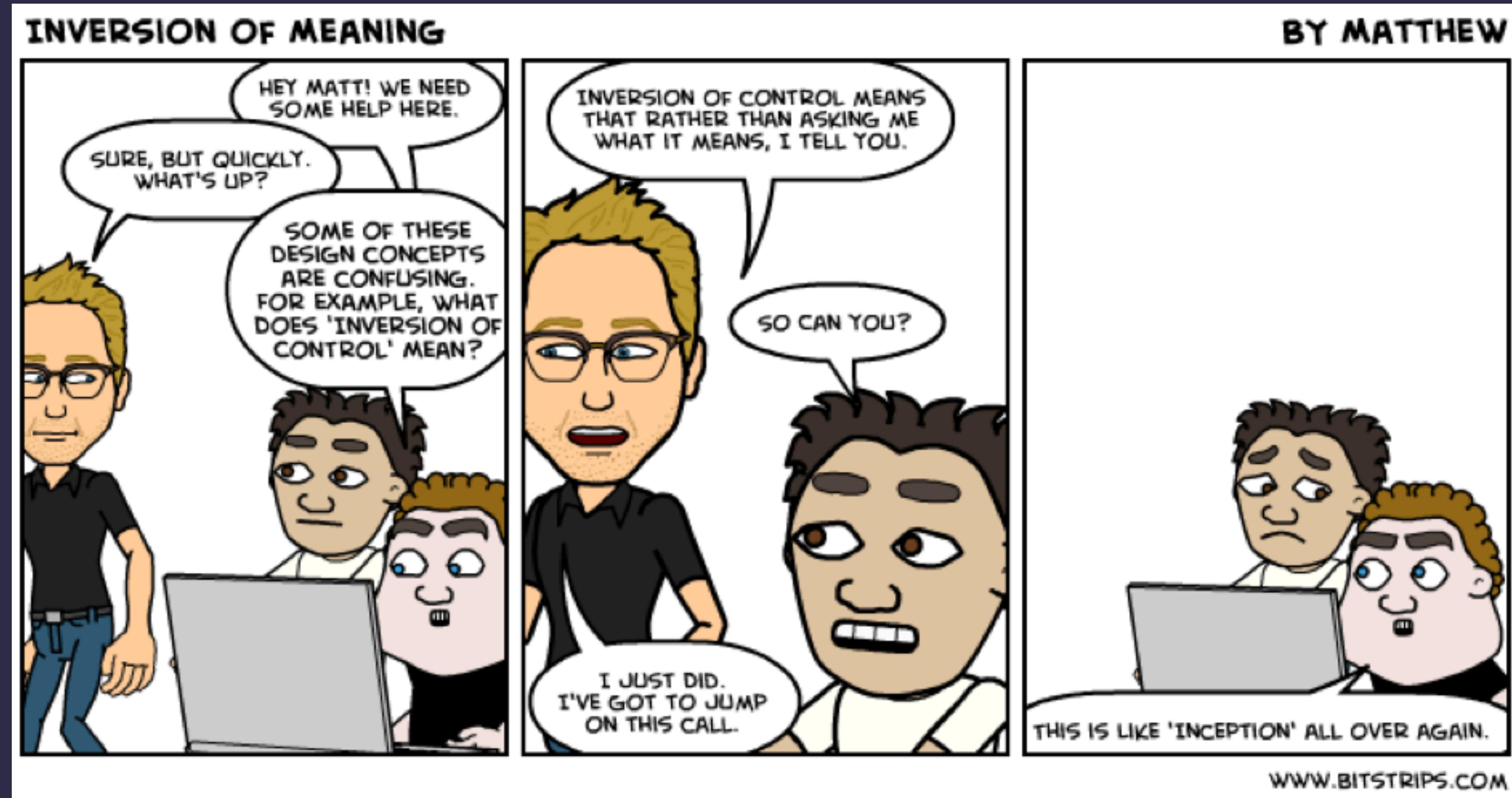




Natchapol Srisang (Beam)

- CP 44
- Former president of Thinc.
- Backend engineer @Brikl
- Personal blog at utopiabeam.dev

Inversion of Control (IoC)



Dependency Injection (DI)



```
class Foo {  
    private bar: Bar  
    private baz: Baz  
  
    constructor() {  
        this.bar = new Bar()  
        this.baz = new Baz()  
    }  
}
```



```
class Foo {  
    private bar: Bar  
    private baz: Baz  
  
    constructor(bar: Bar, baz: Baz) {  
        this.bar = bar  
        this.baz = baz  
    }  
}
```

Why DI?



```
class Foo {  
    private bar: Bar  
    private baz: Baz  
  
    constructor(bar: Bar, baz: Baz) {  
        this.bar = bar  
        this.baz = baz  
    }  
}
```

Foo doesn't care how to
create a Bar and Baz object.

Loose coupling!



Let ' s make an
online shopping
API project .



Quick start

1. Install CLI

```
$ npm i -g @nestjs/cli  
or  
$ yarn global add @nestjs/cli
```

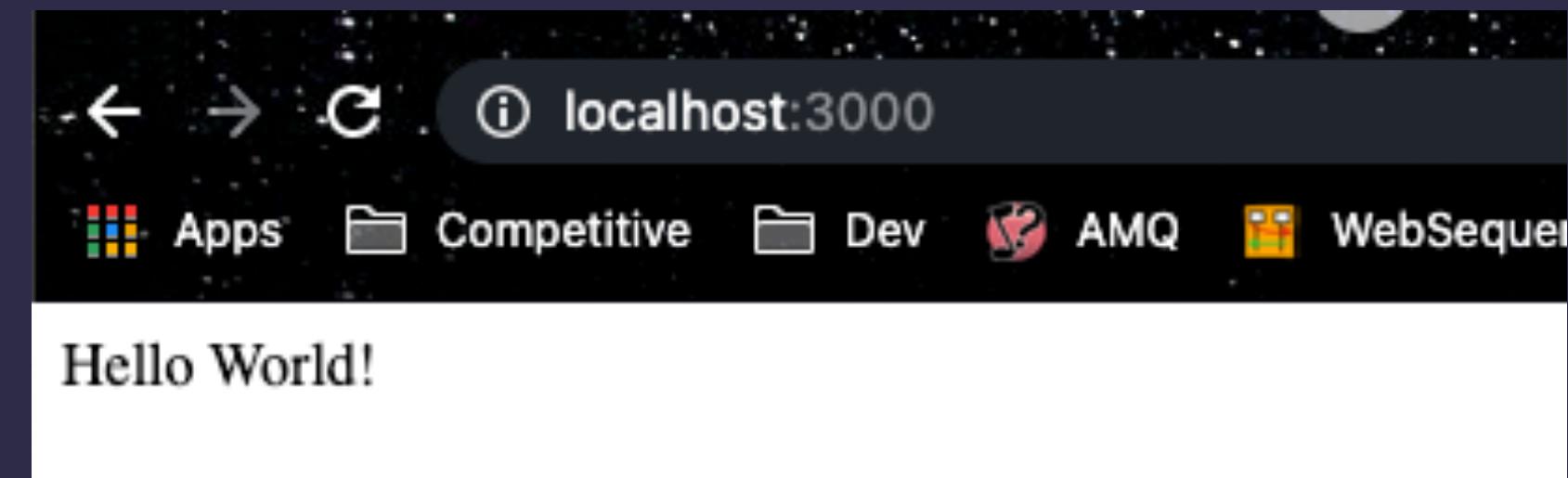
2. Create a project

```
$ nest new shoper-spi
```

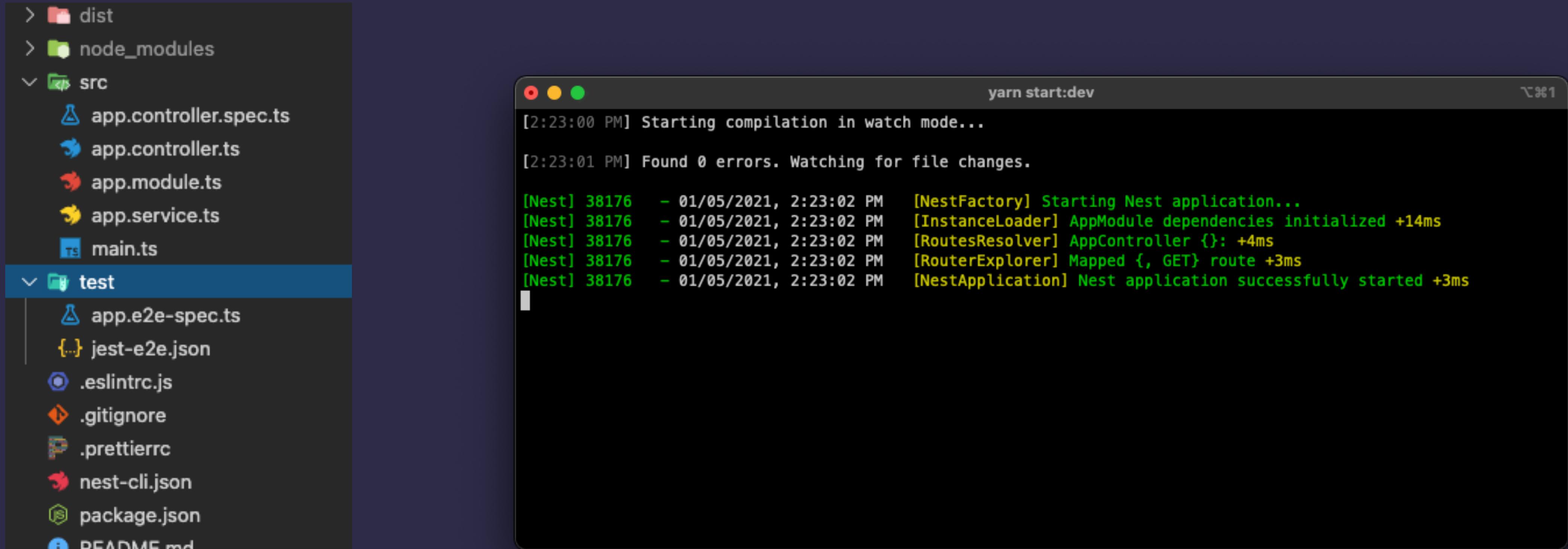
3. Run

```
$ cd shoper-api  
$ npm run start:dev  
or  
$ yarn start:dev
```

4. Go to <http://localhost:3000>



Quick start



The terminal window shows the command `yarn start:dev` running, with the following log output:

```
[2:23:00 PM] Starting compilation in watch mode...
[2:23:01 PM] Found 0 errors. Watching for file changes.
[Nest] 38176 - 01/05/2021, 2:23:02 PM [NestFactory] Starting Nest application...
[Nest] 38176 - 01/05/2021, 2:23:02 PM [InstanceLoader] AppModule dependencies initialized +14ms
[Nest] 38176 - 01/05/2021, 2:23:02 PM [RoutesResolver] AppController {}: +4ms
[Nest] 38176 - 01/05/2021, 2:23:02 PM [RouterExplorer] Mapped {, GET} route +3ms
[Nest] 38176 - 01/05/2021, 2:23:02 PM [NestApplication] Nest application successfully started +3ms
```

The file explorer shows the project structure:

- > dist
- > node_modules
- < src
 - app.controller.spec.ts
 - app.controller.ts
 - app.module.ts
 - app.service.ts
 - main.ts
- < test
 - app.e2e-spec.ts
 - jest-e2e.json
 - .eslintrc.js
 - .gitignore
 - .prettierrc
 - nest-cli.json
 - package.json
 - README.md
 - tsconfig.build.json
 - tsconfig.json
 - yarn.lock

main.ts



```
import { NestFactory } from '@nestjs/core';
import { AppModule } from './app.module';

async function bootstrap() {
  const app = await NestFactory.create(AppModule);
  await app.listen(3000);
}
bootstrap();
```



```
import express from 'express'

const app = express()
app.use(express.json())
app.listen(3000)
```

app.controller.ts



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

@Controller()
export class AppController {
  constructor(private readonly appService: AppService) {}

  @Get()
  getHello(): string {
    return this.appService.getHello();
  }
}
```



```
app.get('/', (_, res: Response) => {
  res.send(getHello())
})
```

app.service.ts



```
import { Injectable } from '@nestjs/common';

@Injectable()
export class AppService {
  getHello(): string {
    return 'Hello World!';
  }
}
```

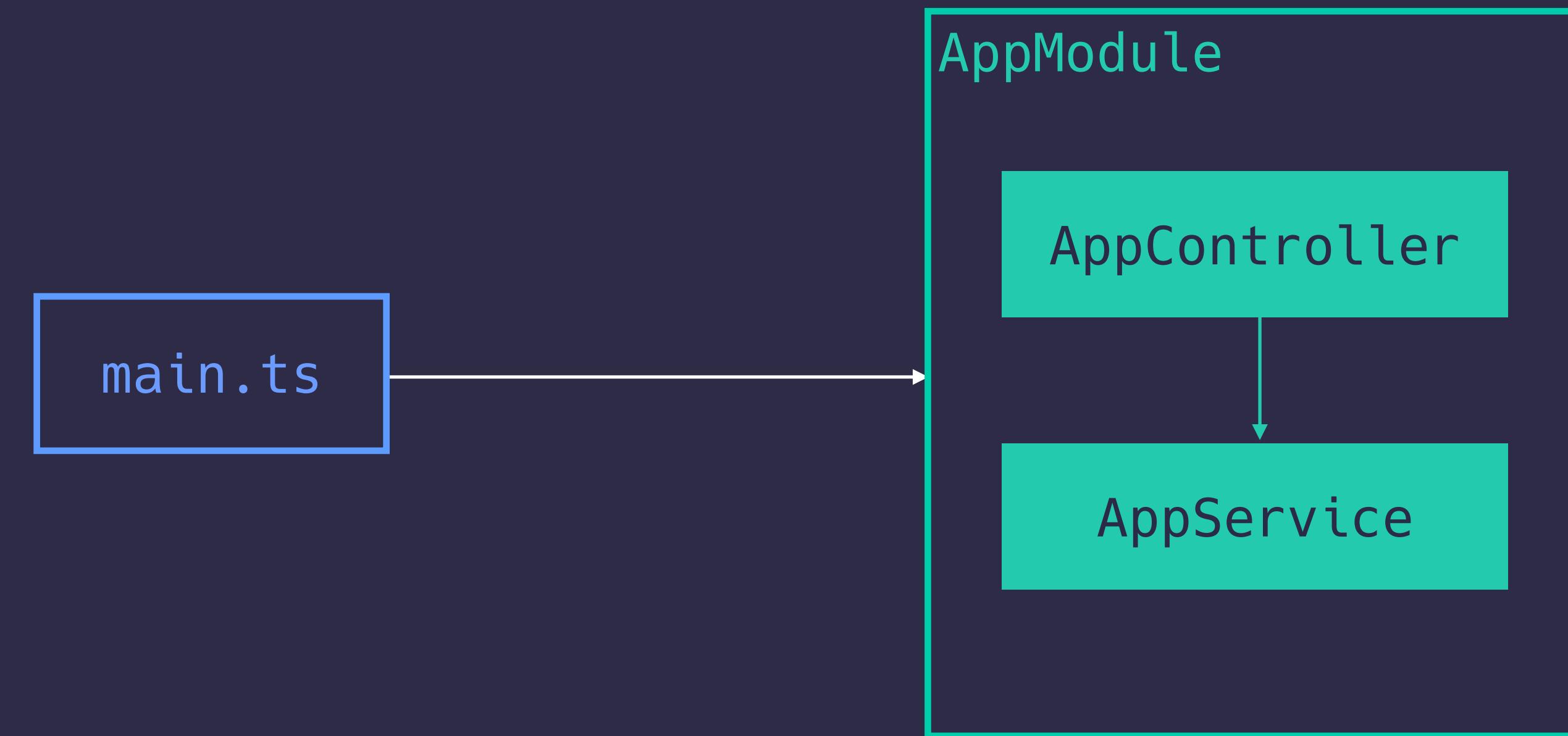
app.module.ts



```
import { Module } from '@nestjs/common';
import { AppController } from './app.controller';
import { AppService } from './app.service';

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

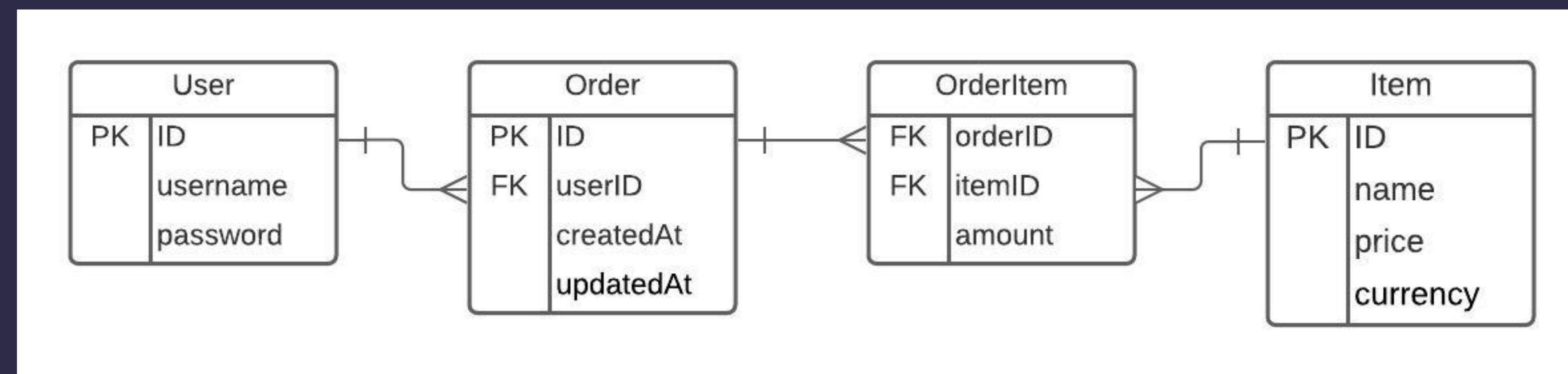
Overview



Let's work on
the database.



ER diagram



Setup DB

1. Create `docker-compose.yml`

2. Start it

```
$ docker-compose up -d
```

3. Check if it's running or not

```
~/Projects/nestjs-in-a-nutshell-talk(master) $ docker-compose ps
          Name      Command     State        Ports
nestjs-in-a-nutshell-talk_mysql_1   docker-entrypoint.sh mysqld   Up      0.0.0.0:3306->3306/tcp, 33060/tcp
```

```
version: '3.7'

services:
  mysql:
    image: mysql:5.7
    volumes:
      - mysql_data:/var/lib/mysql
    environment:
      MYSQL_ROOT_PASSWORD: password
      MYSQL_DATABASE: shoper
      MYSQL_USER: user
      MYSQL_PASSWORD: password
    ports:
      - '3306:3306'

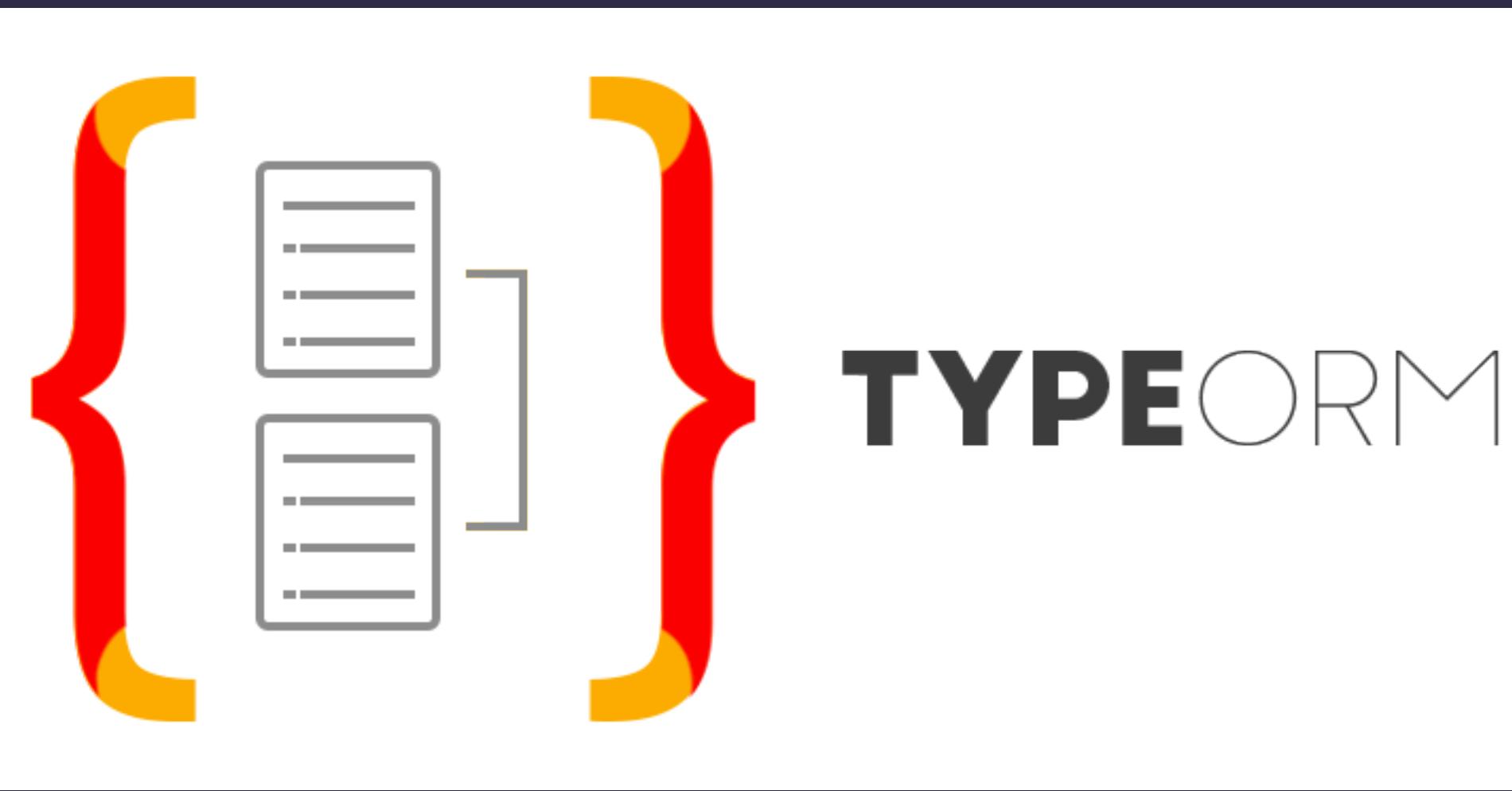
volumes:
  mysql_data:
```

`docker-compose.yml`

Connect DB

1. Install dependencies

```
$ yarn add @nestjs/typeorm typeorm mysql
```



2. Add TypeOrmModule



```
import { Module } from '@nestjs/common';
import { TypeOrmModule } from '@nestjs/typeorm';

@Module({
  imports: [
    TypeOrmModule.forRoot({
      type: 'mysql',
      host: 'localhost',
      port: 3306,
      username: 'user',
      password: 'password',
      database: 'shoper',
      entities: [],
      synchronize: true,
    }),
  ],
})
export class AppModule {}
```

Database config

Entities

1. Create entities directory
2. Create `user.entity.ts` inside it
3. Write schema



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

@Entity()
export class User {
    @PrimaryGeneratedColumn()
    id: number;

    @Column()
    username: string;

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

`user.entity.ts`

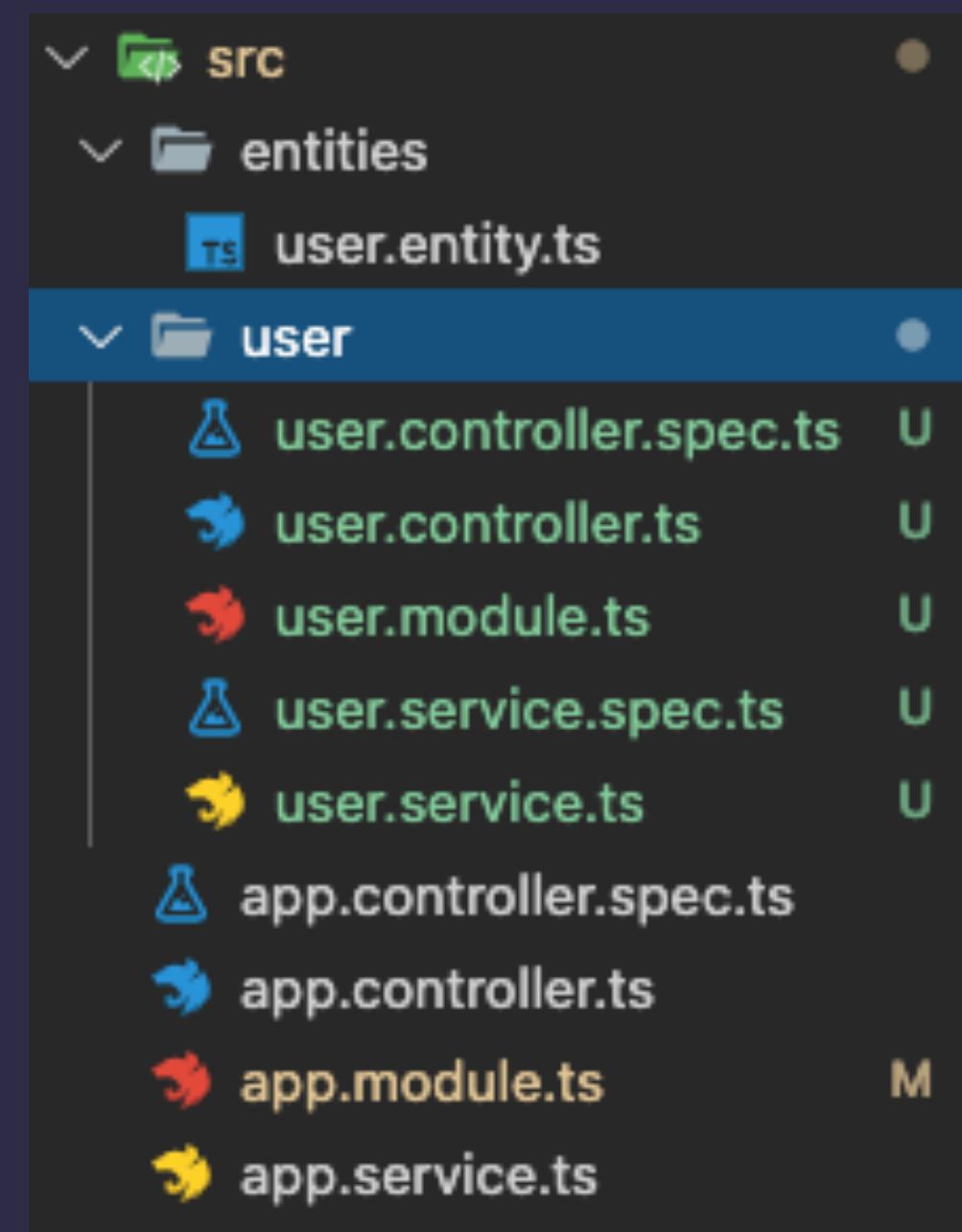
Create user module

1. Create user module
2. Create user controller
3. Create user service



```
$ nest g module user  
$ nest g controller user  
$ nest g service user
```

```
apple ~ ~/Projects/nestjs-in-a-nutshell-talk ✘ master ?1  
› nest g mo user  
CREATE src/user/user.module.ts (81 bytes)  
UPDATE src/app.module.ts (656 bytes)  
  
apple ~ ~/Projects/nestjs-in-a-nutshell-talk ✘ master !1 ?1  
› nest g co user  
CREATE src/user/user.controller.spec.ts (478 bytes)  
CREATE src/user/user.controller.ts (97 bytes)  
UPDATE src/user/user.module.ts (166 bytes)  
  
apple ~ ~/Projects/nestjs-in-a-nutshell-talk ✘ master !1 ?1  
› nest g s user  
CREATE src/user/user.service.spec.ts (446 bytes)  
CREATE src/user/user.service.ts (88 bytes)  
UPDATE src/user/user.module.ts (240 bytes)
```



Implement user service

Inject repository

```
● ● ●  
import { Injectable } from '@nestjs/common';
import { InjectRepository } from '@nestjs/typeorm';
import { User } from 'src/entities/user.entity';
import { Repository } from 'typeorm';  
  
@Injectable()
export class UserService {  
  constructor(  
    @InjectRepository(User) private readonly repo: Repository<User>,  
  ) {}  
  
  findById(id: number): Promise<User> {  
    return this.repo.findOne(id);  
  }  
  
  create(dto: Omit<User, 'id'>): Promise<User> {  
    const user = { ...new User(), ...dto };  
    return this.repo.save(user);  
  }  
  
  async update(id: number, dto: Partial<Omit<User, 'id'>>): Promise<User> {  
    const user = { ...(await this.findById(id)), ...dto };  
    return this.repo.save(user);  
  }  
  
  async delete(id: number) {  
    const user = await this.findById(id);  
    await this.repo.remove(user);  
    return user;  
  }  
}
```

CRUD operations
using repository

```
[Nest] 54130 - 01/06/2021, 10:48:31 AM [ExceptionHandler] Nest can't resolve dependencies of the UserService (?). Please make sure that the argument UserRepository at index [0] is available in the UserModule context.  
Potential solutions:  
- If UserRepository is a provider, is it part of the current UserModule?  
- If UserRepository is exported from a separate @Module, is that module imported within UserModule?  
  @Module({  
    imports: [ /* the Module containing UserRepository */ ]  
  })  
+0ms
```

Inject repository



```
import { Module } from '@nestjs/common';
import { TypeOrmModule } from '@nestjs/typeorm';
import { User } from 'src/entities/user.entity';
import { UserController } from './user.controller';
import { UserService } from './user.service';

@Module({
  imports: [TypeOrmModule.forFeature([User])],
  controllers: [UserController],
  providers: [UserService],
})
export class UserModule {}
```

Inject repository

```
[Nest] 54283 - 01/06/2021, 10:51:57 AM [ExceptionHandler] No repository for "User" was found. Looks like th  
is entity is not registered in current "default" connection? +1ms
```



imgflip.com



imgflip.com

Register entities

Register every entities in
*.entity.ts or *.entity.js
files.

```
import { Module } from '@nestjs/common';
import { TypeOrmModule } from '@nestjs/typeorm';
import { join } from 'path';
import { AppController } from './app.controller';
import { AppService } from './app.service';
import { UserModule } from './user/user.module';
```

```
@Module({
  imports: [
    TypeOrmModule.forRoot({
      type: 'mysql',
      host: 'localhost',
      port: 3306,
      username: 'user',
      password: 'password',
      database: 'shoper',
      entities: [join(__dirname, '**/*.entity.{ts,js}')],
      synchronize: true,
    }),
    UserModule,
  ],
  controllers: [AppController],
  providers: [AppService],
})
export class AppModule {}
```

Implement controller

Inject UserService

GET /user/:id

POST /user

PATCH /user/:id

DELETE /user/:id

```
import {  
  Body,  
  Controller,  
  Delete,  
  Get,  
  Param,  
  ParseIntPipe,  
  Patch,  
  Post,  
} from '@nestjs/common';  
import { User } from 'src/entities/user.entity';  
import { UserService } from './user.service';  
  
@Controller('user')  
export class UserController {  
  constructor(private readonly service: UserService) {}  
  
  @Get(':id')  
  findById(@Param('id', new ParseIntPipe()) id: number): Promise<User> {  
    return this.service.findById(id);  
  }  
  
  @Post()  
  create(@Body() dto: Omit<User, 'id'>): Promise<User> {  
    return this.service.create(dto);  
  }  
  
  @Patch(':id')  
  update(  
    @Param('id', new ParseIntPipe()) id: number,  
    @Body() dto: Partial<Omit<User, 'id'>>,  
  ): Promise<User> {  
    return this.service.update(id, dto);  
  }  
  
  @Delete(':id')  
  delete(@Param('id', new ParseIntPipe()) id: number): Promise<User> {  
    return this.service.delete(id);  
  }  
}
```

Implement entities



```
import { Column, Entity, OneToMany, PrimaryGeneratedColumn } from 'typeorm';
import { Order } from './order.entity';

@Entity()
export class User {
    @PrimaryGeneratedColumn()
    id: number;

    @Column()
    username: string;

    @Column()
    password: string;

    @OneToMany(
        () => Order,
        order => order.user,
    )
    orders: Order[];
}
```

user.entity.ts



```
import {
    Column,
    CreateDateColumn,
    Entity,
    JoinColumn,
    ManyToOne,
    OneToMany,
    PrimaryGeneratedColumn,
    UpdateDateColumn,
} from 'typeorm';
import { OrderItem } from './order-item.entity';
import { User } from './user.entity';

@Entity()
export class Order {
    @PrimaryGeneratedColumn()
    id: number;

    @Column('int')
    userId: number;

    @CreateDateColumn()
    createdAt: Date;

    @UpdateDateColumn()
    updatedAt: Date;

    @ManyToOne(
        () => User,
        user => user.orders,
    )
    user: User;

    @OneToMany(
        () => OrderItem,
        orderItem => orderItem.order,
    )
    orderItems: OrderItem[];
}
```

order.entity.ts

Implement entities



```
import { Column, Entity, ManyToOne, PrimaryColumn } from 'typeorm';
import { Item } from './item.entity';
import { Order } from './order.entity';

@Entity()
export class OrderItem {
    @PrimaryColumn()
    orderId: number;

    @PrimaryColumn()
    itemId: number;

    @Column('int')
    amount: number;

    @ManyToOne(
        () => Order,
        order => order.orderItems,
    )
    order: Order;

    @ManyToOne(() => Item)
    item: Item;
}
```

order-item.entity.ts



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

export enum Currency {
    BHT = 'BHT',
    USD = 'USD',
}

@Entity()
export class Item {
    @PrimaryGeneratedColumn()
    id: number;

    @Column()
    name: string;

    @Column('float')
    price: number;

    @Column('enum', { enum: Currency })
    currency: Currency;
}
```

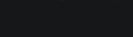
item.entity.ts

Implement item module



```
import { Module } from '@nestjs/common';
import { TypeOrmModule } from '@nestjs/typeorm';
import { Item } from 'src/entities/item.entity';
import { ItemController } from './item.controller';
import { ItemService } from './item.service';

@Module({
  imports: [TypeOrmModule.forFeature([Item])],
  controllers: [ItemController],
  providers: [ItemService],
})
export class ItemModule {}
```



```
import {  
  Body,  
  Controller,  
  Delete,  
  Get,  
  Param,  
  ParseIntPipe,  
  Patch,  
  Post,  
} from '@nestjs/common';  
import { Item } from 'src/entities/item.entity';  
import { ItemService } from './item.service';  
  
@Controller('item')  
export class ItemController {  
  constructor(private readonly service: ItemService) {}  
  
  @Get(':id')  
  findById(@Param('id', new ParseIntPipe()) id: number): Promise<Item> {  
    return this.service.findById(id);  
  }  
  
  @Post()  
  create(@Body() dto: Omit<Item, 'id'>): Promise<Item> {  
    return this.service.create(dto);  
  }  
  
  @Patch(':id')  
  update(  
    @Param('id', new ParseIntPipe()) id: number,  
    @Body() dto: Partial<Omit<Item, 'id'>,  
  ): Promise<Item> {  
    return this.service.update(id, dto);  
  }  
  
  @Delete(':id')  
  delete(@Param('id', new ParseIntPipe()) id: number): Promise<Item> {  
    return this.service.delete(id);  
  }  
}
```

item.controller.ts



```
import { Injectable } from '@nestjs/common';  
import { InjectRepository } from '@nestjs/typeorm';  
import { Item } from 'src/entities/item.entity';  
import { Repository } from 'typeorm';  
  
@Injectable()  
export class ItemService {  
  constructor(  
    @InjectRepository(Item) private readonly repo: Repository<Item>,  
  ) {}  
  
  find(): Promise<Item[]> {  
    return this.repo.find();  
  }  
  
  findById(id: number): Promise<Item> {  
    return this.repo.findOne(id);  
  }  
  
  create(dto: Omit<Item, 'id'>): Promise<Item> {  
    const item = { ...new Item(), ...dto };  
    return this.repo.save(item);  
  }  
  
  async update(id: number, dto: Partial<Omit<Item, 'id'>>): Promise<Item> {  
    const item = { ...(await this.findById(id)), ...dto };  
    return this.repo.save(item);  
  }  
  
  async delete(id: number): Promise<Item> {  
    const item = await this.findById(id);  
    await this.repo.remove(item);  
    return item;  
  }  
}
```

item.service.ts

Implement order module



```
import { Module } from '@nestjs/common';
import { TypeOrmModule } from '@nestjs/typeorm';
import { Order } from 'src/entities/order.entity';
import { OrderController } from './order.controller';
import { OrderService } from './order.service';

@Module({
  imports: [TypeOrmModule.forFeature([Order])],
  controllers: [OrderController],
  providers: [OrderService],
})
export class OrderModule {}
```

```

import {
  Body,
  Controller,
  Delete,
  Get,
  Param,
  ParseIntPipe,
  Patch,
  Post,
} from '@nestjs/common';
import { Order } from 'src/entities/order.entity';
import { OrderService } from './order.service';

@Controller('order')
export class OrderController {
  constructor(private readonly service: OrderService) {}

  @Get(':id')
  findById(@Param('id', new ParseIntPipe()) id: number): Promise<Order> {
    return this.service.findById(id);
  }

  @Post()
  create(@Body() dto: Omit<Order, 'id'>): Promise<Order> {
    return this.service.create(dto);
  }

  @Patch(':id')
  update(
    @Param('id', new ParseIntPipe()) id: number,
    @Body() dto: Partial<Omit<Order, 'id'>>,
  ): Promise<Order> {
    return this.service.update(id, dto);
  }

  @Delete(':id')
  delete(@Param('id', new ParseIntPipe()) id: number): Promise<Order> {
    return this.service.delete(id);
  }
}

```

order.controller.ts

```

import { Injectable } from '@nestjs/common';
import { InjectRepository } from '@nestjs/typeorm';
import { Order } from 'src/entities/order.entity';
import { Repository } from 'typeorm';

@Injectable()
export class OrderService {
  constructor(
    @InjectRepository(Order) private readonly repo: Repository<Order>,
  ) {}

  find(): Promise<Order[]> {
    return this.repo.find();
  }

  findById(id: number): Promise<Order> {
    return this.repo.findOne(id);
  }

  create(dto: Omit<Order, 'id'>): Promise<Order> {
    const order = { ...new Order(), ...dto };
    return this.repo.save(order);
  }

  async update(id: number, dto: Partial<Omit<Order, 'id'>>): Promise<Order> {
    const Order = { ...(await this.findById(id)), ...dto };
    return this.repo.save(Order);
  }

  async delete(id: number): Promise<Order> {
    const order = await this.findById(id);
    await this.repo.remove(order);
    return order;
  }
}

```

order.service.ts