**📁 interfaces/ — Centralized Type Definitions**

**📌 Purpose**

The interfaces/ directory is dedicated to housing TypeScript interfaces that define the shape of data objects used throughout your applicatio. By centralizing these definitions, you promote consistency, reusability, and maintainabilit. Interfaces serve as contracts, ensuring that different parts of your application adhere to expected data structure.

**🗂️ Suggested Directory Structur**



src/

└── interfaces/

├── user.interface.ts

├── product.interface.ts

├── order.interface.ts

└── index.ts

``



- \*\*`user.interface.ts`\*: Defines the structure of a User objet.

- \*\*`product.interface.ts`\*: Defines the structure of a Product objet.

- \*\*`order.interface.ts`\*: Defines the structure of an Order objet.

- \*\*`index.ts`\*: Barrel file to export all interfaces for easy import elsewhere in the applicatin.

### 🧩 Implementation Examples

\*\*1. `user.interface.ts`\*



```typescript

// src/interfaces/user.interface.ts

/\*\*

\* Represents a user in the system.

\*/

export interface User {

/\*\*

\* Unique identifier for the user.

\*/

id: number;

/\*\*

\* Full name of the user.

\*/

name: string;

/\*\*

\* Email address of the user.

\*/

email: string;

/\*\*

\* Indicates whether the user's email is verified.

\*/

isEmailVerified: boolean;

}

``



\*\*2. `product.interface.ts`\*



```typescript

// src/interfaces/product.interface.ts

/\*\*

\* Represents a product available for purchase.

\*/

export interface Product {

/\*\*

\* Unique identifier for the product.

\*/

id: number;

/\*\*

\* Name of the product.

\*/

name: string;

/\*\*

\* Description of the product.

\*/

description: string;

/\*\*

\* Price of the product in USD.

\*/

price: number;

/\*\*

\* Quantity of the product available in stock.

\*/

stock: number;

}

``



\*\*3. `order.interface.ts`\*



```typescript

// src/interfaces/order.interface.ts

import { User } from './user.interface';

import { Product } from './product.interface';

/\*\*

\* Represents an order placed by a user.

\*/

export interface Order {

/\*\*

\* Unique identifier for the order.

\*/

id: number;

/\*\*

\* The user who placed the order.

\*/

user: User;

/\*\*

\* List of products included in the order.

\*/

products: Product[];

/\*\*

\* Total amount for the order.

\*/

totalAmount: number;

/\*\*

\* Date when the order was placed.

\*/

orderDate: Date;

}

``



\*\*4. `index.ts`\*



```typescript

// src/interfaces/index.ts

export \* from './user.interface';

export \* from './product.interface';

export \* from './order.interface';

``



### 🔗 Integration into Other Modules or Services

To utilize these interfaces in other parts of your application, you can import them directly from the `interfaces` directory.

\*\*Example: Using `User` Interface in a Servic\*\*



```typescript

// src/modules/users/users.service.ts

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

import { User } from '../../interfaces';

@Injectable()

export class UsersService {

private users: User[] = [];

/\*\*

\* Retrieves all users.

\* @returns Array of users.

\*/

findAll(): User[] {

return this.users;

}

/\*\*

\* Creates a new user.

\* @param user - User object to create.

\* @returns The created user.

\*/

create(user: User): User {

this.users.push(user);

return user;

}

}

``



\*\*Example: Using `Product` Interface in a Controlle\*\*



```typescript

// src/modules/products/products.controller.ts

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

import { Product } from '../../interfaces';

@Controller('products')

export class ProductsController {

private products: Product[] = [];

/\*\*

\* Retrieves all products.

\* @returns Array of products.

\*/

@Get()

findAll(): Product[] {

return this.products;

}

}

``



### ✅ Best Practices

- \*\*Consistenc\*\*: Use interfaces to define consistent data structures across your applicaion.

- \*\*Reusabilit\*\*: Centralize interfaces to promote reuse and reduce duplicaion.

- \*\*Clarit\*\*: Provide detailed comments and descriptions for each property to enhance code readabiity.

- \*\*Maintenanc\*\*: Update interfaces in one place to reflect changes across the applicaion.

---