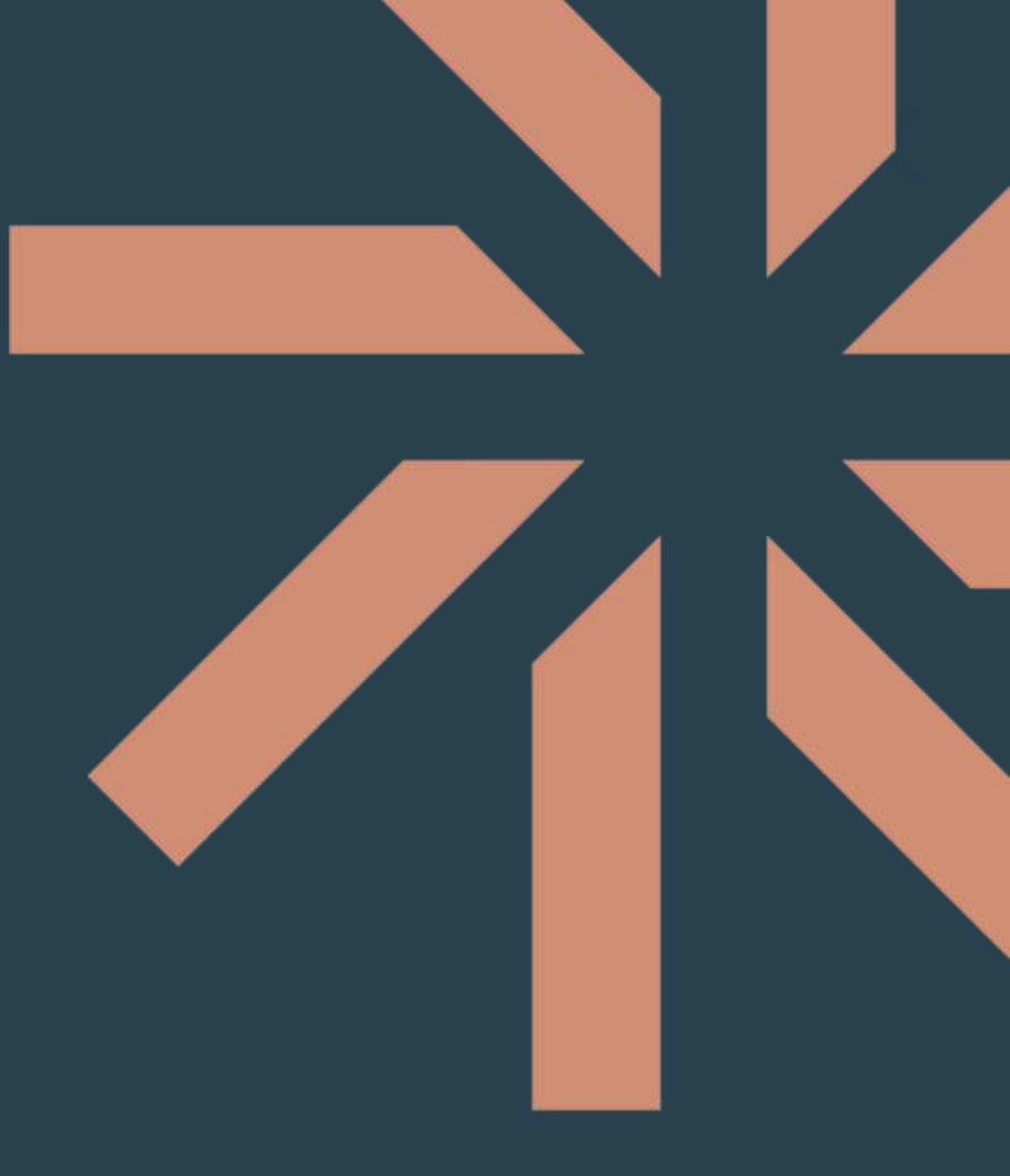
TypeORM

Qinshift Academy

What is TypeORM?

TypeORM is a TypeScript-first ORM (Object Relational Mapper) that allows developers to interact with relational databases like PostgreSQL, MySQL, SQLite, and others using object-oriented code instead of writing raw SQL.

- Designed to work seamlessly with TypeScript
- Fully supports decorators, entities, migrations, and relations
- Integrates perfectly with NestJS



Why Use TypeORM?

- Simplifies database interaction
- Enables type-safe database access
- Maps database tables to classes and objects
- Automatically creates SQL queries behind the scenes
- Supports relations, eager/lazy loading, and cascades

Core Concepts

Concept	Description
Entity	A class that maps to a table in the database
Repository	A class that handles all database operations for an entity
DataSource	The main entry point for database configuration
Migration	A saved change to the database structure
Relations	Links between tables (OneToOne, OneToMany, ManyToMany)

Entity: How it works

When you define a class and decorate it with @Entity(), TypeORM knows that this class should be stored as a table in the database. The fields of the class become the columns.

Behind the scenes, TypeORM uses reflection metadata and decorators to understand how to map the class to the actual SQL table.

Key Decorators Used in Entities

- @Entity(): Marks the class as a database entity (table)
- @PrimaryGeneratedColumn(): Creates a primary key column with auto-increment
- @Column(): Declares a column and its optional config
- @CreateDateColumn() / @UpdateDateColumn(): Automatically managed timestamps
- @OneToMany(), @ManyToOne(), etc.: Define relationships with other entities

Additional Notes

- Entities can include methods — helpful for computed values or formatting.
- If “synchronize: true” is enabled in the config, TypeORM will auto-create tables based on entities.
- Entities are class-first, so you work with JavaScript objects in code, not raw SQL.
- They are used with repositories to interact with the DB (e.g., repo.find(), repo.save()).

What is a Custom Repository?

A custom repository in TypeORM is a class where you define custom database queries and logic, tailored to a specific entity. It builds on top of the standard Repository<Entity> provided by TypeORM, but lets you extend behavior with your own methods.

- While the standard repository (Repository<T>) gives you access to common methods like .find(), .save(), etc., a custom repository is useful when you:
- Need custom queries or complex filtering
- Want to organize your business logic around data access
- Prefer a cleaner separation between persistence and service logic

Summary

An Entity in TypeORM:

- Is a class that maps directly to a table
- Uses decorators to describe database structure
- Makes database interaction object-oriented
- Enables automatic schema generation and ORM functionality

Questions?

Trainer Name

Trainer

trainer@mail.com

Assistant Name

Assistant

asistant@mail.com

Qinshift Academy