# Instructions for Integrating Fastify, GraphQL, and TypeORM

## Step 1: Install Dependencies

Install the required packages using npm:

```
npm install fastify mercurius typeorm reflect-metadata sqlite3 graphql
```

## Step 2: Configure TypeORM

Create a file named `ormconfig.json` with the following content:

```
{
  "type": "sqlite",
  "database": "./db.sqlite",
  "synchronize": true,
  "logging": false,
  "entities": ["src/entity/**/*.ts"],
  "migrations": ["src/migration/**/*.ts"],
  "subscribers": ["src/subscriber/**/*.ts"]
}
```

Next, create entities in the `src/entity` folder. Example of `User.ts`:

```
import { Entity, PrimaryGeneratedColumn, Column, OneToMany } from ’typeorm’;
import { Post } from ’./Post’;

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

  @Column()
  name: string;

  @OneToMany(() => Post, (post) => post.author)
  posts: Post[];
}
```

Example of `Post.ts`:

```
import { Entity, PrimaryGeneratedColumn, Column, ManyToOne } from ’typeorm’;
import { User } from ’./User’;

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

  @Column()
  title: string;

  @ManyToOne(() => User, (user) => user.posts)
  author: User;
}
```

## Step 3: Configure Fastify with GraphQL

Create a file named `index.ts` with the following content:

```
import ’reflect-metadata’;
import { createConnection } from ’typeorm’;
import fastify from ’fastify’;
import mercurius from ’mercurius’;
import { User } from ’./entity/User’;
import { Post } from ’./entity/Post’;

const app = fastify();

const schema = ‘
  type User {
    id: ID!
    name: String!
    posts: [Post!]!
  }

  type Post {
    id: ID!
    title: String!
    author: User!
  }

  type Query {
    user(id: ID!): User
  }
‘;

const resolvers = {
  Query: {
    user: async (_, { id }) => {
      return await User.findOne({ where: { id }, relations: [’posts’] });
    },
  },
};

```

```
const startServer = async () => {
  await createConnection();

  app.register(mercurius, {
    schema,
    resolvers,
    graphiql: true,
  });

  app.listen({ port: 3000 }, (err, address) => {
    if (err) {
      console.error(err);
      process.exit(1);
    }
    console.log(‘     Server ready at ${address}‘);
  });
};

startServer();
```

## Step 4: Testing

1. Start the server:

```
npx ts-node index.ts
```

2. Open your browser at:

```
http://localhost:3000/graphiql
```

3. Send a sample GraphQL query:

```
query {
  user(id: 1) {
    name
    posts {
      title
    }
  }
}
```