

```
mkdir <nama_project>
```

```
cd <nama_project>
```

```
npm init --yes
```

```
npm install --save express esm mysql
```

```
npm install --save-dev nodemon dotenv
```

Ubah package.json pada bagian scripts

```
"start": "node index",  
"dev": "nodemon index || true",
```

buat file index.js pada folder project

```
require = require("esm")(module);  
module.exports = require("./src/app.js");
```

buat file .env pada folder project

```
APP_NAME=Hello World Database  
APP_PORT=3006  
DB_HOST=localhost  
DB_PORT=3306  
DB_USERNAME=root  
DB_PASSWORD=P@ssw0rd  
DB_NAME=enigma
```

buat file src > config.js

```
import dotenv from "dotenv";  
  
export default function configure() {  
  dotenv.config();  
  if (!process.env.APP_NAME) {  
    console.error('Environment file (.env) cannot  
    process.exit(1);  
  } else {  
    return {  
      appName: process.env.APP_NAME,  
      appPort: process.env.APP_PORT,  
      dbPort: process.env.DB_PORT,  
      dbHost: process.env.DB_HOST,  
      dbUser: process.env.DB_USERNAME,  
      dbPassword: process.env.DB_PASSWORD,  
      dbName: process.env.DB_NAME  
    };  
  }  
}
```

be found');

buat file src > database > connection.js

```
import mysql from 'mysql';
import configure from "../config";

const {dbHost, dbUser, dbPassword, dbName} = configure;
export const connection = mysql.createConnection({
  host: dbHost || 'localhost',
  user: dbUser || 'root',
  password: dbPassword || '',
  database: dbName || 'test',
});
```

buat file src > app.js

```
import configure from './config';
import express from "express";
import {connection} from "../database/connection";

configure();
connection.connect((err) => {
  if (err) {
    throw err
  }else{
    const app = express();
    app.listen(process.env.APP_PORT, () => {
      console.log(`${process.env.APP_NAME} listen
        ${process.env.APP_PORT}!`);
    });
  }
});
```

Coba jalankan **npm run dev**

);

ing on port

buat file src > middlewares > app-middleware.js



```
graph LR; A[buat file src > middlewares > app-middleware.js] --> B[import express from 'express';  
export default express.Router()  
.use(express.json());];
```

```
import express from 'express';  
export default express.Router()  
.use(express.json());
```

buat file src > repository > product.repository.js



```
import {connection} from '../database/connection';

export default class ProductRepository {

  findAllProduct() {
    return new Promise(function (resolve, reject) {
      connection.query('SELECT * from master')
        .then((err, rows, fields) => {
          if (err) {
            reject(err);
          } else {
            resolve(rows);
          }
        })
    });
  }
}
```

```
ct) {  
  _product',
```


buat file src > services > product.service.js



```
graph LR; A[buat file src > services > product.service.js] --> B[Code Block];
```

```
import ProductRepository from "../repository/productRepository";  
  
export default class ProductService {  
  
  async getAllProduct() {  
    return await new ProductRepository().findAll();  
  }  
}
```

```
t.repository";
```

```
lProduct());
```

buat file src > routes > product.route.js

```
import {Router} from 'express';
import ProductService from "../services/product.service";

const ProductRouter = Router()
  .get('/', async (req, res, next) => {
    const rows = await new ProductService().getA
    res.json(rows)
  });

export default ProductRouter;
```

buat file src > routes > index.js

```
import express from 'express';
import ProductRouter from "./product.router";

export default express.Router()
  .use('/product', ProductRouter)
  .use((req, res, next) => {
    res.status(404).json({message: 'Not Found.'})
  });
```

```
vice";
```

```
AllProduct();
```

```
});
```

ubah file src > app.js

```
import configure from './config';
import express from "express";
import AppMiddleware from './middlewares/app-middleware';
import AppRouter from './routes';
import {connection} from './database/connection';

configure();

connection.connect((err) => {
  if (err) {
    throw err
  }else{
    const app = express();
    app.use(AppMiddleware);
    app.use(AppRouter);
    app.listen(process.env.APP_PORT, () => {
      console.log(`${process.env.APP_NAME} liste
        ${process.env.APP_PORT}!`);
    });
  }
});
```

e';

ning on port

POST To Insert Data

ubah file src > repository > product.repository.js

```
findProductById(id) {  
  return new Promise(function (resolve, reject) {  
    connection.query('SELECT * from master_product where id=?', [id],  
      (err, rows, fields) => {  
        if (err) {  
          reject(err);  
        } else {  
          resolve(rows);  
        }  
      })  
  });  
}  
  
createProduct(product) {  
  return new Promise(function (resolve, reject) {  
    connection.query('insert into master_product(product_id,product_name,price  
      'values(?,?,?)', [product.productId, product.productName, product.price],  
      (err, result) => {  
        if (err) {  
          reject(err);  
        } else {  
          resolve(result);  
        }  
      })  
  });  
}
```

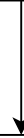
) ' +
el,

ubah file src > services > product.service.js



```
async getProductById(id) {  
    return await new ProductRepository().findProductById(id);  
}  
  
async createProduct(product) {  
    const result = await new ProductRepository().createProduct(product);  
    const newId = result.insertId;  
    return await this.getProductById(newId);  
}
```

ubah file src > routers > product.route.js



```
.post('/', async (req, res, next) => {  
  const product = { ... req.body};  
  const result = await new ProductService().createProduct(product);  
  res.json(result)  
});
```

Coba jalankan **npm run dev**

TypeORM

```
npm install --save typeorm
```

Buat file src > models > product.model.js

```
export default class Product {  
  constructor(id, productId, productName, price, c  
    this.id = id;  
    this.productId = productId;  
    this.productName = productName;  
    this.price = price;  
    this.categoryId = categoryId;  
  }  
}
```

Buat file src > models > category.model.js

```
export default class Category {  
  constructor(id, categoryId, categoryName) {  
    this.id = id;  
    this.categoryId = categoryId;  
    this.categoryName = categoryName;  
  }  
}
```

```
categoryId) {
```

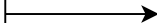
Buat file src > entities > product.schema.js



```
import {EntitySchema} from "typeorm";
import Product from "../models/product.model";

const ProductSchema = new EntitySchema({
  name: 'Product',
  target: Product,
  tableName: 'master_product',
  columns: {
    id: {
      primary: true,
      type: 'int',
      generated: true,
    },
    productId: {
      name: 'product_id',
      type: 'varchar'
    },
    productName: {
      name: 'product_name',
      type: 'varchar'
    }
  },
  relations: {
    category: {
      target: 'Category',
      type: 'many-to-one',
      eager: true,
      joinColumn: {name: 'category_id'}
    }
  }
});
export default ProductSchema
```

Buat file src > entities > category.schema.js



```
import {EntitySchema} from "typeorm";
import Category from "../models/category.model";

const CategorySchema = new EntitySchema({
  name: 'Category',
  target: Category,
  tableName: 'master_category',
  columns: {
    id: {
      primary: true,
      type: 'int',
      generated: true,
    },
    categoryId: {
      name: 'category_id',
      type: 'varchar'
    },
    categoryName: {
      name: 'category_name',
      type: 'varchar'
    }
  },
  relations: {
    product: {
      target: 'Product',
      type: 'one-to-many',
      inverseSide: 'category',
    }
  }
});
export default CategorySchema
```

Ubah file src > database > connection.js

```
import mysql from 'mysql';
import configure from "../config";
import {createConnection} from "typeorm";
import ProductSchema from "../entities/product.schema";
import CategorySchema from "../entities/category.schema";

const {dbType, dbHost, dbPort, dbUser, dbPassword, dbName} = configure();

const createDbConnection = async () => {
  const connection = await createConnection({
    type: dbType || 'mysql',
    host: dbHost || 'localhost',
    port: dbPort || 3306,
    username: dbUser || 'root',
    password: dbPassword || '',
    database: dbName || 'db_dban',
    entities: [ProductSchema, CategorySchema],
  });

  return connection;
};

export default createDbConnection;
```

Ubah file src > repository > category.repository.js

```
import {getRepository} from "typeorm";
import CategorySchema from "../entities/category.schema";

export default class CategoryRepository {
  categoryRepository() {
    return getRepository(CategorySchema);
  }

  async findOne(id) {
    const category = await this.categoryRepository().find({where: {id: id}}, {relations: ['product']});
    return category;
  }

  async findAllCategory() {
    const category = await this.categoryRepository().find();
    return category;
  }

  async findAllCategoryProduct() {
    const category = await this.categoryRepository().find({relations: ['product']});
    return category;
  }

  async createCategory(category) {
    return await this.categoryRepository().save(category);
  }
}
```



```
uct' ]});
```

Ubah file src > repository > product.repository.js

```
import {getRepository} from "typeorm";
import ProductSchema from "../entities/product.schema";

export default class ProductRepository {
  productRepository() {
    return getRepository(ProductSchema);
  }

  async findOne(id) {
    const product = await this.productRepository().find({where: {id: id}}, {relations: ['category']});
    return product;
  }

  async findAllProduct() {
    const product = await this.productRepository().find({relations: ['category']});
    return product;
  }

  async createProduct(product) {
    return await this.productRepository().save(product);
  }
}
```

```
ry' ]});
```

Ubah file src > app.js

```
import configure from './config';
import express from "express";
import AppMiddleware from './middlewares/app-middleware';
import AppRouter from './routes';
import createDbConnection from "./database/connection";

configure();

createDbConnection()
  .then((connection) => {
    if (connection.isConnected) {
      const app = express();
      app.use(AppMiddleware);
      app.use(AppRouter);
      app.listen(process.env.APP_PORT, () => {
        console.log(`${process.env.APP_NAME} listening on port
        ${process.env.APP_PORT}!`);
      });
    }
  })
  .catch((error) => {
    console.error(`Error starting up server.`);
    console.error(error);
  });
```

Ubah file src > config.js

```
import dotenv from "dotenv";

export default function configure() {
  dotenv.config();
  if (!process.env.APP_NAME) {
    console.error(`Environment file (.env) cannot be found in the root folder, copy .env.example to .env`);
    process.exit(1);
  } else {
    return {
      appName: process.env.APP_NAME,
      appPort: process.env.APP_PORT,
      dbType: process.env.DB_TYPE,
      dbPort: process.env.DB_PORT,
      dbHost: process.env.DB_HOST,
      dbUser: process.env.DB_USERNAME,
      dbPassword: process.env.DB_PASSWORD,
      dbName: process.env.DB_NAME
    };
  }
}
```

```
e file to .env.`);
```

Ubah file src > repository > product.repository.js

```
import {getConnection, getRepository} from "typeorm";  
  
...  
  
async updateProduct(id, product) {  
    return await getConnection()  
        .createQueryBuilder()  
        .update(Product)  
        .set(product)  
        .where("id = :id", {id: id})  
        .execute();  
}
```

Ubah file src > services > product.service.js

```
updateProduct(id, product) {  
  return new ProductRepository().updateProduct(id, product);  
}
```


Ubah file src > router > product.router.js

```
.put('/', async (req, res) => {  
  const {id, ...product} = req.body;  
  const result = await new ProductService().updateProduct(id, product);  
  res.json(result)  
});
```