Sequelize

Node | Express | Sequelize | MySQL

Example 01 SQL Database

Users

id	email	name
1	basit@gmail.com	Basit Hussain
2	ammar@test.com	Ammar Ahmed
3	ali@xyz.com	Ali

Products

id	title	price
1	Laptop	$1200
2	Smart Phone	$1000
3	Book	$400

Orders

id	user_id	product_id
1	2	2
2	1	2
3	3	3

{id: 1, name: ‘Afzal’}

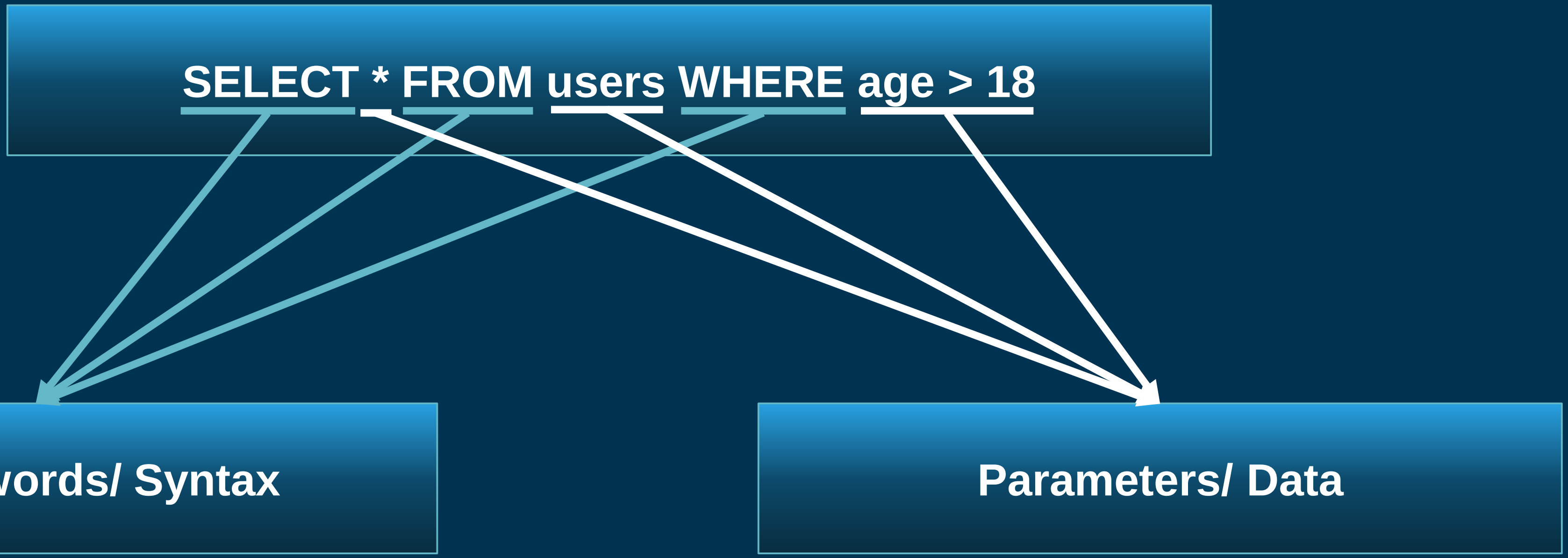
{id: 1, title: ‘Book’}

{1, {id: 1, name: ‘Afzal’}, {id: 1, title: ‘Book’}}

Data Relations

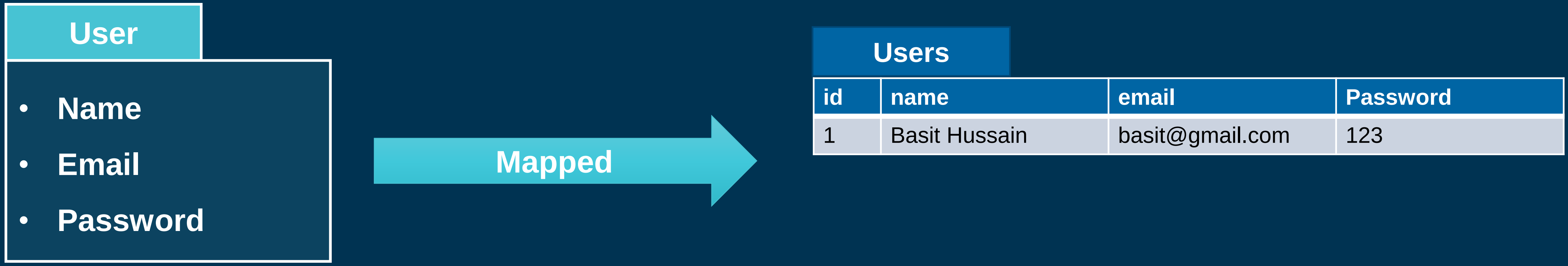
- One to One
- One to Many
- Many to Many

SQL Queries


SELECT * FROM users WHERE age > 18
SQL Keywords/ Syntax
Parameters/ Data

What is Sequelize

An **O**bject-**R**elational **M**apping Library



id	name	email	Password
1	Basit Hussain	basit@gmail.com	123

INSERT INTO users VALUES (1, ‘Basit Hussain’, ‘basit@gmail.com’, ‘123’)

Sequelize Core Concepts

Models	e.g. User, Products
Instances	const user = User.build()
Queries	User.findAll()
Associations	User.hasMany(Products)

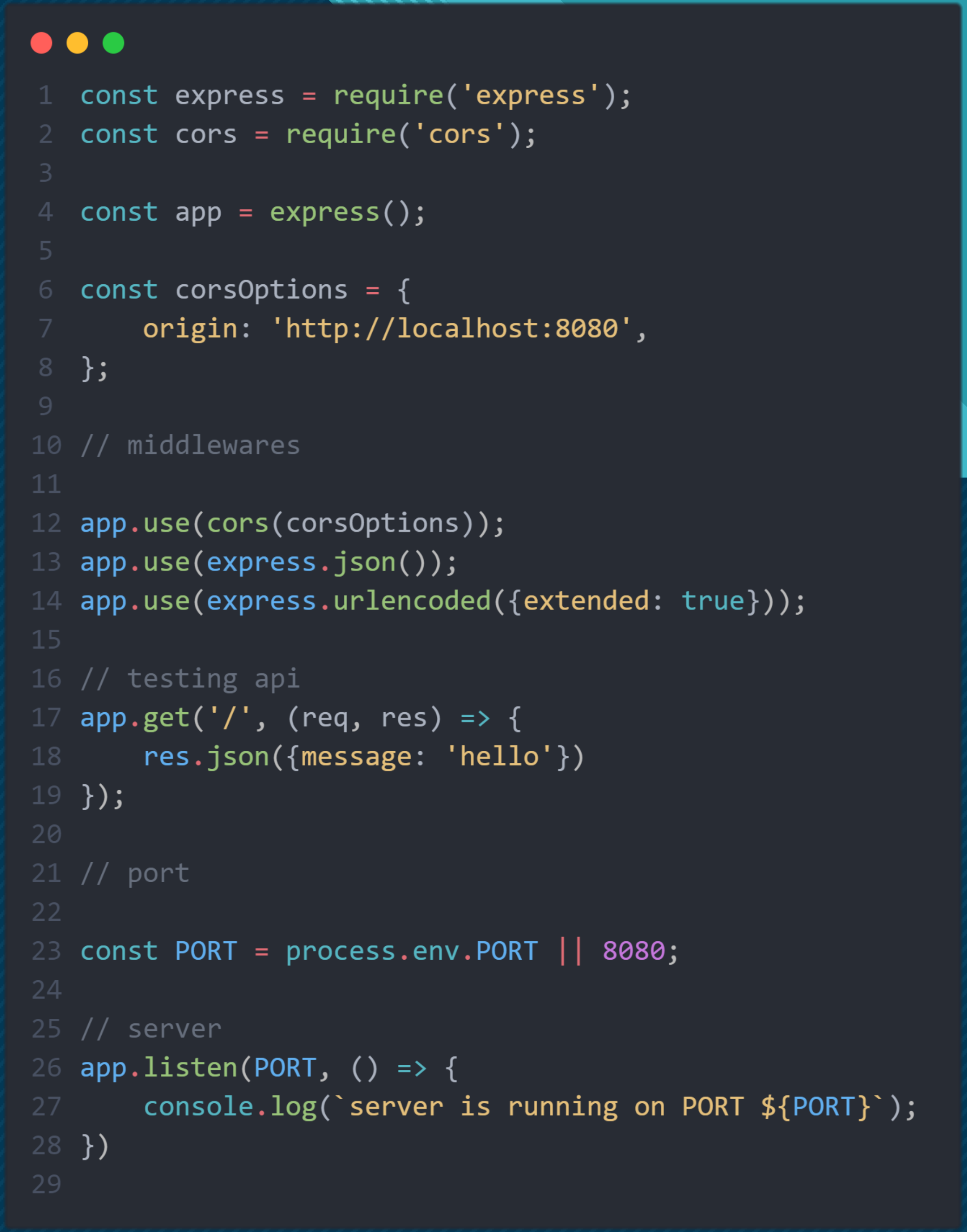
Get to Ready to start the Code…

n p m i e x p r e s s s e q u e l i z e c o r s m y s q l 2

CORS

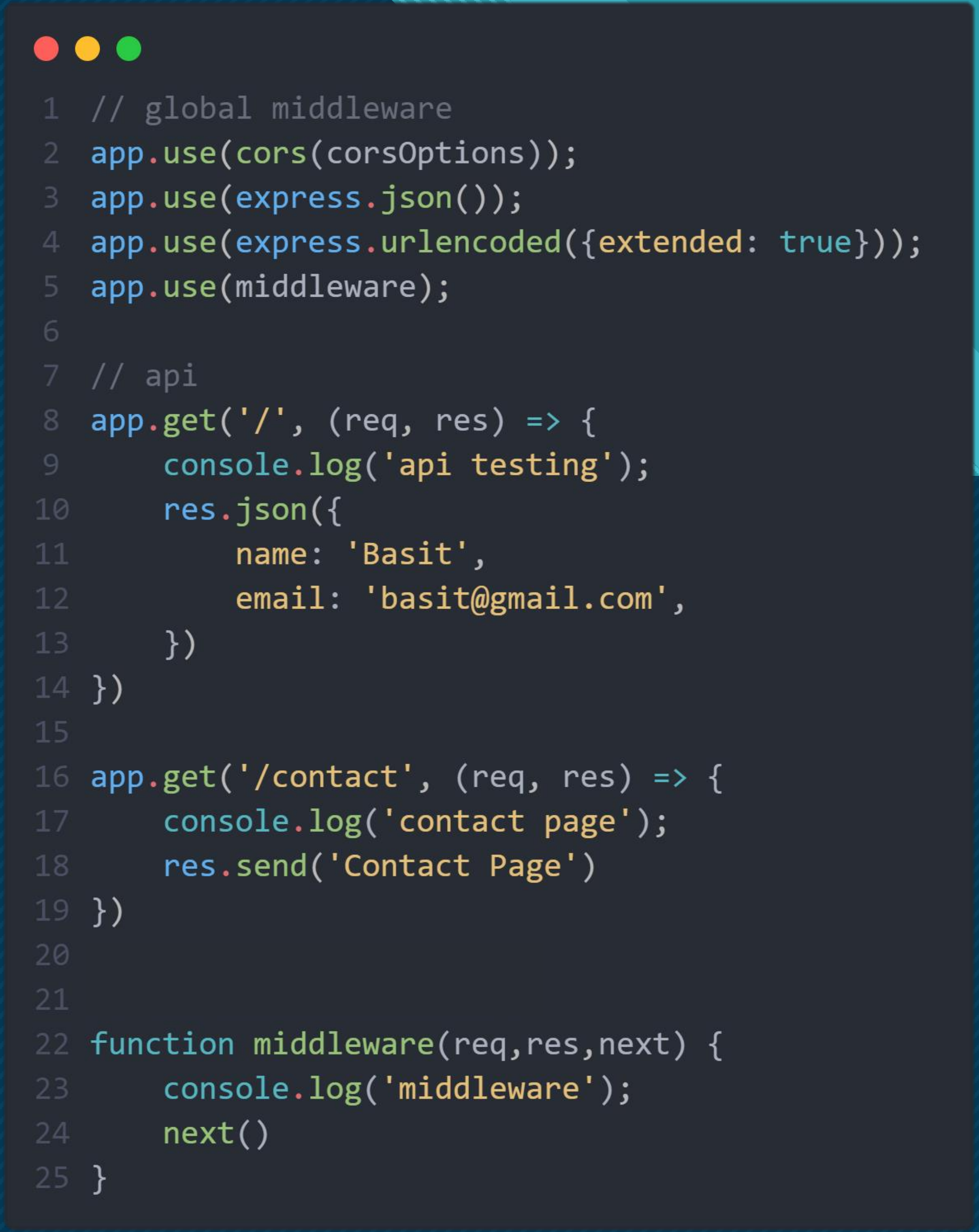
- Cross Origin Resource Sharing (CORS)

Create Server



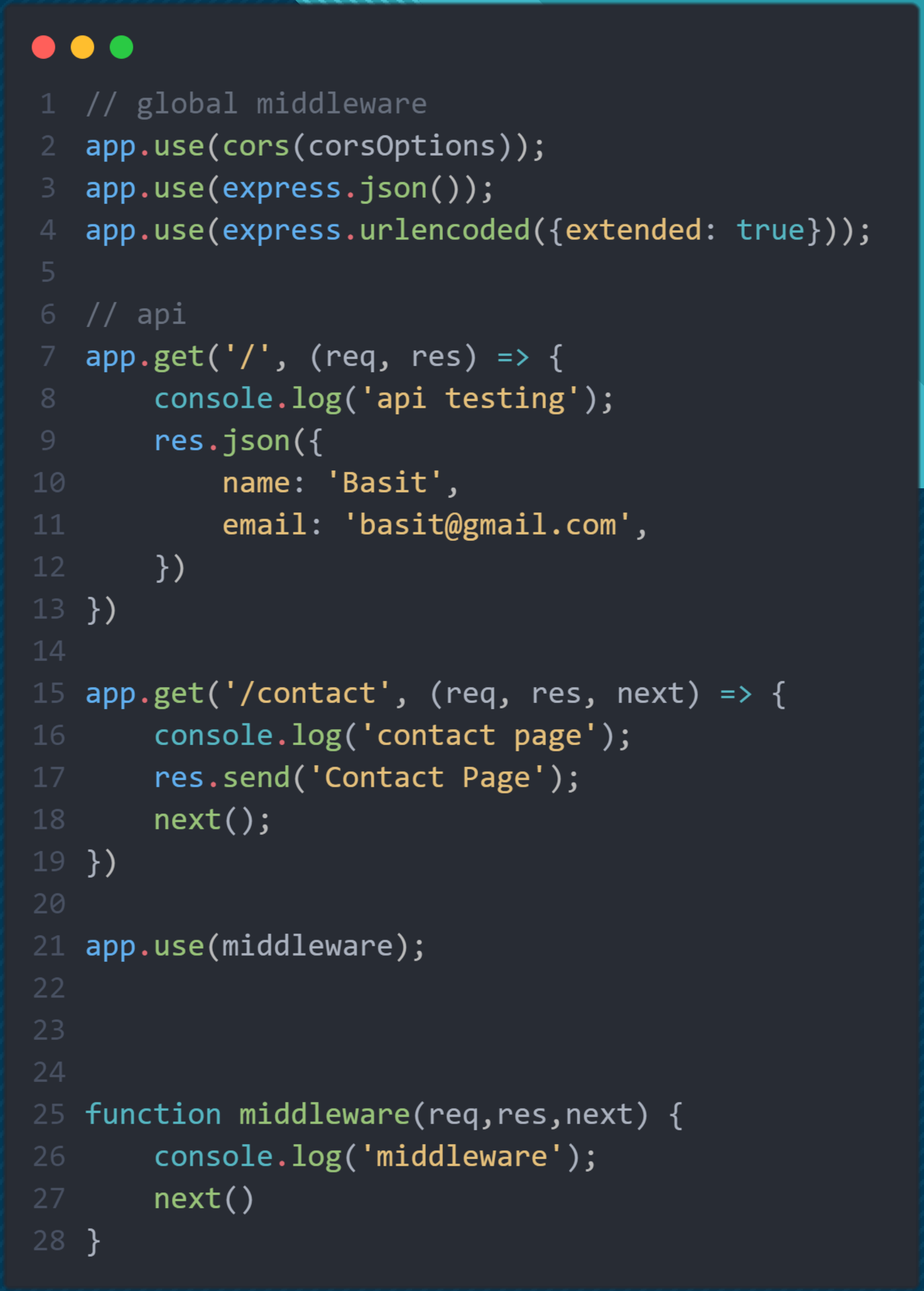
```
const express = require('express');
const cors = require('cors');

const app = express();

const corsOptions = {
    origin: 'http://localhost:8080',
};

// middlewares

app.use(cors(corsOptions));
app.use(express.json());
app.use(express.urlencoded({extended: true}));

// testing api
app.get('/', (req, res) => {
    res.json({message: 'hello'})
});

// port

const PORT = process.env.PORT || 8080;

// server
app.listen(PORT, () => {
    console.log(`server is running on PORT ${PORT}`);
})
```

Middleware



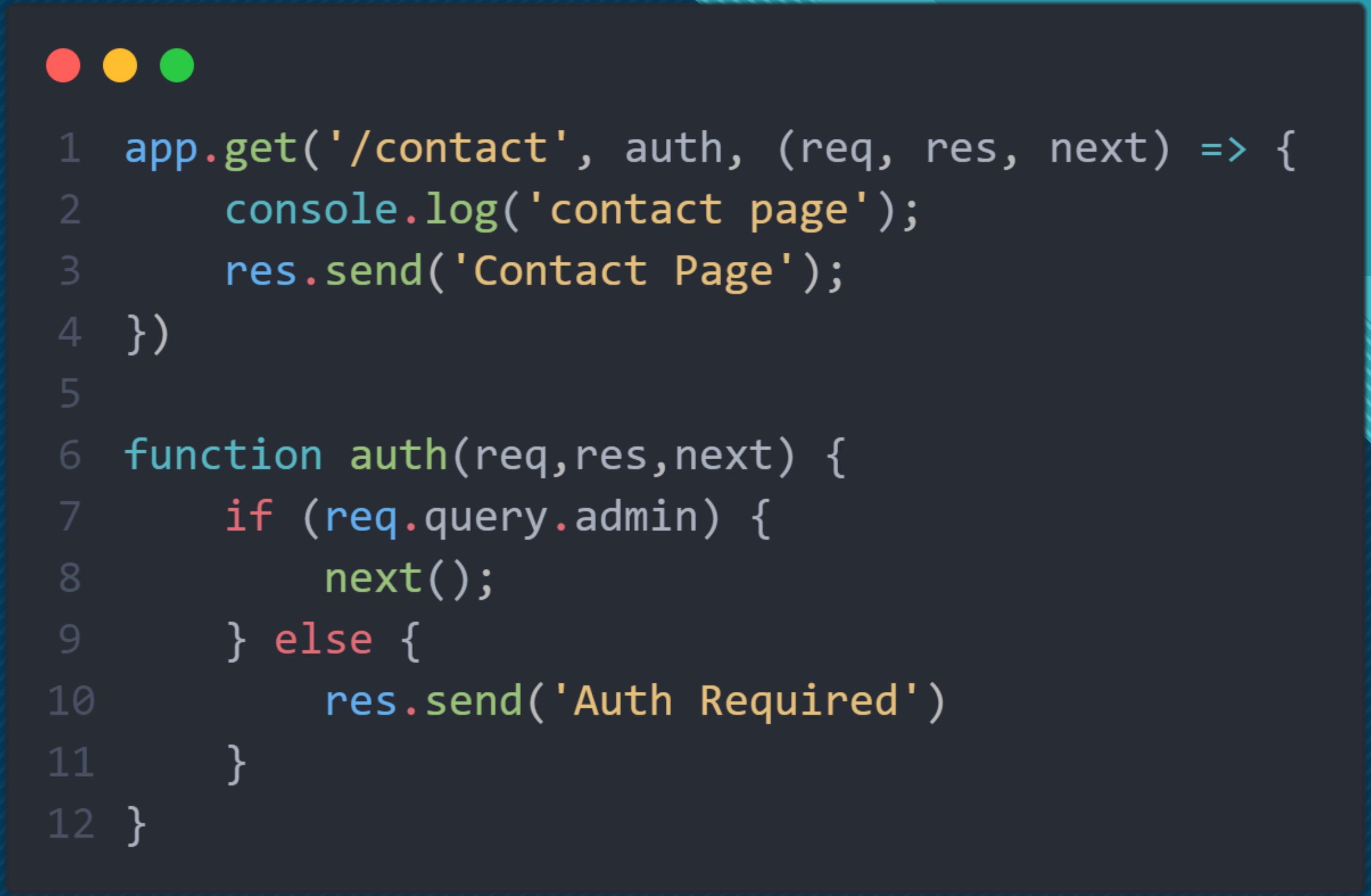
```javascript
// global middleware
app.use(cors(corsOptions));
app.use(express.json());
app.use(express.urlencoded({extended: true}));
app.use(middleware);

// api
app.get('/', (req, res) => {
    console.log('api testing');
    res.json({
        name: 'Basit',
        email: 'basit@gmail.com',
    })
})

app.get('/contact', (req, res) => {
    console.log('contact page');
    res.send('Contact Page')
})


function middleware(req,res,next) {
    console.log('middleware');
    next()
}
```

Middleware



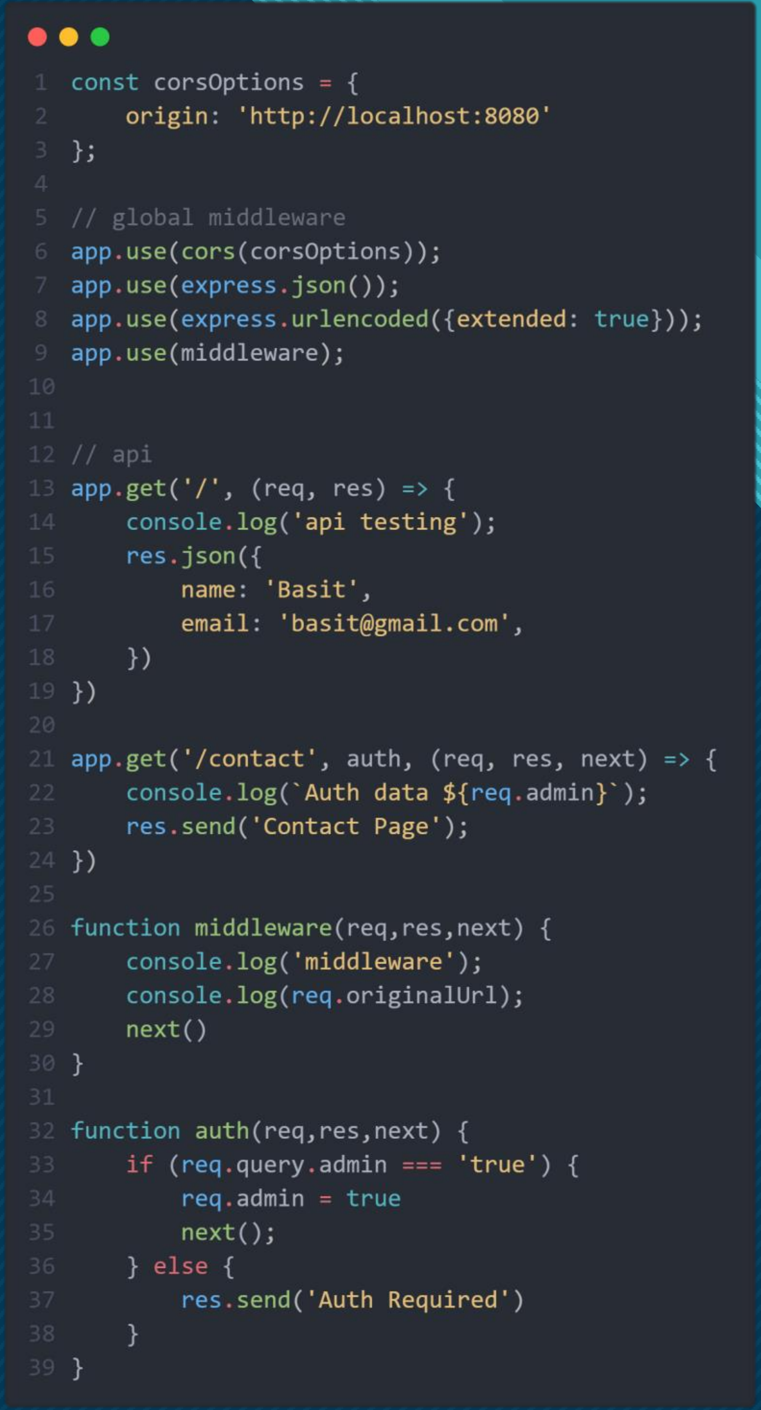
```javascript
// global middleware
app.use(cors(corsOptions));
app.use(express.json());
app.use(express.urlencoded({extended: true}));

// api
app.get('/', (req, res) => {
    console.log('api testing');
    res.json({
        name: 'Basit',
        email: 'basit@gmail.com',
    })
})

app.get('/contact', (req, res, next) => {
    console.log('contact page');
    res.send('Contact Page');
    next();
})

app.use(middleware);



function middleware(req,res,next) {
    console.log('middleware');
    next()
}
```

Middleware



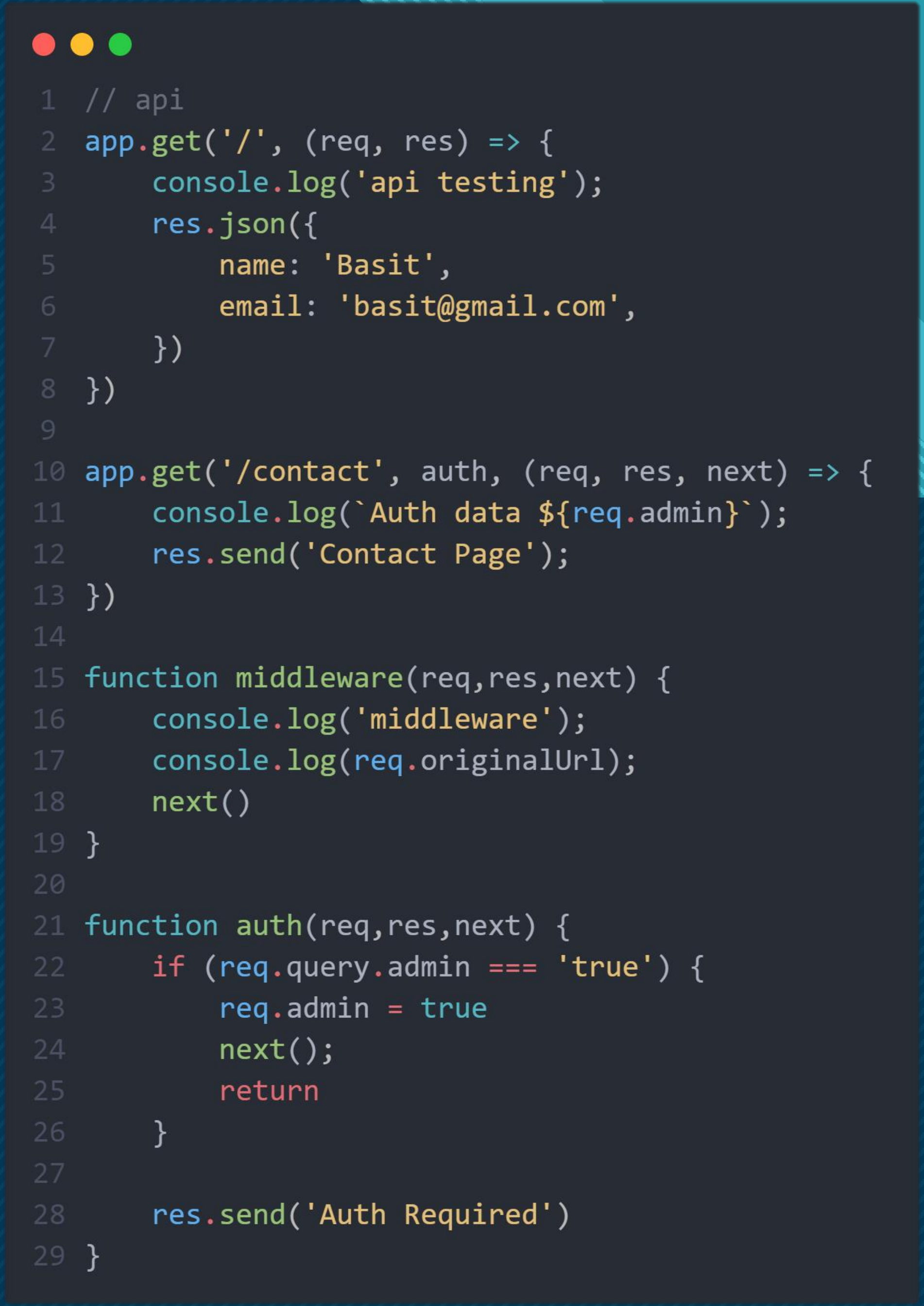
```
app.get('/contact', auth, (req, res, next) => {
    console.log('contact page');
    res.send('Contact Page');
})

function auth(req,res,next) {
    if (req.query.admin) {
        next();
    } else {
        res.send('Auth Required')
    }
}
```

Middleware Data Pass

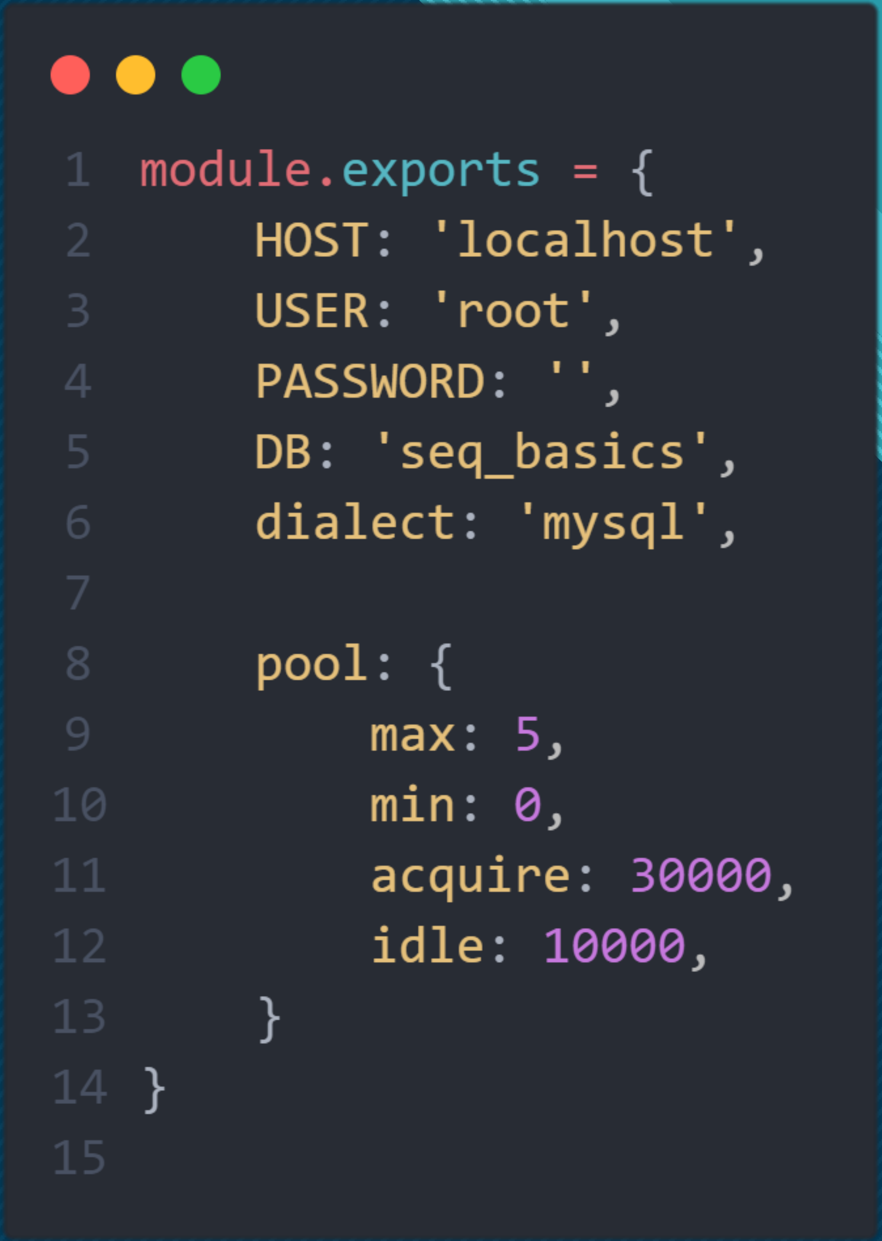


```
const corsOptions = {
    origin: 'http://localhost:8080'
};

// global middleware
app.use(cors(corsOptions));
app.use(express.json());
app.use(express.urlencoded({extended: true}));
app.use(middleware);


// api
app.get('/', (req, res) => {
    console.log('api testing');
    res.json({
        name: 'Basit',
        email: 'basit@gmail.com',
    })
})

app.get('/contact', auth, (req, res, next) => {
    console.log(`Auth data ${req.admin}`);
    res.send('Contact Page');
})

function middleware(req,res,next) {
    console.log('middleware');
    console.log(req.originalUrl);
    next()
}

function auth(req,res,next) {
    if (req.query.admin === 'true') {
        req.admin = true
        next();
    } else {
        res.send('Auth Required')
    }
}
```

Middleware Data Pass

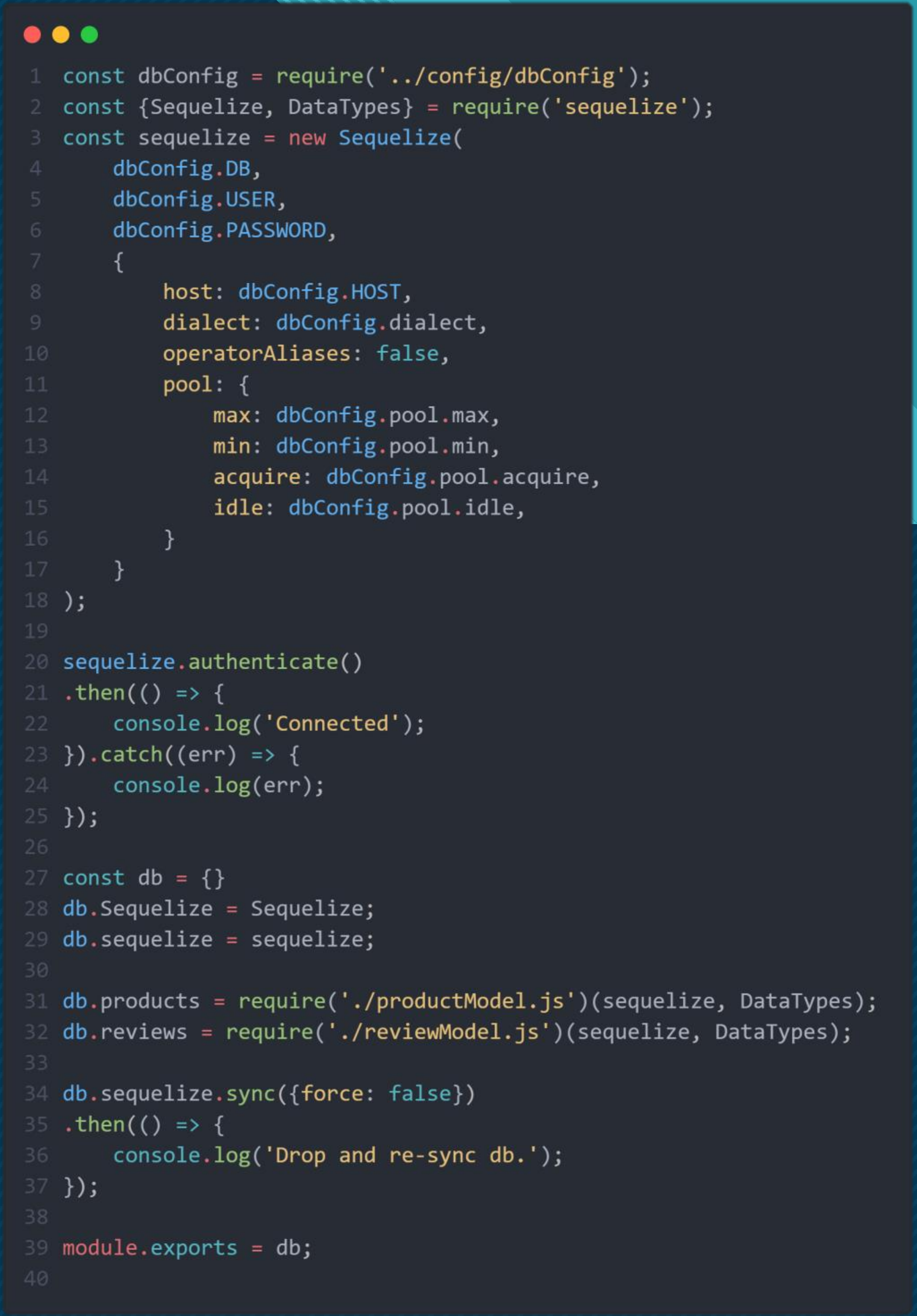

```
// api
app.get('/', (req, res) => {
    console.log('api testing');
    res.json({
        name: 'Basit',
        email: 'basit@gmail.com',
    })
})

app.get('/contact', auth, (req, res, next) => {
    console.log(`Auth data ${req.admin}`);
    res.send('Contact Page');
})

function middleware(req,res,next) {
    console.log('middleware');
    console.log(req.originalUrl);
    next()
}

function auth(req,res,next) {
    if (req.query.admin === 'true') {
        req.admin = true
        next();
        return
    }

    res.send('Auth Required')
}
```

Config.js



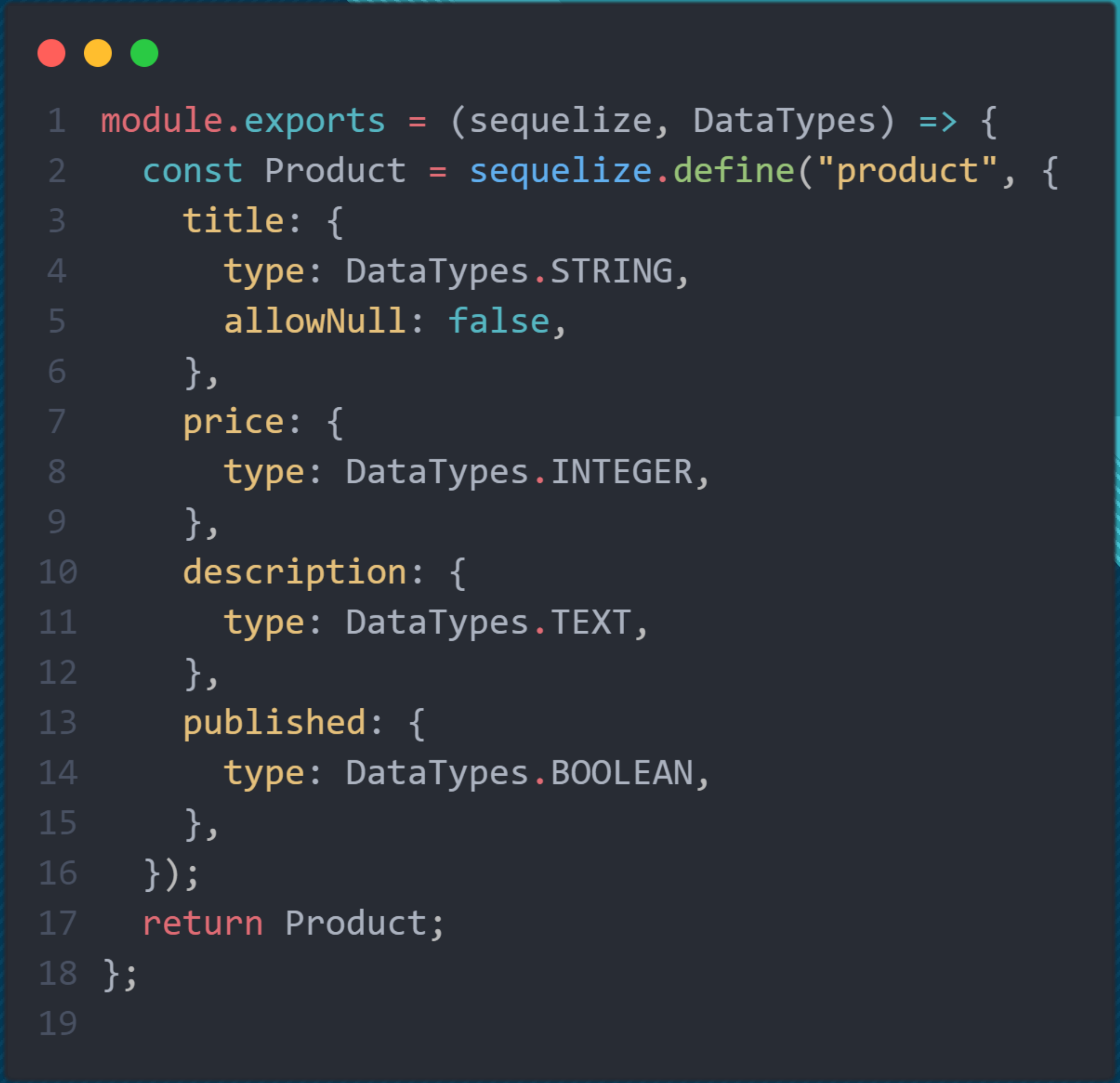
```
module.exports = {
    HOST: 'localhost',
    USER: 'root',
    PASSWORD: '',
    DB: 'seq_basics',
    dialect: 'mysql',

    pool: {
        max: 5,
        min: 0,
        acquire: 30000,
        idle: 10000,
    }
}
```

model/index.js



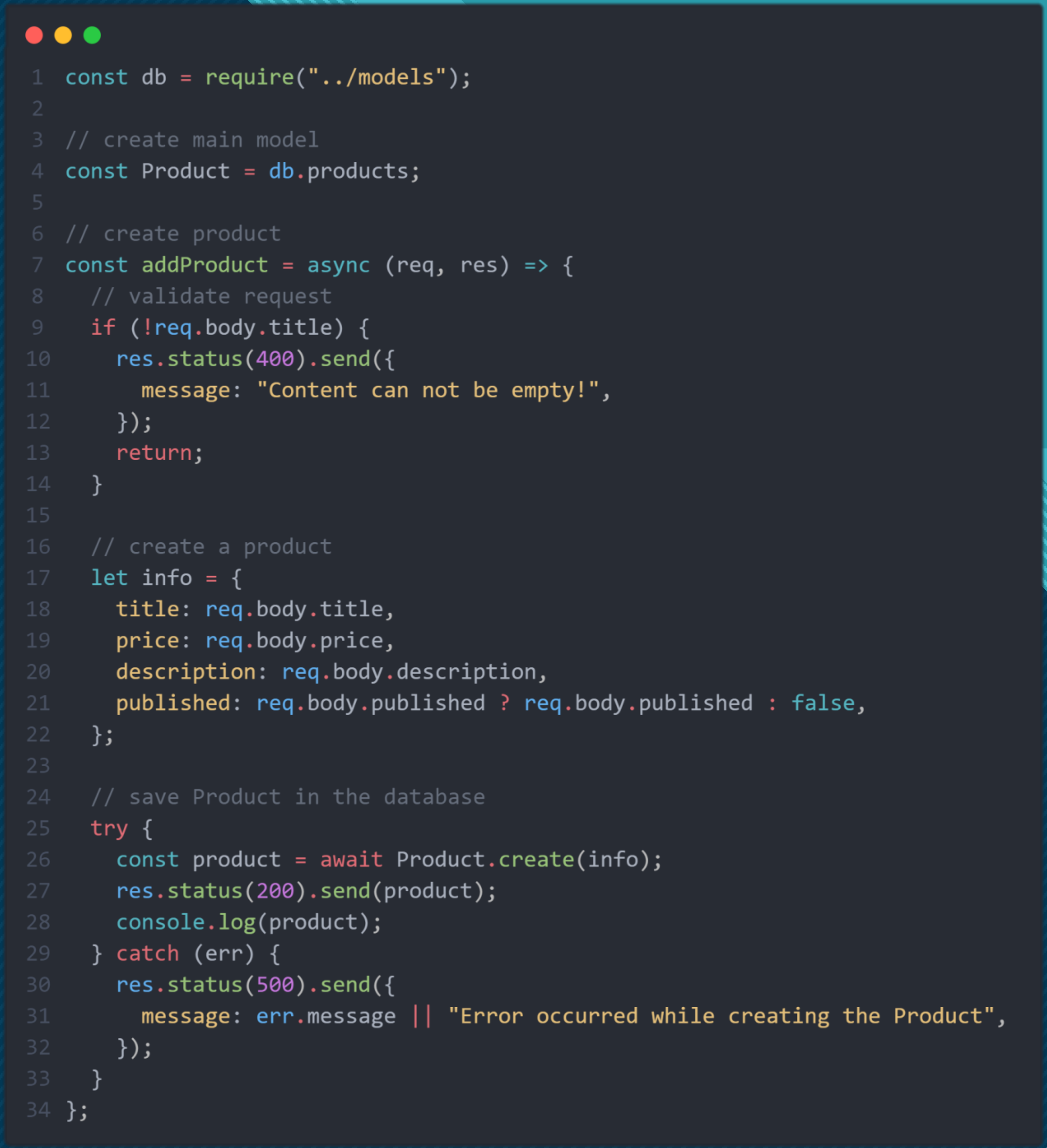
```javascript
const dbConfig = require('../config/dbConfig');
const {Sequelize, DataTypes} = require('sequelize');
const sequelize = new Sequelize(
    dbConfig.DB,
    dbConfig.USER,
    dbConfig.PASSWORD,
    {
        host: dbConfig.HOST,
        dialect: dbConfig.dialect,
        operatorAliases: false,
        pool: {
            max: dbConfig.pool.max,
            min: dbConfig.pool.min,
            acquire: dbConfig.pool.acquire,
            idle: dbConfig.pool.idle,
        }
    }
);

sequelize.authenticate()
.then(() => {
    console.log('Connected');
}).catch((err) => {
    console.log(err);
});

const db = {}
db.Sequelize = Sequelize;
db.sequelize = sequelize;

db.products = require('./productModel.js')(sequelize, DataTypes);
db.reviews = require('./reviewModel.js')(sequelize, DataTypes);

db.sequelize.sync({force: false})
.then(() => {
    console.log('Drop and re-sync db.');
});

module.exports = db;
```

model/ productModel.js



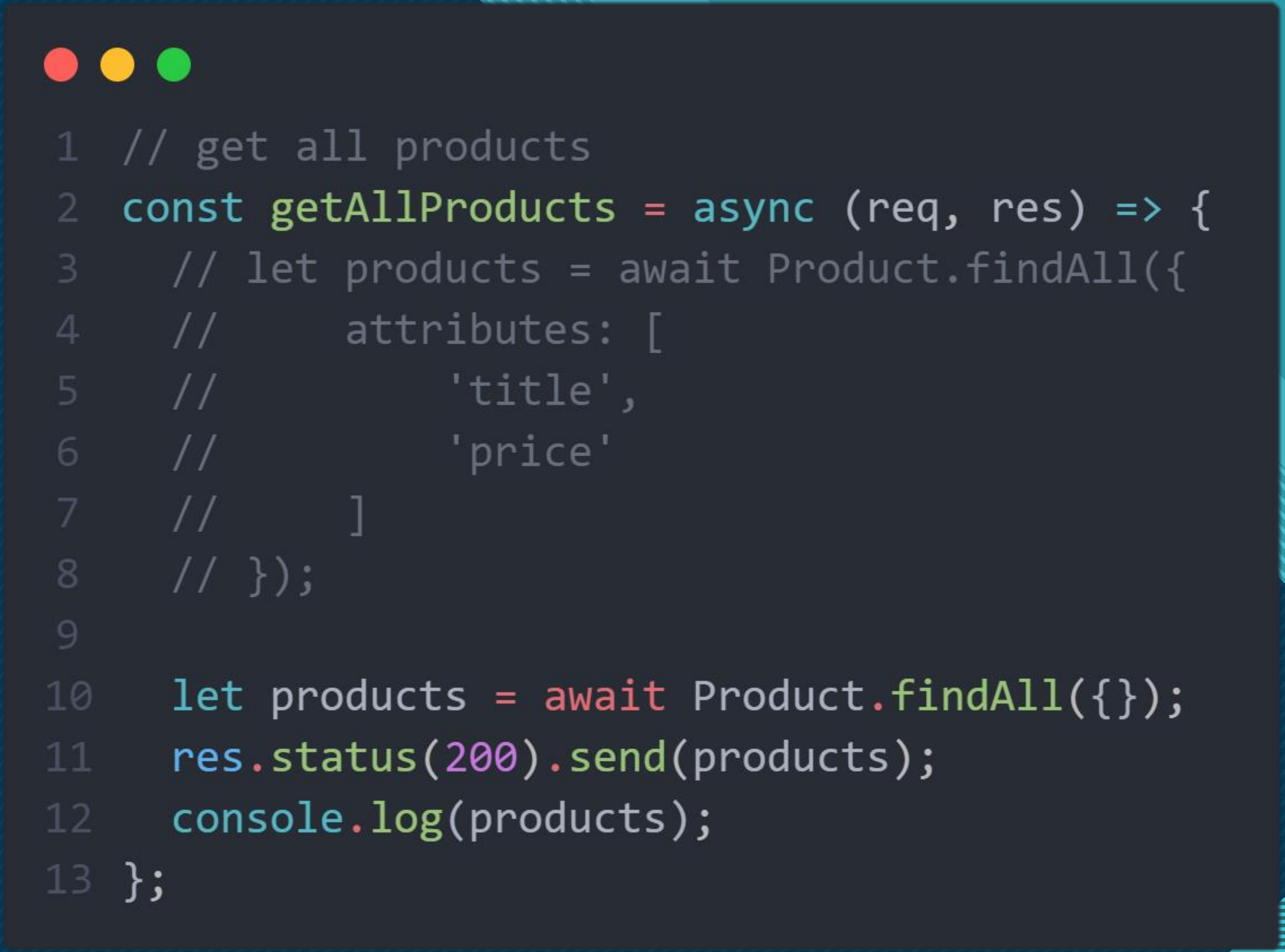
```javascript
module.exports = (sequelize, DataTypes) => {
  const Product = sequelize.define("product", {
    title: {
      type: DataTypes.STRING,
      allowNull: false,
    },
    price: {
      type: DataTypes.INTEGER,
    },
    description: {
      type: DataTypes.TEXT,
    },
    published: {
      type: DataTypes.BOOLEAN,
    },
  });
  return Product;
};

```

Controller/ProductController.js



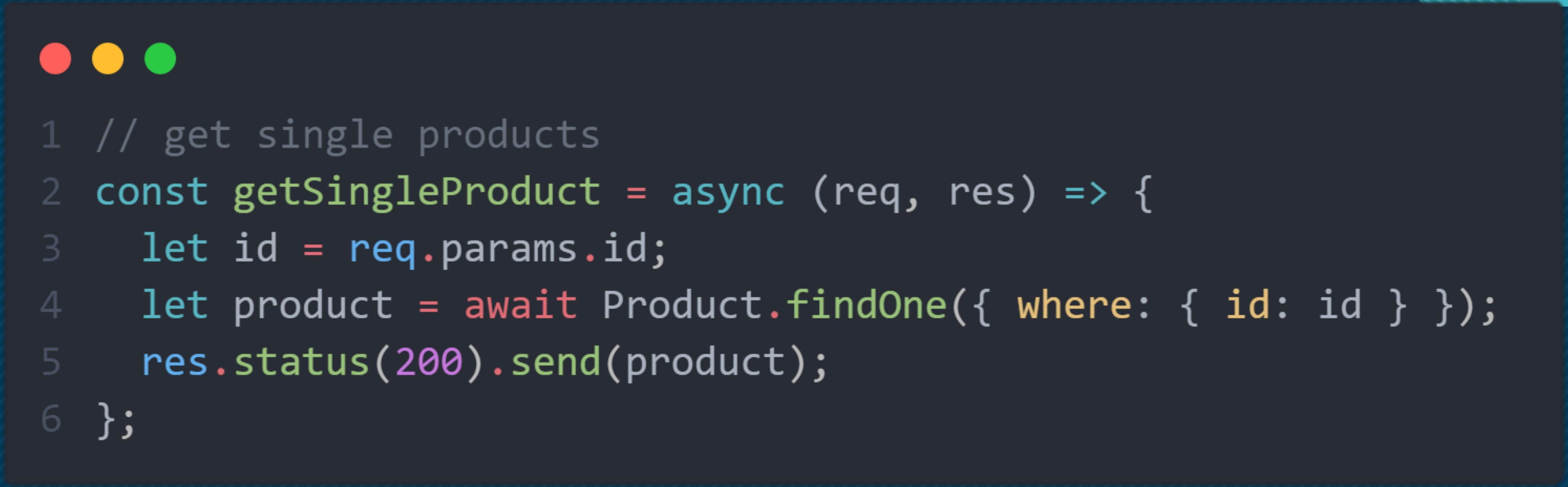
```javascript
const db = require("../models");

// create main model
const Product = db.products;

// create product
const addProduct = async (req, res) => {
  // validate request
  if (!req.body.title) {
    res.status(400).send({
      message: "Content can not be empty!",
    });
    return;
  }

  // create a product
  let info = {
    title: req.body.title,
    price: req.body.price,
    description: req.body.description,
    published: req.body.published ? req.body.published : false,
  };

  // save Product in the database
  try {
    const product = await Product.create(info);
    res.status(200).send(product);
    console.log(product);
  } catch (err) {
    res.status(500).send({
      message: err.message || "Error occurred while creating the Product",
    });
  }
};
```

Controller/ ProductController.js



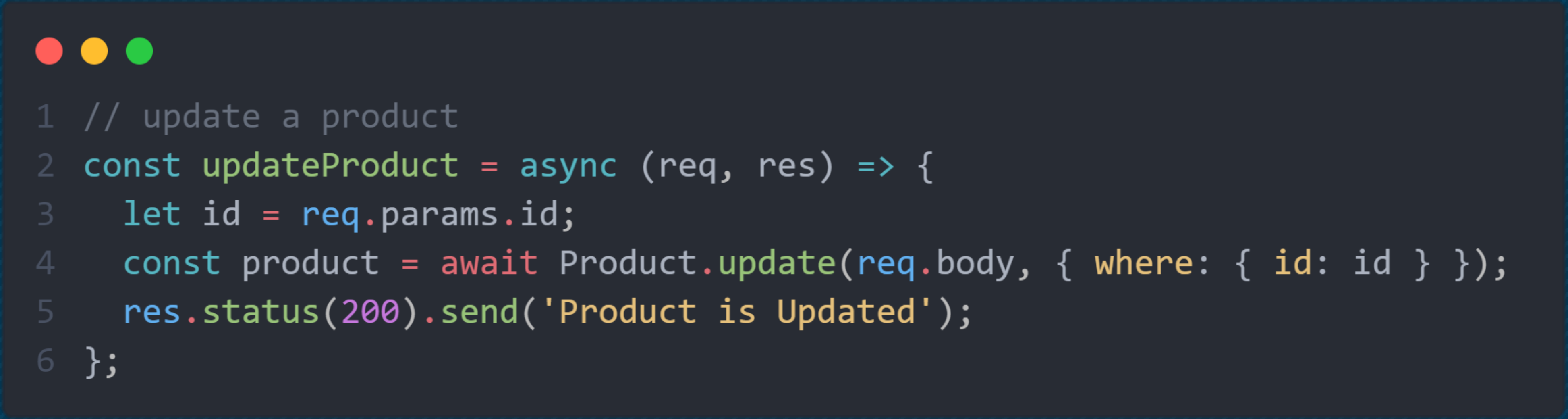
```javascript
// get all products
const getAllProducts = async (req, res) => {
  // let products = await Product.findAll({
  //     attributes: [
  //         'title',
  //         'price'
  //     ]
  // });

  let products = await Product.findAll({});
  res.status(200).send(products);
  console.log(products);
};
```

Controller/ ProductController.js



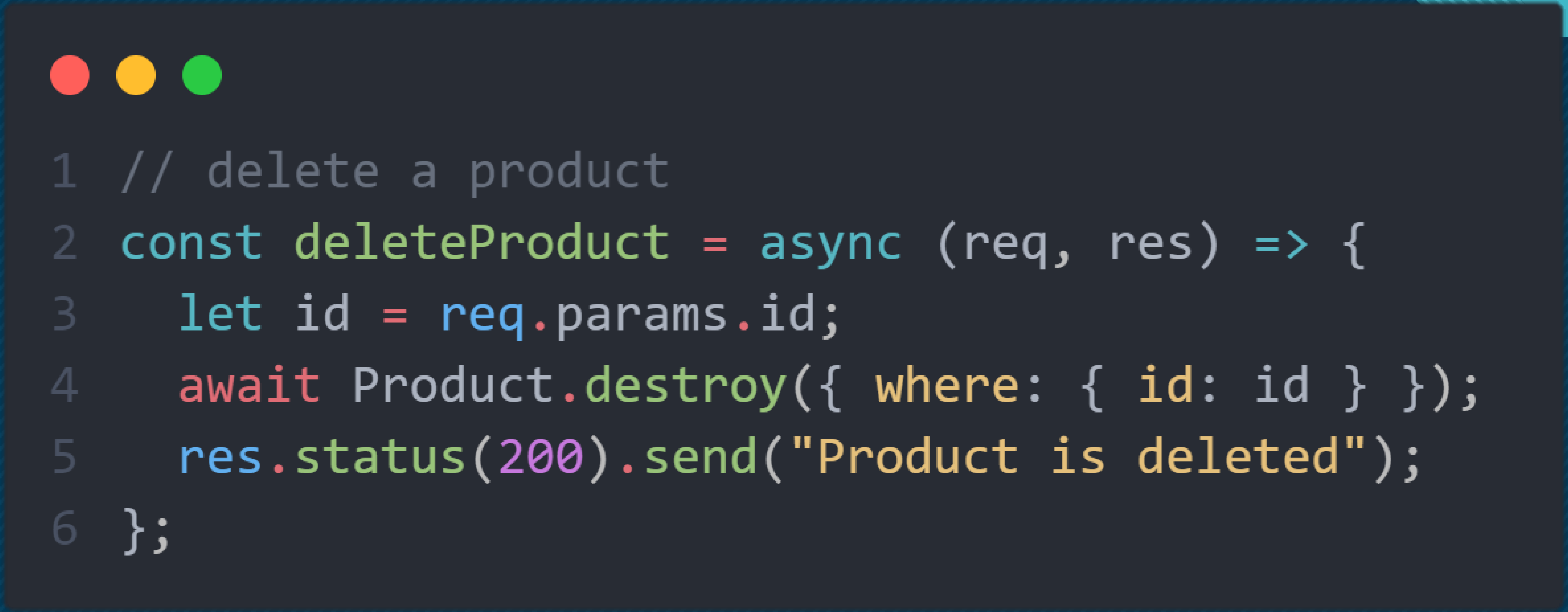
```
// get single products
const getSingleProduct = async (req, res) => {
  let id = req.params.id;
  let product = await Product.findOne({ where: { id: id } });
  res.status(200).send(product);
};
```

Controller/ ProductController.js



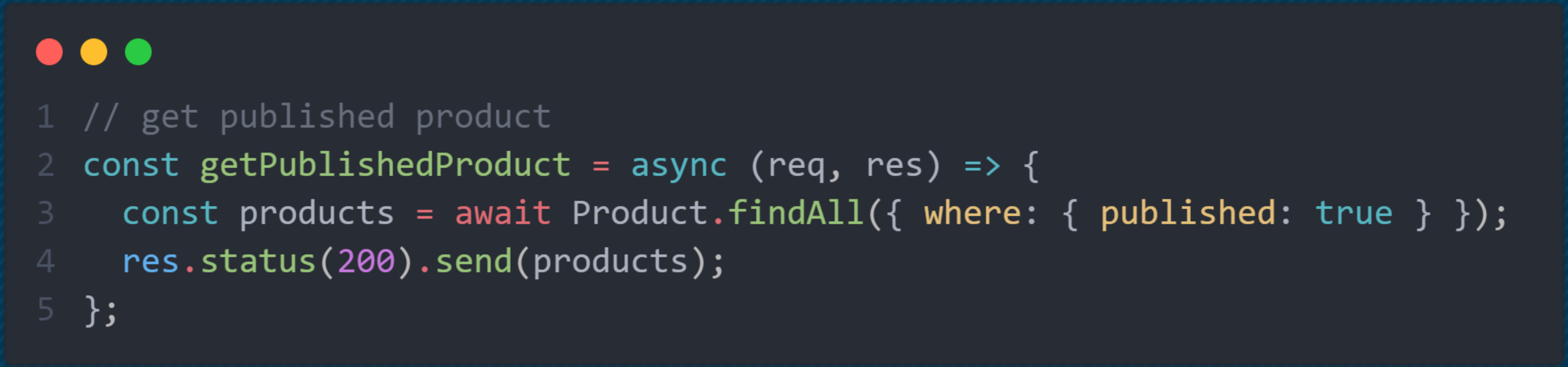
```
// update a product
const updateProduct = async (req, res) => {
  let id = req.params.id;
  const product = await Product.update(req.body, { where: { id: id } });
  res.status(200).send('Product is Updated');
};
```

Controller/ ProductController.js

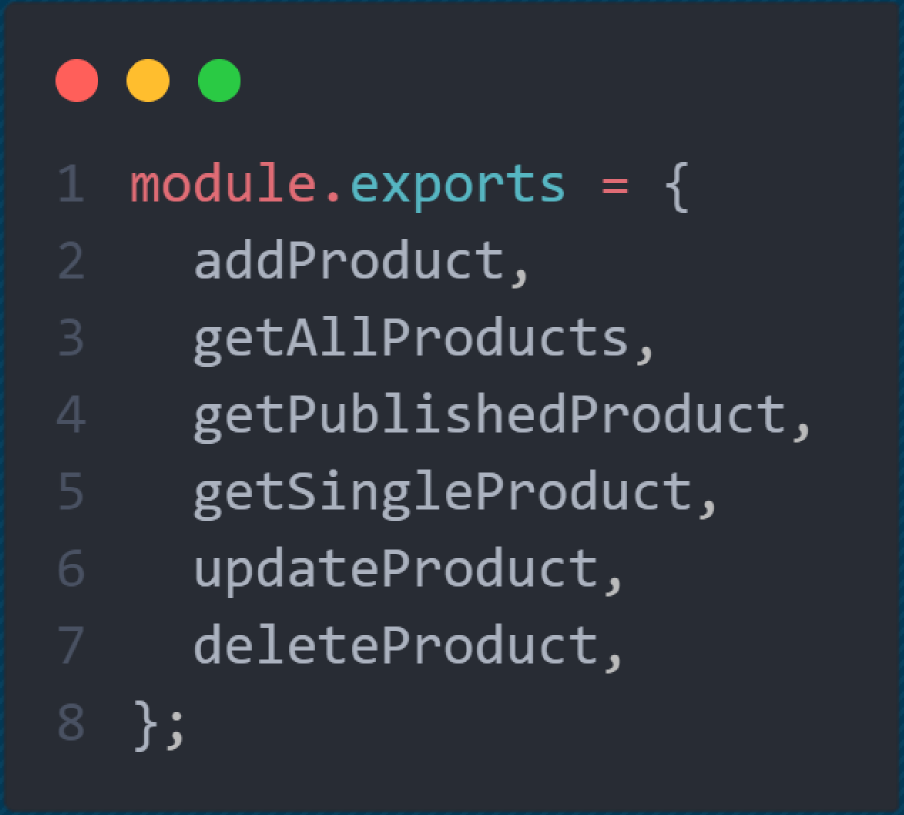


```javascript
// delete a product
const deleteProduct = async (req, res) => {
  let id = req.params.id;
  await Product.destroy({ where: { id: id } });
  res.status(200).send("Product is deleted");
};
```

Controller/ ProductController.js

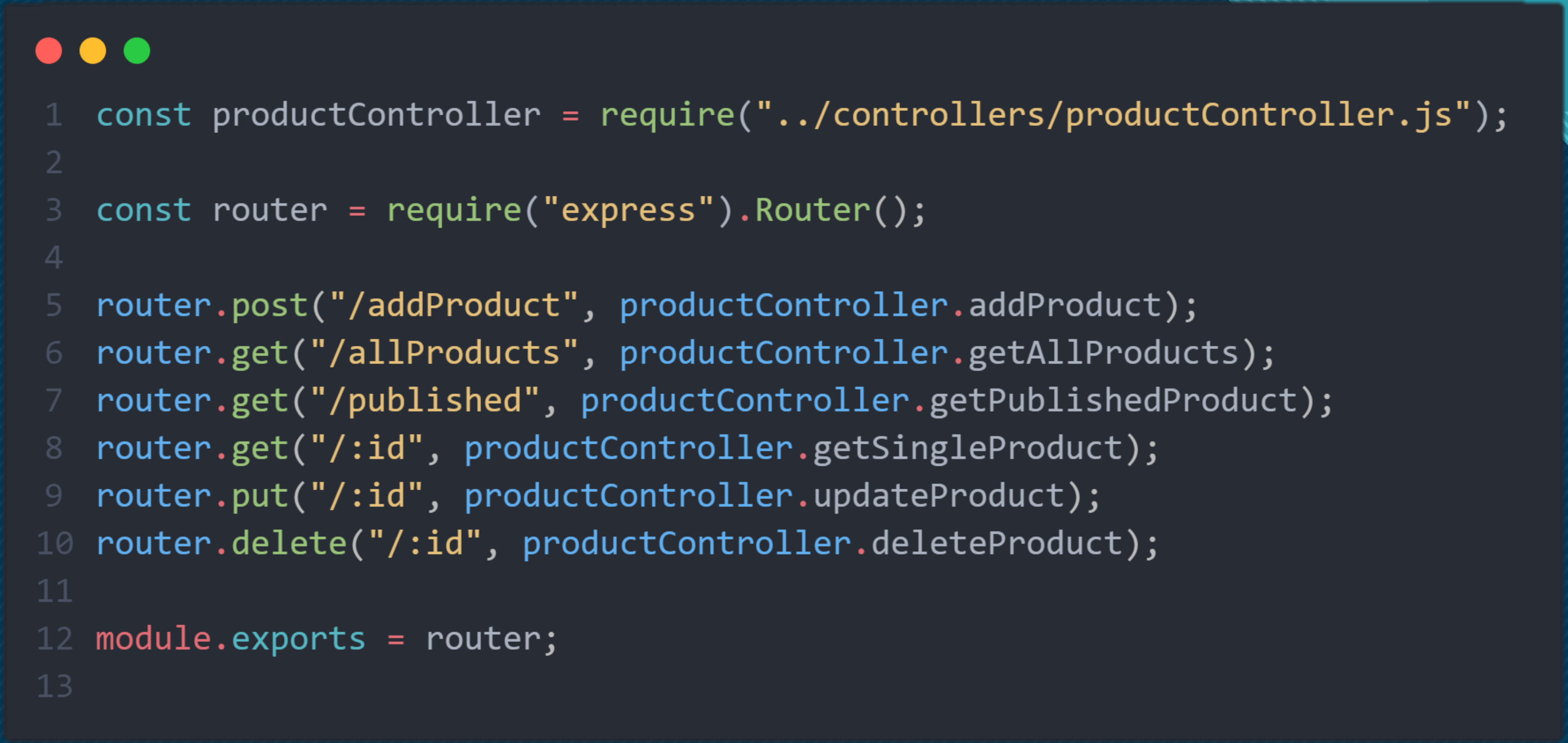

```javascript
// get published product
const getPublishedProduct = async (req, res) => {
  const products = await Product.findAll({ where: { published: true } });
  res.status(200).send(products);
};
```

Controller/ ProductController.js



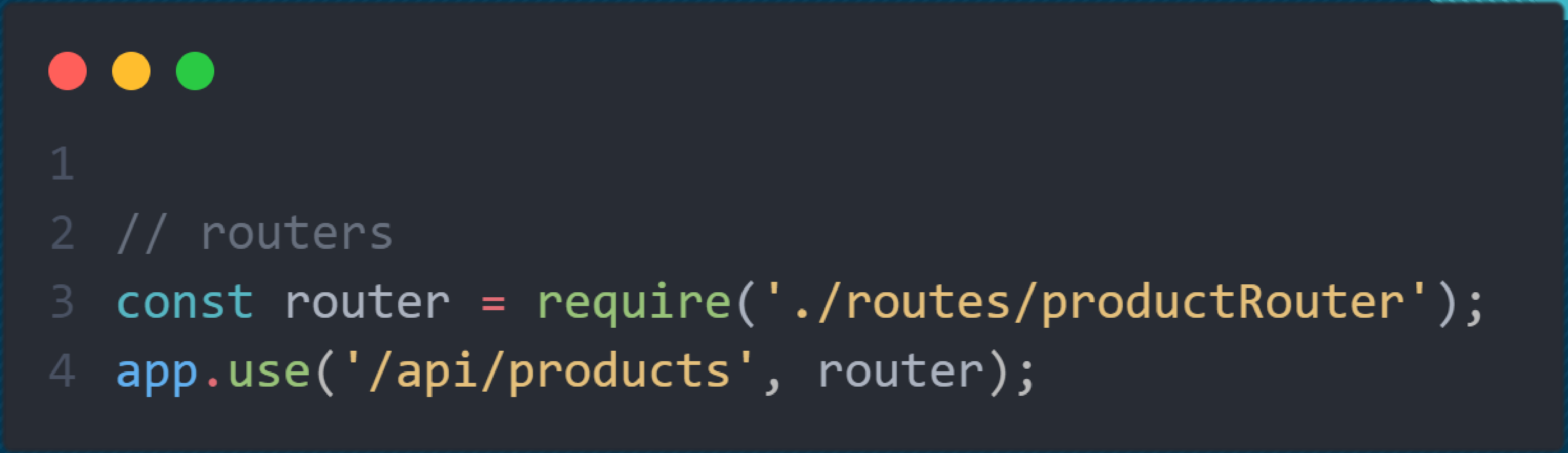
```
module.exports = {
  addProduct,
  getAllProducts,
  getPublishedProduct,
  getSingleProduct,
  updateProduct,
  deleteProduct,
};
```

routes/productRoutes.js



```
const productController = require("../controllers/productController.js");

const router = require("express").Router();

router.post("/addProduct", productController.addProduct);
router.get("/allProducts", productController.getAllProducts);
router.get("/published", productController.getPublishedProduct);
router.get("/:id", productController.getSingleProduct);
router.put("/:id", productController.updateProduct);
router.delete("/:id", productController.deleteProduct);

module.exports = router;

```

Server.js



```

// routers
const router = require('./routes/productRouter');
app.use('/api/products', router);
```

Content Title

Caption01 appears here

Caption02 appears here

Caption03 appears here

Caption04 appears here

Caption05 appears here

Content Title 03

Caption01 appears here

Caption03 appears here

Caption04 appears here

Content Title 04

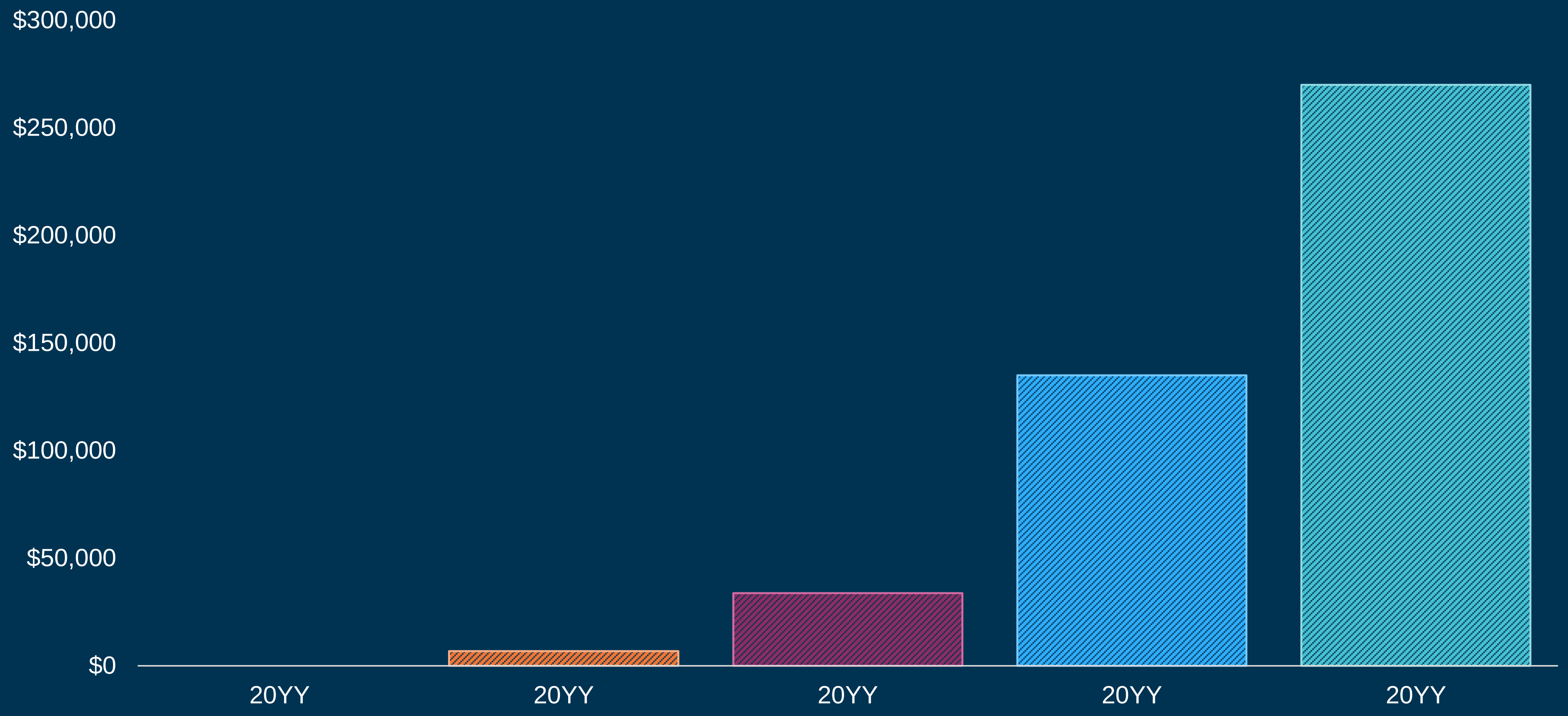
Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Maecenas porttitor congue massa. Fusce posuere, magna sed pulvinar ultricies, purus lectus malesuada libero, sit amet commodo magna eros quis urna. Nunc viverra imperdiet enim. Fusce est. Vivamus a tellus.

Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Proin pharetra nonummy pede. Mauris et orci.

Table

Title	Title	Title	Title

Chart



$300,000
$250,000
$200,000
$150,000
$100,000
$50,000
$0
20YY
20YY
20YY
20YY
20YY

Quote appears here
Lorem ipsum dolor sit amet, consectetuer adipiscing elit."
- Author

Thank You 1

Thank You 2

Customize this Template

Template Editing Instructions and Feedback