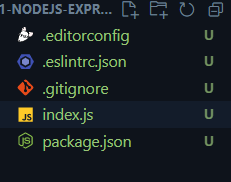
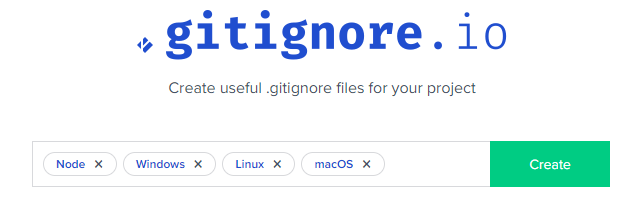
# 2. Creando estructura de proyecto



Página donde encontrarás configuraciones de gitignore:

<https://www.toptal.com/developers/gitignore/>



## . EDITORCONFIG

Sirve para que todos tus desarrolladores tengan un mismo entorno de desarrollo.

Instalar la extensión: EditorConfig for VS Code



Copiar en el archivo:

*# Editor configuration, see https://editorconfig.org*

root = true

[\*]

charset = utf-8

indent\_style = space

indent\_size = 2

insert\_final\_newline = true

trim\_trailing\_whitespace = true

[\*.js]

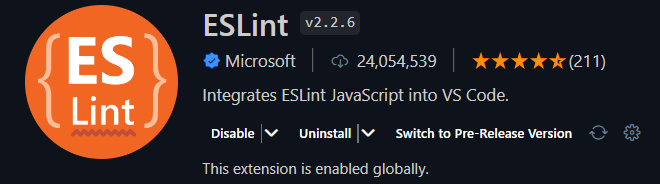
quote\_type = single

[\*.md]

max\_line\_length = off

trim\_trailing\_whitespace = false

## . ESLINTRC.JSON



{

  "parserOptions": {

    "ecmaVersion": 2018

  },

  "extends": [

    "eslint:recommended",

    "prettier"

  ],

  "env": {

    "es6": true,

    "node": true,

    "jest": true

  },

  "rules": {

    "no-console": "warn"

  }

}

## Agregar script en el package.json:

"scripts": {

    "dev":"nodemon index.js",

    "start":"node index.js",

    "lint":"eslint"

  },

## Instalar dependencias:

npm i nodemon eslint eslint-config-prettier eslint-plugin-prettier prettier -D

# 3. Instalación de Express.js y tu primer servidor HTTP

const express = require('express');

const app = express();

const port = 3000;

app.listen(port,()=>{

  console.log('Corriendo en el puerto :',port);

});

Comando: npm i express

# 4. Rutas en Express

app.get('/',(req,res)=>{

  res.send('Hola mi server en express')

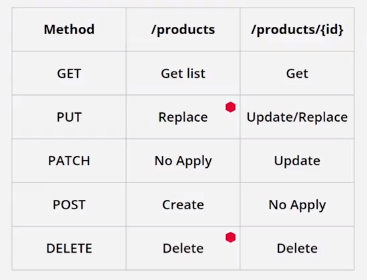
});

app.get('/otra-ruta',(req,res)=>{

  res.send('Hola soy otra ruta')

});

# 5. ¿Qué es una RESTful API?



# 6. GET: recibir parámetros

app.get('/categories/:categoryId/products/:productId',(req,res)=>{

  const {categoryId,productId} = req.params;

  res.json(

    {

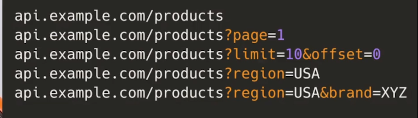
      categoryId,

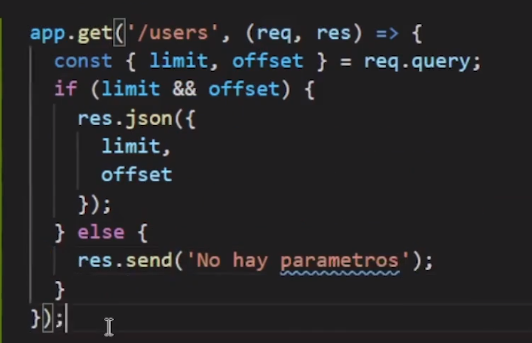
      productId

    })

});

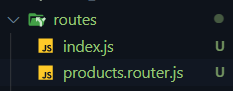
# 7. GET: parámetros query





**Adicional**, para que rutas estáticas y dinámicas no choquen, deben ir antes las estáticas.

# 8. Separación de responsabilidades con express.Router



const express = require('express');

const faker = require('faker');

const router = express.Router();

router.get('/',(req,res)=>{

  const products = [];

  for (let index = 0; index < 100; index++) {

    products.push({

      name:faker.commerce.productName(),

      price:parseInt(faker.commerce.price(),10),

      image:faker.image.imageUrl(),

    })

  }

  res.json(products)

});

module.exports = router;

const producstRouter = require('./products.router');

*// const userRouter = require('./user.router');*

function routerApi(app){

  app.use('/products',producstRouter);

*// app.use('/users',producstRouter);*

}

module.exports = routerApi;

const express = require('express');

const routerApi = require('./routes');

const app = express();

const port = 3000;

app.get('/',(req,res)=>{

  res.send('Hola mi server en express')

});

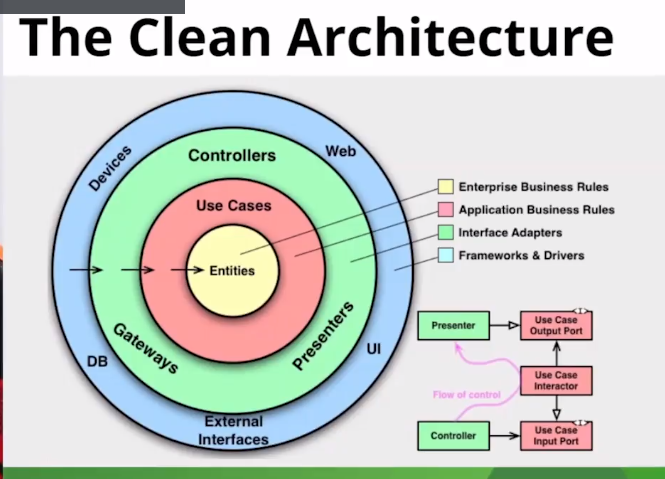
app.get('/otra-ruta',(req,res)=>{

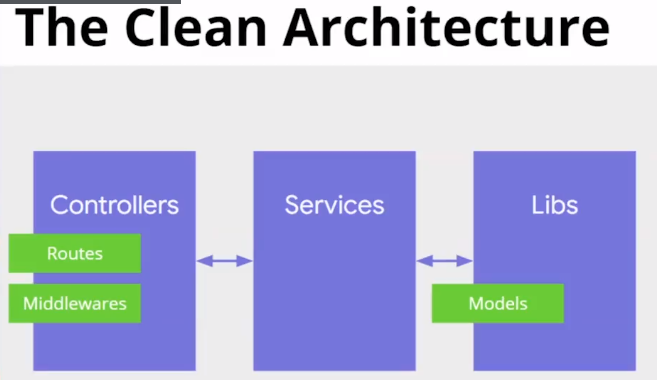
  res.send('Hola soy otra ruta')

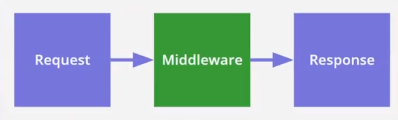
});

routerApi(app);

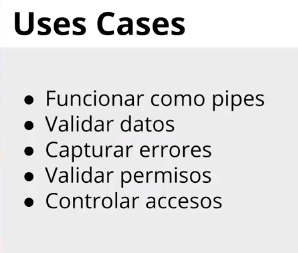
# 13. Introducción a servicios: crea tu primer servicio







# 16. ¿Qué son los Middlewares?



# 17. Middleware para HttpErrors

En index.js

const {logErrors, errorHandler} = require('./middlewares/error.handler')

app.use(logErrors);

app.use(errorHandler);

En carpeta creada por nosotros: middlewares/error.hadler.js

function logErrors (err,req,res,next){

  console.log('logErros')

  console.log(err);

  next(err);

}

function errorHandler(err,req,res,next){

  console.log('errorHandler')

  res.status(500).json({

    message:err.message,

    stack:err.stack

  });

}

module.exports = {

  logErrors,

  errorHandler

}

# 18. Manejo de errores con Boom

Instalar: npm i @hapi/boom

Service:

const boom = require('@hapi/boom');

async findOne(id){

    const product =  *this*.products.find((item) => item.id === id);

    if(!product){

      throw boom.notFound('product not found');

    }

    if(product.isBlock){

      throw boom.conflict('product is block');

    }

    return product;

  }

Middleware:

function boomErrorHandler(err,req,res,next){

  if(err.isBoom){

    const {output} = err;

    res.status(output.statusCode).json(output.payload);

  }

  next(err)

}

# 19. Validación de datos con Joi

npm i joi

Se crea la carpeta “schemas” o tbn conocida como “dto”

Schema o DTO (Data Transfer objects), (objetos de Transferencia de datos)

Validar los datos que nos envían desde el cliente

Middelware para validar

const boom = require('@hapi/boom');

function validatorHandler(schema,property){

  return (req,res,next) =>{

    const data = req[property];

    const {error} = schema.validate(data);

    if(error){

      next(boom.badRequest(error));

    }

    next();

  }

}

module.exports = validatorHandler;

Tambien debemos crear una carpeta “schema”

const Joi = require('joi');

const id = Joi.string().uuid();

const name = Joi.string().alphanum().min(3).max(15);

const price = Joi.number().integer().min(10);

const createProductSchema = Joi.object({

  name: name.required(),

  price: price.required(),

});

# 20 . Probando nuestros endpoints

const validatorHandler   = require('./../middlewares/validator.handler')

const  {createProductSchema,updateProductSchema,getProductSchema} = require('./../schemas/product.schema')

router.patch('/:id',

validatorHandler(getProductSchema,'params'),

  validatorHandler(updateProductSchema,'body'),

  async (req,res,next)=>{

  try {

    const {id} = req.params;

    const body = req.body;

    const product = await service.update(id,body);

    res.json(product);

  } catch (error) {

    next(error);

  }

});

# 21. Middlewares populares en Express.js

CORS

Middleware para habilitar CORS (Cross-origin resource sharing) en nuestras rutas o aplicación. <http://expressjs.com/en/resources/middleware/cors.html>

Morgan

Un logger de solicitudes HTTP para Node.js. <http://expressjs.com/en/resources/middleware/morgan.html>

Helmet

Helmet nos ayuda a proteger nuestras aplicaciones Express configurando varios encabezados HTTP. ¡No es a prueba de balas de plata, pero puede ayudar! <https://github.com/helmetjs/helmet>

Express Debug

Nos permite hacer debugging de nuestras aplicaciones en Express mediante el uso de un toolbar en la pagina cuando las estamos desarrollando. <https://github.com/devoidfury/express-debug>

Express Slash

Este middleware nos permite evitar preocuparnos por escribir las rutas con o sin slash al final de ellas. <https://github.com/ericf/express-slash>

Passport

Passport es un middleware que nos permite establecer diferentes estrategias de autenticación a nuestras aplicaciones. <https://github.com/jaredhanson/passport>

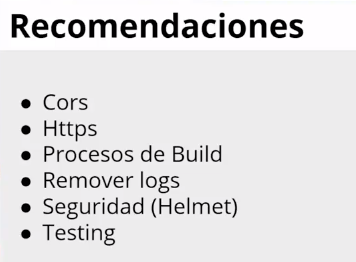
Puedes encontrar más middlewares populares en el siguiente enlace: <http://expressjs.com/en/resources/middleware.html>

Otros middlewares interesantes:

* express-session
* cookie-parser
* method-override
* multer
* express-validator

# 22. Consideraciones para producción

<https://developer.mozilla.org/en-US/docs/Web/HTTP/CORS>



# 23. Problema de CORS

Instalamos: npm i cors

Trabajos en index.js

const cors  = require('cors');

const whiteList = ['http://localhost:8080','https://myapp.com'];

const options = {

  origin:(origin, callback)=> {

    if(whiteList.includes(origin)){

      callback(null,true);

    } else{

      callback(new Error('NO PERMITIDO'));

    }

  }

}

app.use(cors(options));