

nodejs

logicstudio
INNOVATION | SERVICES | SOLUTIONS

Diego Zúñiga



Octubre 2019

Diego Zúñiga

Agenda

- Introducción
 - npm/node
 - scripts
 - modules
 - event loop
- fs
 - streams
- http
 - http server

Agenda(2)

- ExpressJS
 - routes
 - router
 - controller
- EJS
 - config
 - views

Agenda(3)

- Test
 - mocha
 - chai
 - nyc

INTRODUCCIÓN



<https://v8.dev/>

- Convierte JavaScript en código de máquina
- Quién lo usa? Chrome, NodeJS, ...
- Sistema Operativo: Windows 7+, macOS 10.12+, Linux systems
- Arquitectura: x64, IA-32, ARM, *MIPS*
- Esta escrito en C++
- Standalone o Embebidas en otras aplicaciones C++

Qué es NodeJS?

JavaScript Runtime

- Esta escrito en C++
- Usa la V8
- Provee librerías que pueden ser utilizadas durante la ejecución de código JavaScript

Instalar NodeJS

1. <https://nodejs.org/en/download/> + next..next..next
2. <https://github.com/nvm-sh/nvm> + nvm install node latest

Comandos

```
node  
node -v  
node index.js
```

Comandos (2)

```
npm -v  
npm init  
npm install  
npm uninstall
```

Comandos (3)

```
npm start  
npm test  
npm build  
npm run script
```

Comandos (4)

```
npm list  
npm audit  
npm publish  
...
```

Hola Mundo

```
// index.js  
console.log('Hello World!');
```

```
node index.js
```

Hola Mundo --http

```
const http = require('http');
const server = http.createServer((req, res) => {
  res.writeHead(200);
  res.write('Hello World!');
  res.end();
});
server.listen(3000, () => {
  console.log('http://localhost:3000');
});
```

Eventos en el Cliente

```
const btn = document.getElementById('action');  
btn.addEventListener('click', event => { ... }):
```


Eventos en el Servidor

```
const http = require('http');

const server = http.createServer();

server.on('request', (req, res) => {
  res.writeHead(200);
  res.write('Hola Mundo');
  res.end();
});

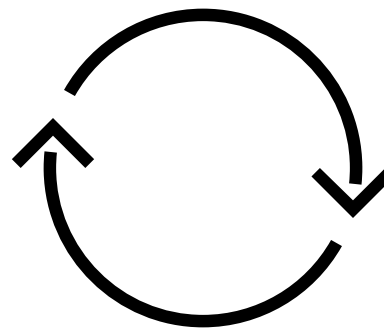
server.on('listening', () => {
  console.log('http://localhost:3000');
});

server.listen(3000);
```

Paradigma

- Event-driven - Asynchronous
- Non-blocking IO model - ***event loop***

Event loop



- **timers:** `setTimeout()` `setInterval()`
- **pending** callbacks: IO Callbacks
- **idle:** prepare - internal
- **poll:** new IO events
- **check:** `setImmediate()`
- **close callbacks:** `socket.on('close', ...)`

libuv



- Asynchronous IO!!!
- Asynchronous:
 - TCP/UDP, DNS resolution
 - fs, sockets
 - child process, signal handling
 - thread pool, high resolution clock
- <http://docs.libuv.org/en/v1.x/design.html>

Módulos

- Cada archivo es un módulo independiente
- Wrapper

```
(function(exports, require, module, __filename, __dirname) {  
  // Module code actually lives in here  
});
```

require/exports

```
// logger.js  
function info(message) {...}  
module.exports = { info };
```

```
// app.js  
const logger = require('./logger');  
logger.info('module system...')
```

npm package

- package.json ✓
- new project ✓

```
mkdir node-app-fs  
cd node-app-fs  
npm init  
npm init -y
```

FILE SYSTEM (fs)

Leer un archivo

```
const fs = require('fs');  
  
fs.readFile('./data.txt', (err, data) => {  
  if (err) throw err;  
  console.log(data);  
});
```

Escribir un archivo

```
fs.write('./message.txt', 'Hello', err => {  
  if (err) throw err;  
  console.log('done!');  
});
```

Stream

- Writable, Readable, Transform
- Instance of ***EventEmitter***

```
const {  
  Readable,  
  Writable,  
  Transform,  
} = require('stream');
```

fs.createReadStream

```
const fs = require('fs');
const rs = fs.createReadStream('./data.txt');
rs.on('data', chunk => {...});
rs.on('readable', () => {...});
rs.on('error', err => {...});
rs.on('pause', () => {...});
rs.on('resume', () => {...});
rs.on('end', () => {...});
rs.on('close', () => {...});
```

fs.createReadStream (2)

```
const fs = require('fs');  
const rs = fs.createReadStream('./data.txt',  
  { highWaterMark: 16 * 1024});  
rs.on('data', chunk => {  
  console.log(chunk.toString());  
});  
rs.on('end', () => {  
  console.log('----END')  
});
```

fs.createWriteStream

```
const fs = require('fs');  
const ws = fs.createWriteStream('./copy.txt');  
ws.on('error', err => {...});  
ws.on('drain', () => {...});  
ws.on('pipe', () => {...});  
ws.on('unpipe', () => {...});  
ws.on('finish', () => {...});  
ws.on('close', () => {...});
```

fs.createWriteStream (2)

```
const fs = require('fs');  
const rs = fs.createWriteStream('./example.txt');  
file.write('Hello World!');  
// Writing more now is allowed!  
file.end();  
// Writing more now is NOT allowed!
```

Ejercicio 1

- ***data.txt -> copy.txt***
- Readable Stream
- Writable Stream

HTTP

New Project

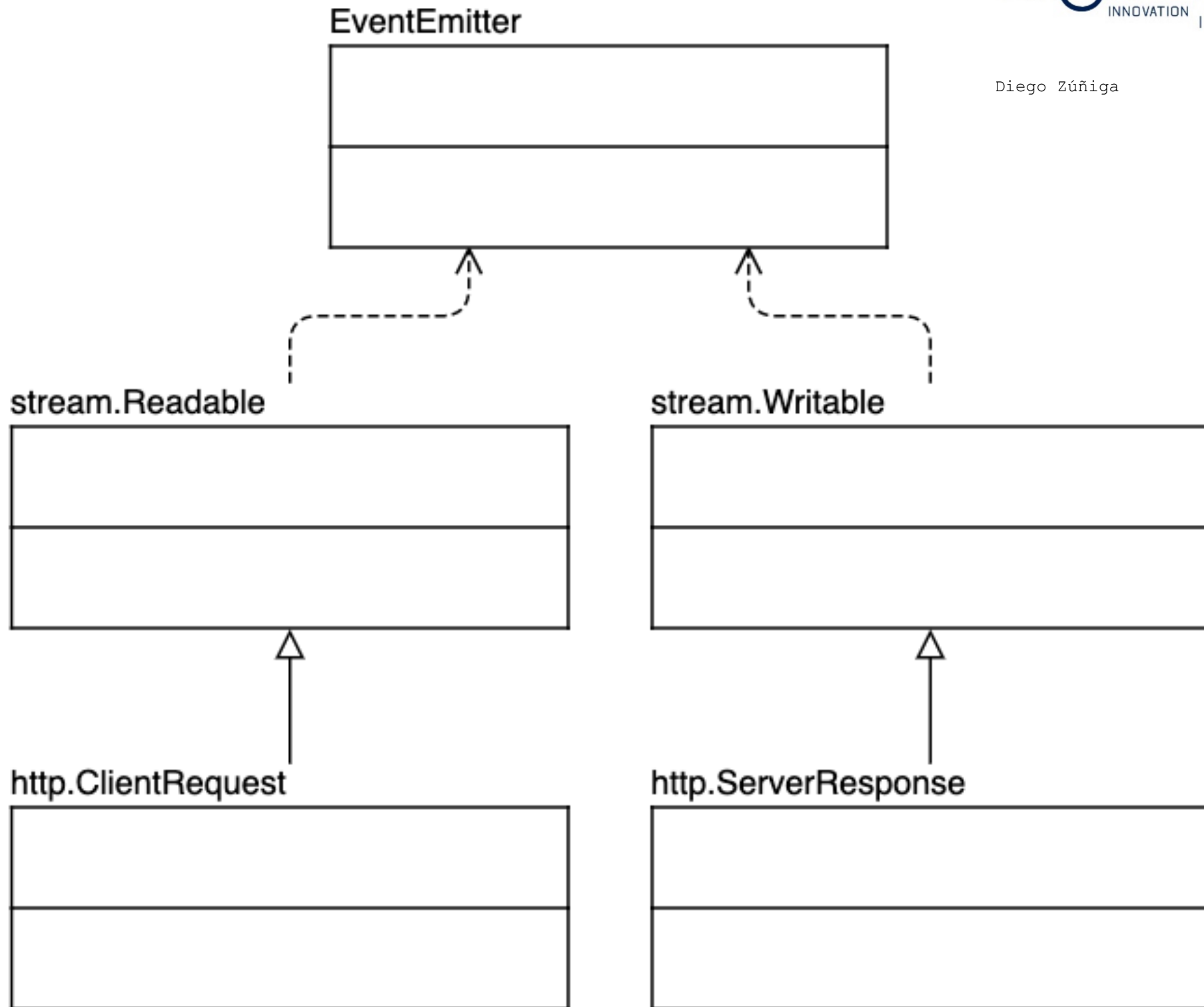
```
mkdir node-app-http  
cd node-app-http  
npm init
```

Hello world

```
const http = require('http');
const server = http.createServer((req, res) => {
  res.writeHead(200);
  res.write('Hello World!');
  res.end();
});
server.listen(3000, () => {
  console.log('http://localhost:3000');
});
```

Http + Streams

Diego Zúñiga



Echo Server

```
const server = http.createServer((req, res) => {  
  res.writeHead(200);  
  req.on('data', chunk => {  
    res.write(chunk);  
  });  
  req.on('end', () => {  
    res.end();  
  })  
});
```

Echo Server (2)

```
curl -d 'Streams' http://localhost:8080  
curl --upload-file nodejs.txt http://localhost:8080
```

Echo Server --pipe

```
const server = http.createServer((req, res) => {  
  res.writeHead(200);  
  req.pipe(res);  
});
```

Ejercicio 2

- ***data.txt -> copy.txt***
- pipe

Ejercicio 3

- http server
- carga de archivo
- progreso %

```
const contentLength = req.headers['content-length'];
```

Lectura

- backpressure
- <https://nodejs.org/es/docs/guides/backpressuring-in-streams/>

index.html

```
const http = require('http');
const fs = require('fs');
http.createServer((req, res) => {
  res.writeHead(200);
  const index = fs.createReadStream('./public/index.html');
  index.pipe(res);
});
```

EXPRESSJS

Introducción

- Framework para desarrollar aplicaciones web
- Basado en Rutas (URL + Callback)

```
mkdir node-app-express  
cd node-app-express  
npm init  
npm install --save express
```

Hello World

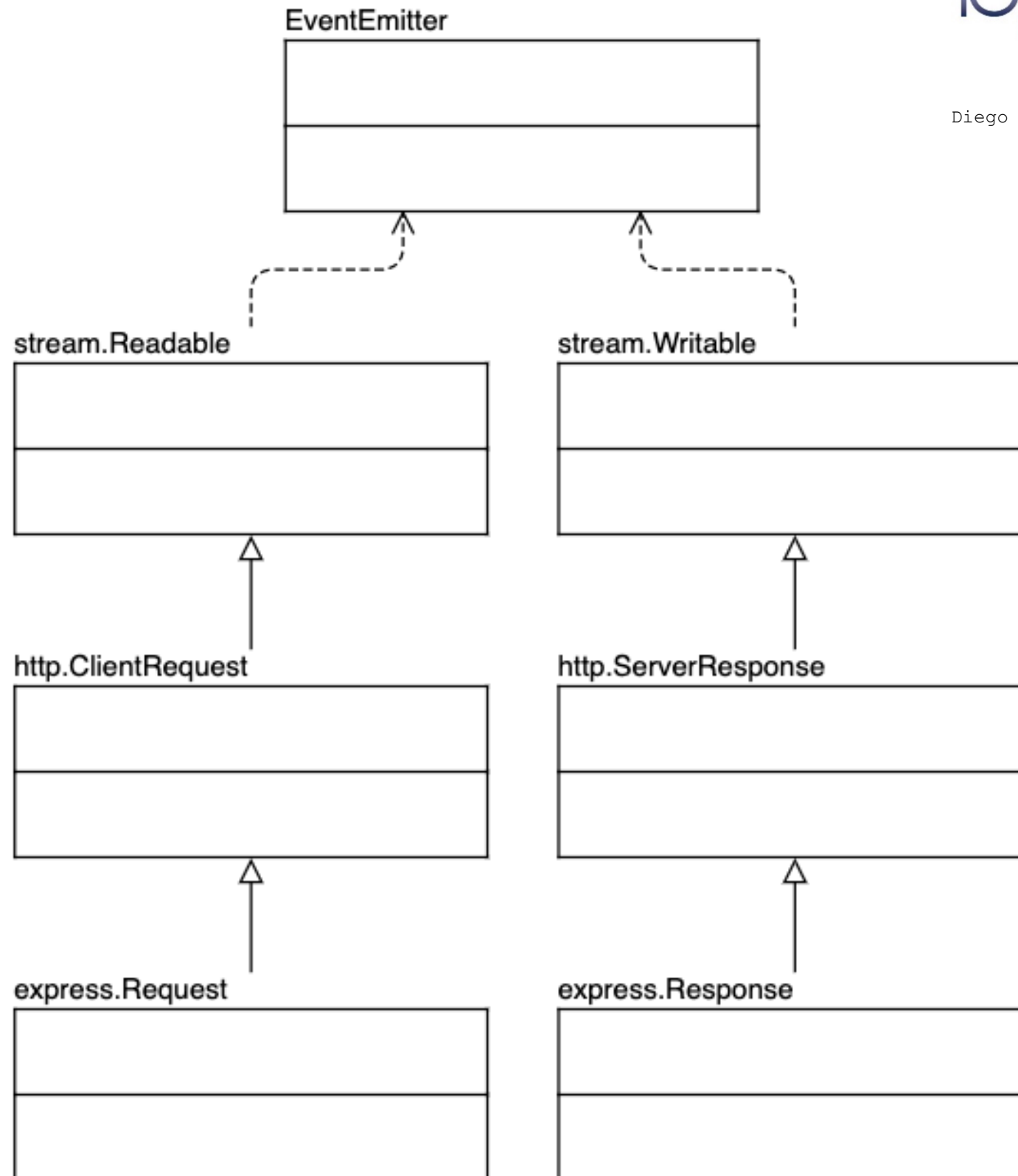
```
//index.js
const express = require('express');
const app = express();

app.get('/', (req, res) => {
  res.writeHead(200);
  res.write('Hello World! --express');
  res.end();
});

app.listen(3000, () => {
  console.log('http://localhost:3000');
})
```

Express + Streams

Diego Zúñiga



response.send

```
app.get('/', (req, res) => {  
  res.writeHead(200);  
  res.write('Hello World! --express');  
  res.end();  
});
```


response.send (2)

```
app.get('/', (req, res) => {  
  res.send('Hello World! --express');  
});
```

response.sendFile

```
app.get('/', (req, res) => {  
  res.writeHead(200);  
  const index = fs.createReadStream('./public/index.html');  
  index.pipe(res);  
});
```

response.sendFile (2)

```
app.get('/', (req, res) => {  
  res.sendFile('./public/index.html');  
});
```

response.send --JSON

```
app.get('/keywords', (req, res) => {  
  const keywords = ['nodejs', 'express'];  
  res.send(keywords);  
});
```

response.send --JSON (2)

```
curl -i http://localhost:3000/keywords
```

```
HTTP/1.1 200 OK
```

```
X-Powered-By: Express
```

```
Content-Type: application/json; charset=utf-8
```

```
Content-Length: 20
```

```
...
```

```
["nodejs","express"]
```

static

```
app.get('/', (req, res) => {  
  res.writeHead(200);  
  const index = fs.createReadStream('./public/index.html');  
  index.pipe(res);  
});  
...  
app.get('/', (req, res) => {  
  res.sendFile('./public/index.html');  
});
```

static + middleware

```
app.use(express.static('public'));
```

middleware

- Funciones que tienen acceso a los objetos **req**, **res**, y la función **next**
- Pueden realizar las siguientes tareas:
 - Ejecutar cualquier código antes y después de la ejecución del ciclo *request-response*
 - Modificar los objetos **req** y **res**
 - Terminar la ejecución del ciclo *request-response*

tipos de middleware

- Application-level middleware
- Router-level middleware
- Error-handling middleware
- Built-in middleware
- Thrid-party middleware

Application-level middleware

```
//logger.js
function logger(req, res, next) {
  next();
}

//index.js
const express = require('express');
const logger = require('./logger');

const app = express();
app.use(logger);
```

Ejercicio 4

- Application-level middleware **logger.js**
 - Mensaje de inicio
 - Mensaje de fin con tiempo total
 - Identificar el mensaje de inicio y fin con un ID único.
 - TIP: Eventos de un Writable Stream

```
`INICIO: ${id}: ${message}`  
`FIN: ${id}: ${message}, ${duration}`
```

```
ws.on('error', err => {...});  
ws.on('drain', () => {...});  
ws.on('pipe', () => {...});  
ws.on('unpipe', () => {...});  
ws.on('finish', () => {...});  
ws.on('close', () => {...});
```

Query Params

- <http://localhost:3000/keywords?limit=2>

```
app.get('/keywords', (req, res) => {  
  const limit = req.query.limit;  
  // use the limit query parameter!  
  const keywords = ['nodejs', 'express'];  
  res.send(keywords);  
});
```

Keywords

```
[  
  {  
    id: 'javascript',  
    desc: 'Lenguaje de Programación',  
    url: 'http://javascript.com'  
  },  
  {  
    id: 'javascript',  
    desc: 'Lenguaje de Programación',  
    url: 'http://javascript.com'  
  },  
  {  
    id: 'javascript',  
    desc: 'Lenguaje de Programación',  
    url: 'http://javascript.com'  
  }  
]
```

Rutas Dinámicas

- <http://localhost:3000/keywords/javascript>
 - {...}
- <http://localhost:3000/keywords/nodejs>
 - {...}
- <http://localhost:3000/keywords/express>
 - {...}

Rutas Dinámicas (2)

```
app.get('/keywords/:id', (req, res) => {  
  const id = req.params.id;  
  // use the name url parameter  
});
```

404 Not Found

```
app.get('/keywords/:id', (req, res) => {  
  const id = req.params.id;  
  const desc = ...;  
  if (!desc) {  
    return res.status(404).send(`${name} not found`);  
  }  
  return res.send(desc);  
});
```


Built-in middleware

- `express.static`
- `express.json`
- `express.urlencoded`

Ejercicio 6

- Usar los siguiente métodos HTTP
- GET/POST/PUT/DELETE/PATCH
- CRUD de keywords

CRUD

```
app.get('/keywords', (req, res) => {...}));  
app.get('/keywords/:id', (req, res) => {...}));  
app.post('/keywords', (req, res) => {...}));  
app.put('/keywords/:id', (req, res) => {...}));  
app.delete('/keywords/:id', (req, res) => {...}));
```

app.param

```
app.param('id', () => {  
  const id = req.params.id;  
  req.keywordId = name.toLowerCase();  
  next();  
});  
...  
app.get('/keywords/:id', function (req, res) {  
  const desc = ...req.keywordId...;  
  ...  
});
```

Routes

```
app.route('/keywords')  
  .get((req, res) => {...})  
  .post((req, res) => {...});  
  
app.route('/keywords/:id')  
  .get((req, res) => {...})  
  .put((req, res) => {...})  
  .delete((req, res) => {...});
```

Router

```
//keywords.js
const express = require('express');
const router = express.Router();

...

module.exports = router;
```

Router (2)

```
//keywords.js
router.route('/')
  .get((req, res) => {...})
  .post((req, res) => {...});

router.route('/:id')
  .get((req, res) => {...})
  .put((req, res) => {...})
  .delete((req, res) => {...});
```

Router (3)

```
//index.js
var express = require('express');
var keywords = require('./keywords');
var app = express();
...
app.use('/keywords', keywords);
```


route('...').all

```
router.route('/:id')
  .all(function (req, res, next) {
    var id = req.params.id;
    req.keywordId = id.toLowerCase();
    next();
  })
  .get(function (req, res) {...})
  .put(function (req, res) {...})
  .delete(function (req, res) {...});
```

file structure

- middleware.js
- routes.js
- api/
 - keywords/
 - controller.js
 - index.js
- index.js

middleware.js

```
const express = require('express');
function middleware(app) {
  app.use(express.static('public'));
  app.use(express.json());
  app.use(express.urlencoded({ extended: true }));
}
module.exports = middleware;
```

routes.js

```
const keywords = require('./api/keywords');  
function routes(app) {  
  app.use('/api/keywords');  
}  
module.exports = routes;
```

index.js

```
const express = require('express');  
const middleware = require('./middleware');  
const routes = require('./routes');  
const app = express();  
  
middleware(app);  
routes(app);  
  
app.listen(3000, () => {...});
```

api/keywords/controller.js

```
function create(req, res) {...}  
function update(req, res) {...}  
function remove(req, res) {...}  
function search(req, res) {...}  
function readById(req, res) {...}  
  
module.exports = { create, update, search, readById };
```

api/keywords/index.js

```
const express = require('express');
const controller = require('./controller');
const router = express.Router();

router.route('/')
  .get(controller.search)
  .post(controller.create);

router.route('/:id')
  .get(controller.readById)
  .put(controller.update)
  .delete(controller.remove);

module.exports = router;
```

MONGOOSE

start db

```
mongod --dbpath=./mongodb-data
```

install

```
npm install --save mongoose
```

database.js

```
const mongoose = require('mongoose');  
function connect() {  
  mongoose.connect(  
    'mongodb://localhost:27017/nodeapp',  
    { useNewUrlParser: true, poolSize: 10 }  
  );  
}  
  
module.exports = { connect };
```

database - error handling

```
const conn = mongoose.connection;  
conn.on('error', error => {  
  console.log('Error trying to connect to the db...');  
});  
conn.on('open', error => {  
  console.log('Db connection is ready...');  
});
```

schema - model

```
//api/keywords/model.js
const mongoose = require('mongoose');
const schema = new mongoose.Schema({
  name: String,
  desc: String,
  url: String,
});
const Keyword = mongoose.model('Keyword', schema);
module.exports = Keyword;
```

Validation - built-ins

```
const schema = new mongoose.Schema({  
  name: { type: String, required: true },  
  email: { type: String, required: true,  
    unique: [true, 'Email already exists'] },  
  password: { type: String, required: true,  
    select: false },  
});
```

Validation - custom

```
const schema = new mongoose.Schema({
  phone: {
    type: String,
    validate: {
      validator: function(v) {
        return /\d{3}-\d{3}-\d{4}/.test(v);
      },
      message: props =>
        `${props.value} is not a valid phone number!`
    },
    required: [true, 'User phone number required']
  }
});
```

hooks

```
schema.pre('save', function(next) {...});  
schema.pre('remove', function(next) {...});  
schema.pre('find', function(next) {...});  
...  
schema.post(...);
```


hooks example

```
// api/users/model.js
schema.pre('save', function(next) {
  const user = this;
  if (!user.isModified('password')) return next();
  const hashedPassword = hash(user.password, 10);
  user.password = hashedPassword;
  next();
});
```

Temas pendientes

- Subdocuments
- Populate
- Discriminators
- Plugins
- Aggregate
- ...

JSON Web Tokens

Dependencies

```
npm install --save jsonwebtoken bcryptjs
```

Tokens

```
jwt.sign(payload, SECRET_KEY, { expiresIn: '2h' });  
jwt.verify(token, SECRET_KEY);
```

Hash

```
await bcrypt.hash(text, 10);  
await bcrypt.compare(hash1, hash2);
```

Auth

```
//- api/auth/index.js
async function isLoggedIn(req, res, next) {...}
function createToken(user) {...}
async function createHash(text) {...}
async function compareHash(text, hash) {...}
```

EJS

Dependencies

```
npm install --save ejs
```

Config

```
const app = express();  
app.set('views', path.join(__dirname, 'views'));  
app.set('view engine', 'ejs');
```

Render

```
app.get('/', (req, res) => {  
  return res.render(  
    'index',  
    { message: 'EJS', values: ['a', 'b', 'c'] },  
  ),  
});
```

View

```
...  
<h1><%=message%></h1>  
...
```

View (2)

```
...  
<h1><%=message%></h1>  
<ul>  
    <% values.forEach(value => { %>  
        <li><%=value%></li>  
    <% }); %>  
</ul>  
...
```

Partials

- views
 - templates
 - about
 - keywords
 - partials
 - header
 - footer
 - menu

include

```
<!-- views/templates/about -->
<body>
  <header>
    <% include('../partials/header', {...}) %>
  </header>
  <p><%=message%></p>
  <footer>
    <% include('../partials/footer', {...}) %>
  </footer>
</body>
```

ENV

cross-env - dependencies

```
npm install --save-dev cross-env
```

cross-env - scripts

```
{  
  "scripts": {  
    "dev": "cross-env PORT=3000 SECRET_KEY=shhhhh nodemon index.js",  
    "start": "node index.js"  
  }  
}
```

cross-env - prod - unix

prod!!!

```
# unix  
export PORT=8080  
export SECRET_KEY=shhhhh+++  
npm start
```

cross-env - prod - win

prod!!!

```
# windows  
set PORT="8080"  
set SECRET_KEY="shhhhh+++"  
npm start
```

cross-env - process.env

```
//- index.js  
...  
const PORT = process.env.PORT || 3000;  
...  
app.listen(PORT, ...);
```

dotenv

```
npm install --save dotenv;
```

dotenv - init

```
//- index.js  
const dotenv = require('dotenv');  
dotenv.config();
```

dotenv - .env

```
PORT=3000  
SECRET_KEY=shhhh  
TOKEN_EXPIRES_IN=2h  
HASH_SALT=10  
DATABASE_URI=mongodb://localhost:27017/nodeapp
```


config

```
npm install --save config
```

config - files

- /config
 - default.json
 - qa.json
 - test.json
 - production.json
 - local.json <- .gitignore
- index.js

config - formats

- .yml, .yaml, .xml, .coffee, .cson,
- .properties, .json, .json5, .hjson,
- .ts, or .js

config - default.json

```
{
  "database": {
    "uri": "mongodb://localhost:27017/nodeapp"
  },
  "auth": {
    "secretKey": "shhhh",
    "tokenExpiresIn": "2h"
  },
  "crypt": {
    "hashSalt": 10
  },
  "server": {
    "host": "localhost",
    "port": 3000
  }
}
```

config - development.json

```
{  
  "auth": {  
    "tokenExpiresIn": "8h"  
  }  
}
```

config - production.json

```
{  
  "auth": {  
    "tokenExpiresIn": "4h"  
  }  
}
```

config - custom-environment-variables.json

```
{  
  "database": {  
    "uri": "MONGODB_URI"  
  },  
  "auth": {  
    "secretKey": "SECRET_KEY"  
  },  
  "server": {  
    "host": "HOST",  
    "port": "PORT"  
  }  
}
```

config - scripts

```
{  
  "scripts": {  
    "start": "cross-env NODE_ENV=production node index.js",  
    "dev": "cross-env NODE_ENV=development nodemon index.js"  
  }  
}
```


config - get

```
const config = require('config');  
...  
const { host, port } = config.get('server');
```

TESTING LIBRARIES

Mocha

- Testing Framework

```
{  
  "scripts": {  
    "test": "mocha"  
  }  
}
```

Chai

- Assert Library

```
const chai = require('chai');  
const expect = chai.expect;  
const assert = chai.assert;  
chai.should();  
  
const foo = 'bar';  
  
foo.should.equal('bar');  
expect(foo).to.be.a('string');  
assert.lengthOf(foo, 3);
```

Dependencies

```
npm install --save-dev mocha  
npm install --save-dev chai
```

Test example

```
//- test/hello.js
const chai = require('chai');
const expect = chai.expect;
const assert = chai.assert;
chai.should();

describe('Hello', () => {
  it('World', () => {
    const foo = 'bar';
    foo.should.equal('bar');
    expect(foo).to.be.a('string');
    assert.lengthOf(foo, 3);
  });
});
```

Test Mongoose

```
npm install --save-dev mongodb-memory-server
```

Test Model

```
let mongoServer, options;  
before(async () => {  
  mongoServer = new MongoMemoryServer();  
  options = config.get('database.options');  
  const uri = await mongoServer.getConnectionString();  
  console.log('MONGODB_URI', uri);  
  await mongoose.connect(uri, options);  
});  
after(async() => {  
  await mongoose.disconnect();  
  await mongoServer.stop();  
});
```



```
beforeEach(async () => {
  await Keyword.deleteMany();
});

it('should create a keyword', async () => {
  const newKeyword = await Keyword.create({
    name: 'fake-name',
    desc: 'fake-desc',
    url: 'fake-url'
  });
  expect(newKeyword).to.have.property('_id');
  expect(newKeyword).to.have.property('name')
    .to.equal('fake-name');
  expect(newKeyword).to.have.property('desc')
    .to.equal('fake-desc');
  expect(newKeyword).to.have.property('url')
    .to.equal('fake-url');

  const count = await Keyword.countDocuments();
  expect(count).to.equal(1);
});

it('should throw if there is no name', async () => {
  let hasError = false;
  try {
    await Keyword.create({
      desc: 'fake-desc',
      url: 'fake-url'
    });
  } catch {
    hasError = true;
  }
  expect(hasError).to.be.true;
});
```

Test Http

```
npm install --save-dev chai-http
```

```
chai.use(chaiHttp);  
chai.should();
```

Test Routes

```
it("should get all keywords", (done) => {  
  chai.request(app)  
    .get('/api/keywords')  
    .end((err, res) => {  
      res.should.have.status(200);  
      res.body.should.be.a('array');  
      done();  
    });  
});
```

FILE STRUCTURE

api

- api
 - controller
 - keyword-controller.js
 - user-controller.js
 - service
 - keyword-service.js
 - user-service.js
 - model
 - keyword-model.js
 - user-model.js
 - router
 - keyword-router.js
 - user-router.js
 - index.js

api - controller

```
//- api/controller/keyword-controller.js
const keywordService = require('../service/keyword-service');
async function create(req, res) {
  try {
    const newKeyword = req.body;
    const persistedKeyword =
      await keywordService.create(newKeyword);
    res.status(201).send(persistedKeyword);
  } catch (error) {
    res.status(400).send(error);
  }
}
module.exports = { create };
```

api - service

```
// - api/service/keyword-service.js
const keywordModel = require('../model/keyword-model');
async function create(newKeyword) {
  try {
    return await keywordModel.create(newKeyword);
  } catch (error) {
    throw handleError(error);
  }
}
module.exports = { create };
```

api - router

```
// - api/router/keyword-router.js
const express = require('express');
const keywordController =
  require('../controller/keyword-controller');
const router = express.Router();

router.route('/')
  .post(keywordController.create);

module.exports = router;
```


api - index

```
// - api/index.js
const keywordRouter = require('./router/keyword-router');
const router = express.Router();

router.use('/keywords');

module.exports = router;
```