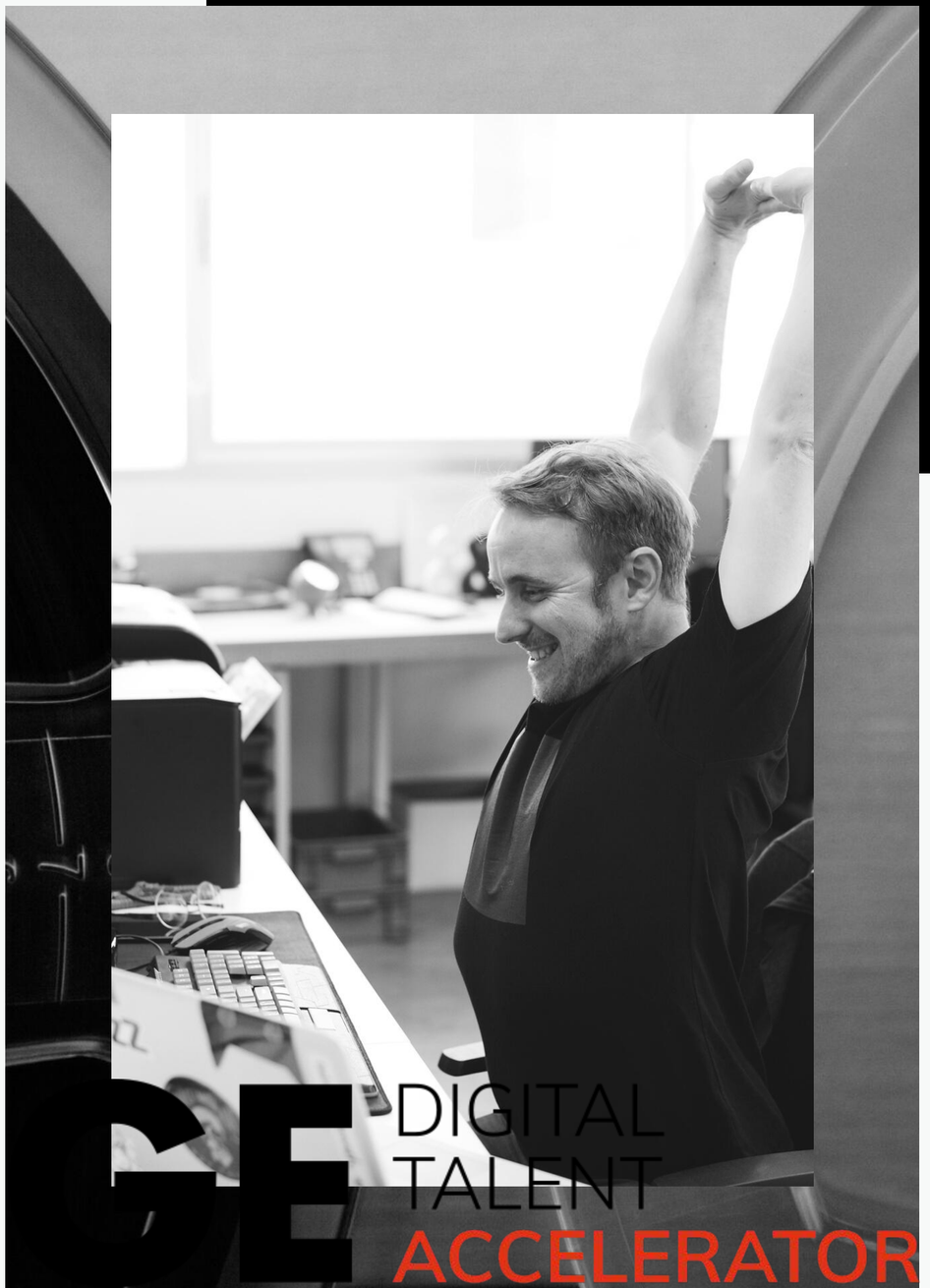
Express & SQL

Fundación ONCE | Inserta | Cofinanciado por la Unión Europea | Ministerio de Trabajo y Economía Social | Fondos Europeos

Índice



THE BRIDGE DIGITAL TALENT ACCELERATOR

Instalación

Express

```
$ npm install mysql2
```

Empezando

- Creamos un archivo llamado "index.js" y agregamos el siguiente código:

Importamos MySQL

```
const express = require("express");
const app = express();
const mysql = require('mysql2');
app.use(express.json())

const db = mysql.createConnection({
  host     : 'localhost',
  user     : 'root',
  password : '123456',
});

db.connect();
```

Creamos la conexión con la bd

Nos conectamos con la base de datos

Crear base de datos

- Creamos endpoint para crear nuestra base de datos:

```
app.get('/createdb',(req,res)=>{
  let sql ='CREATE DATABASE expressDB';
  db.query(sql,(err,result)=>{
    if(err)throw err;
    console.log(result);
    res.send('Database created...')
  })
})
```

Guardamos la query en una variable

Si hay algún error lanzamos una excepción

Implementar base de datos

- Añadimos la base de datos que hemos creado a la conexión con mysql

```
const db = mysql.createConnection({
  host     : 'localhost',
  user     : 'root',
  password : '123456',
  database:'expressDB'
});
```

añadimos la base de datos

Crear tabla

Aquí creamos nuestra tabla posts igual que lo haríamos con SQL, pero esta vez desde el código

```
app.get('/createpoststable',(req,res)=>{
   let sql = 'CREATE TABLE posts(id INT AUTO_INCREMENT,title VARCHAR(255), body VARCHAR(255), PRIMARY KEY(id))'
  db.query(sql,(err,result)=> {
    if(err) throw err;
    console.log(result);
    res.send('Posts table created...')
  })
})
```

Añadir un post

En una variable sql guardamos nuestra query de la siguiente forma:

```
app.post("/", (req, res) => {
  let sql = `INSERT INTO posts (title, body) values
    ('Post one', 'This is post number one');`;
  db.query(sql, (err, result) => {
    if (err) throw err;
    console.log(result);
    res.send("Post added...");
  });
});
```

Get posts

Creamos un endpoint para traernos los posts.

Para ello hacemos la query SQL para traernos los posts, y luego devolvemos el resultado.

```
app.get('/',(req,res)=> {
  let sql = 'SELECT * FROM posts';
  db.query(sql,(err,result)=> {
    if(err) throw err;
    res.send(result)
  })
})
```

Get post por id

Creamos un endpoint para traernos el post por su id.

Para ello hacemos la query SQL para traernos el post por su id, y le pasamos el id recogido en la request.

```
app.get('/id/:id',(req,res)=>{
  let sql = `SELECT * FROM posts WHERE id = ${req.params.id}`;
  db.query(sql,(err,result)=> {
    if(err) throw err;
    res.send(result)
  })
})
```

Actualizar un post

- Actualizamos el título del post por su id de la siguiente forma:

```
app.put('/id/:id',(req,res)=>{
  let newTitle = 'Updated Title';
  let sql = `UPDATE posts SET title = '${newTitle}' WHERE id = ${req.params.id}`;
  db.query(sql, (err,result)=> {
    if(err) throw err;
    res.send('Post updated...')
  })
})
```

Eliminar post por id

Creamos un endpoint para eliminar el post por su id.

Para ello hacemos la query SQL para eliminar el post por su id, y le pasamos el id recogido en la request.

```
app.delete('/id/:id',(req,res)=>{
  let sql = `DELETE FROM posts WHERE id = ${req.params.id}`;
  db.query(sql, (err,result)=> {
    if(err) throw err;
    res.send('Post deleted')
  })
})
```