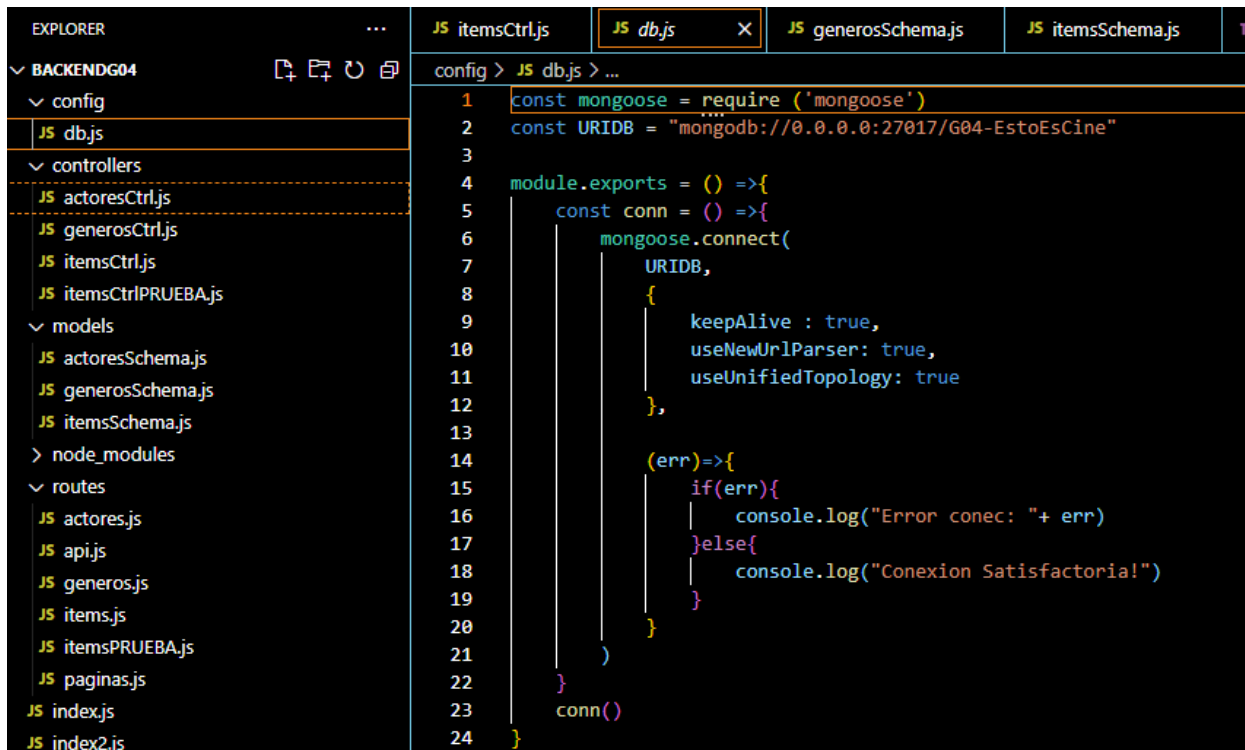


SPRINT 3: Desarrollo del Backend

Identificación Proyecto	
Nombre Proyecto:	EstoEsCine
Número Equipo:	01
Integrantes del equipo	
Rol (Líder-Desarrollador – Cliente)	Nombre
Líder/desarrollador	Daniel Fernando Gil García
Desarrollador	Paula Andrea Gutiérrez Avendaño
Desarrollador/cliente	Melissa Girón Rodríguez

Evidencia construcción del Backend

Como evidencia de la construcción del Backend, se debe presentar capturas de pantalla donde se visualice el proceso de construcción del Backend, como la creación en Node JS, modelo, controlador, rutas y vinculandose con la base de datos Mongodb.



```

EXPLORER
  BACKENDG04
    config
      JS db.js
    controllers
      JS actoresCtrl.js
      JS generosCtrl.js
      JS itemsCtrl.js
      JS itemsCtrlPRUEBA.js
    models
      JS actoresSchema.js
      JS generosSchema.js
      JS itemsSchema.js
    node_modules
    routes
      JS actores.js
      JS api.js
      JS generos.js
      JS items.js
      JS itemsPRUEBA.js
      JS paginas.js
      JS index.js
      JS index2.js

  JS itemsCtrl.js
  JS db.js
  JS generosSchema.js
  JS itemsSchema.js

config > JS db.js > ...
1  const mongoose = require ('mongoose')
2  const URIDB = "mongodb://0.0.0.0:27017/G04-EstoEsCine"
3
4  module.exports = () =>{
5    const conn = () =>{
6      mongoose.connect(
7        URIDB,
8        {
9          keepAlive : true,
10         useUrlParser: true,
11         useUnifiedTopology: true
12       },
13       (err)=>{
14         if(err){
15           console.log("Error conec: "+ err)
16         }else{
17           console.log("Conexion Satisfactoria!")
18         }
19       }
20     )
21   }
22   conn()
23 }
24
  
```

JS index.js	JS Login.js	JS Pelicula.js	JS Header.js	JS ListaPeliculas.js
src > JS index.js > ...				
<pre> 1 import React from "react"; 2 import ReactDOM from "react-dom/client" 3 import { BrowserRouter as Router, Routes,Route } from "react-router-dom"; 4 import "../css/bootstrap.min.css" 5 import {Menu } from "../components/Header"; 6 import {ListaPeliculas} from "../components/ListaPeliculas" 7 import { Login } from "../components/Login"; 8 const root= ReactDOM.createRoot(document.getElementById("root")) 9 root.render(<> 10 <Router> 11 <Menu/> 12 <div className="container"> 13 <div className="row my-5"> 14 <Routes> 15 <Route path="/" element={<ListaPeliculas/>}></Route> 16 <Route path="/login" element={<Login/>}></Route> 17 </Routes> 18 </div> 19 </div> 20 </Router> 21 </> 22) </pre>				

JS index.js	JS Login.js	JS Pelicula.js	JS Header.js	JS ListaPeliculas.js
src > components > JS ListaPeliculas.js > ListaPeliculas				
<pre> 1 import {useState,useEffect} from "react" 2 import {Pelicula} from "../Pelicula" 3 export function ListaPeliculas() 4 const [items,setItems]= useState([]) 5 console.log ("something") 6 function cargar(){ 7 fetch("http://localhost:3000/api/items") 8 .then (response => response.json()) 9 .then(data =>{console.log(data) 10 setItems(data) 11 }) 12 .catch(err=> console.log("error "+ err) 13 }) 14 useEffect(()=>{cargar(),[]}) 15 return <> 16 <div className="row my-2"> 17 <div className="container"> 18 <div className="row align-center"> 19 <div className="col m-5"> 20 <div classname="row"> 21 {items.map(dato=>{ 22 <Pelicula titulo={dato.titulo} 23 imagen={dato.imagen} 24 duracion={dato.duracion} 25 tipo={dato.tipo} 26 año={dato.año} 27 />}}) 28 </div></div></div></div></div> 29 </> </pre>				



ADD DATA		VIEW	Displaying documents 1 - 6 of 6		REFRESH
_id: ObjectId('637288859eb3dbaa58747ff5')					
titulo: "El Viaje de Chihiro"					
tipo: "Película"					
genero: "Fantasía"					
duracion: 125					
created_at: "2022-11-14"					
creado_por: "admin"					
actores: Array					
imagen: "https://ramenparados.com/wp-content/uploads/2021/05/POSTER_CHIHIRO_202..."					
ano: "2001"					
_id: ObjectId('637d5a20f05ecf27615cffe6')					
titulo: "El diario de Bridget Jones"					
tipo: "Película"					
genero: "comedia"					
duracion: 97					
created_at: "2022-11-22"					
creado_por: "admin"					
actores: Array					
imagen: "https://es.web.img2.acsta.net/pictures/14/02/19/10/39/284080.jpg"					
ano: "2001"					
_id: ObjectId('637d5c56f05ecf27615cfff0')					
titulo: "El silencio de los inocentes"					
tipo: "Película"					
genero: "suspense"					

G04-EstoEsCine.actores

10 DOCUMENTS 1 INDEXES

Documents	Aggregations	Schema	Explain Plan	Indexes	Validation
<div> <div>FILTER { field: 'value' }</div> <div>OPTIONS</div> <div>FIND</div> <div>RESET</div> <div>...</div> </div>					
ADD DATA		VIEW	Displaying documents 1 - 10 of 10		REFRESH
_id: ObjectId('6371722a58ee4060a2bb60f4')					
nombre: "Arnold Schwarzeneger"					
_id: ObjectId('6371724f58ee4060a2bb60f5')					
nombre: "Linda Hamilton"					
_id: ObjectId('6371727558ee4060a2bb60f6')					
nombre: "Leonardo Di Caprio"					
_id: ObjectId('6371728958ee4060a2bb60f7')					
nombre: "Kate Winslet"					
_id: ObjectId('637287279eb3dbaa58747feb')					
nombre: "Brendan Fraser"					
_id: ObjectId('637287439eb3dbaa58747fec')					
nombre: "Rachel Weisz"					



G04-EstoEsCine.generos

5 DOCUMENTS 1 INDEXES

Documents

Aggregations

Schema

Explain Plan

Indexes

Validation

FILTER

{ field: 'value' }

OPTIONS

FIND

RESET

ADD DATA

VIEW

Displaying documents 1 - 5 of 5

<

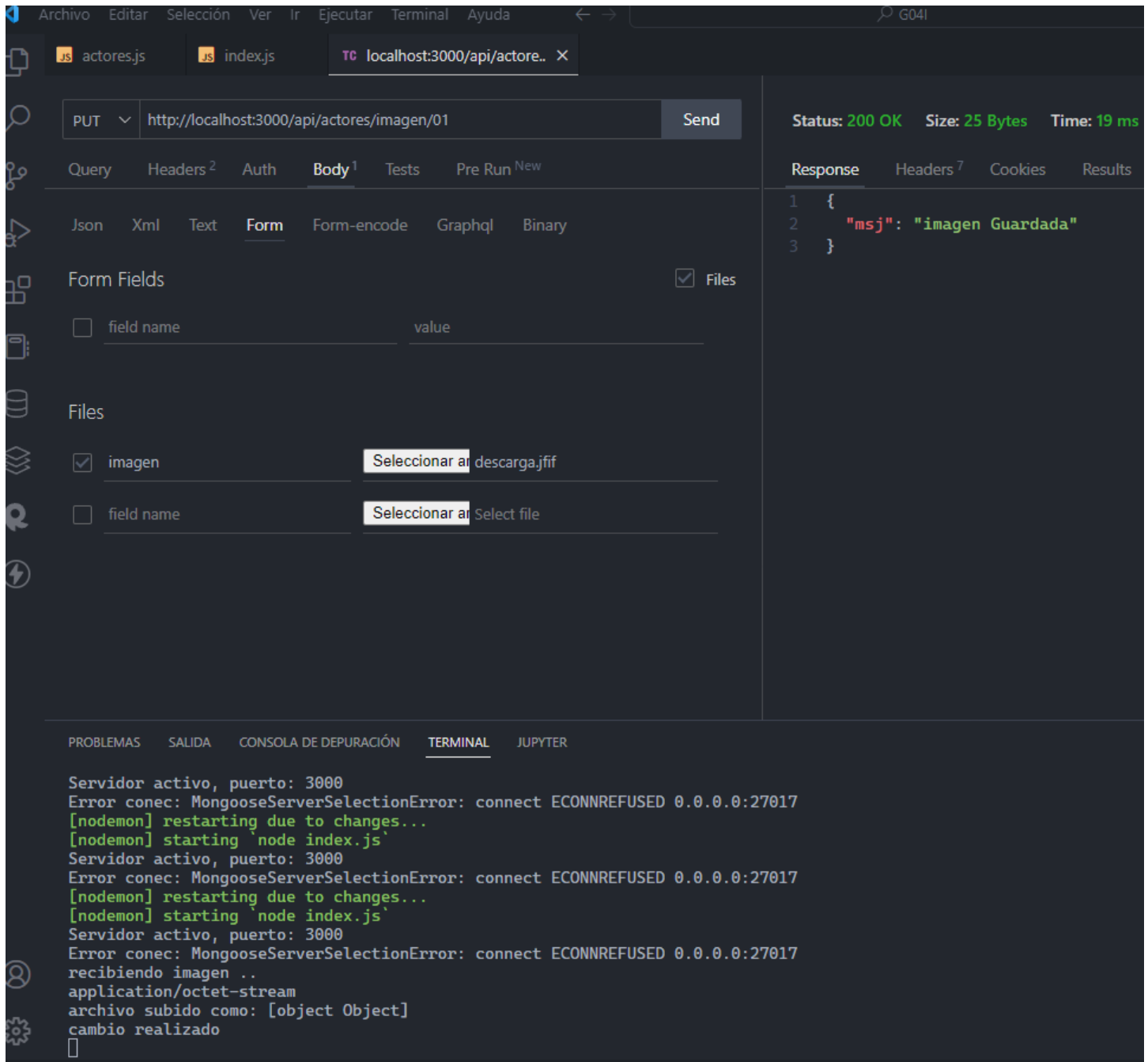
>

REFRESH

Evidencias de los “endpoint” con el consumo de recursos del API REST

Como evidencia de los “endpoint” donde se visualiza el consumo de recursos del API REST.

CARGA DE LA IMAGEN



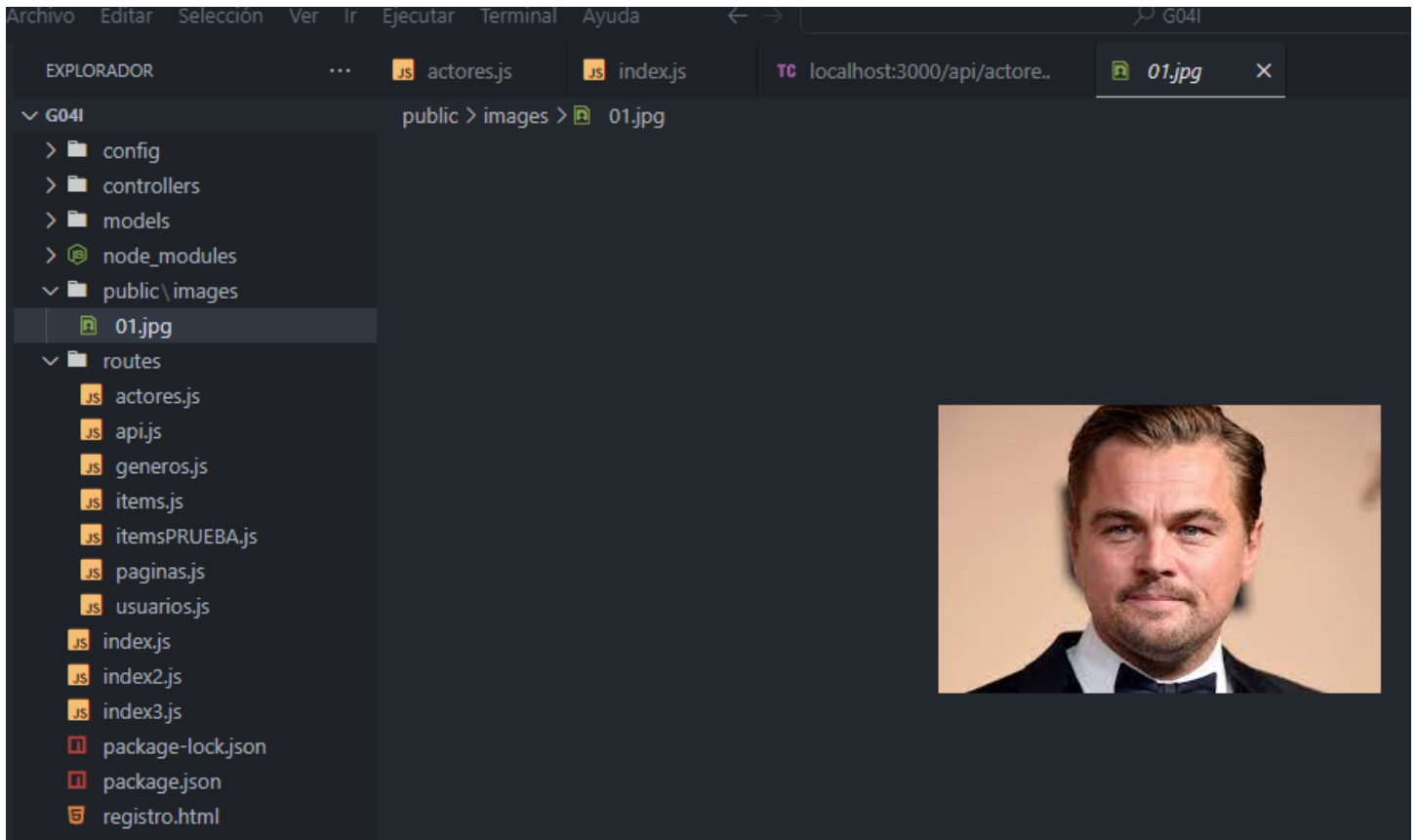
The screenshot shows a REST client interface with the following details:

- Request Method:** PUT
- URL:** `http://localhost:3000/api/actores/imagen/01`
- Status:** 200 OK
- Size:** 25 Bytes
- Time:** 19 ms
- Response Body:**

```
{
  "msj": "imagen Guardada"
}
```
- Form Fields:** A table with columns for field name and value.
- Files:** A table with columns for field name and value, showing the upload of a file named `imagen`.
- Terminal:**

```

Servidor activo, puerto: 3000
Error conec: MongooseServerSelectionError: connect ECONNREFUSED 0.0.0.0:27017
[nodemon] restarting due to changes...
[nodemon] starting 'node index.js'
Servidor activo, puerto: 3000
Error conec: MongooseServerSelectionError: connect ECONNREFUSED 0.0.0.0:27017
[nodemon] restarting due to changes...
[nodemon] starting 'node index.js'
Servidor activo, puerto: 3000
Error conec: MongooseServerSelectionError: connect ECONNREFUSED 0.0.0.0:27017
recibiendo imagen ..
application/octet-stream
archivo subido como: [object Object]
cambio realizado
  
```





Archivo Editor Selección Ver Ir Ejecutar Terminal Ayuda

EXPLORADOR ... JS actores.js JS index.js TC localhost:3000/api/actores.. X 01.jpg

▼ G04I

- > config
- > controllers
- > models
- > node_modules
- ▼ public\images
 - 01.jpg
 - 02.jpg
- ▼ routes
 - JS actores.js
 - JS api.js
 - JS generos.js
 - JS items.js
 - JS itemsPRUEBA.js
 - JS paginas.js
 - JS usuarios.js
- JS index.js
- JS index2.js
- JS index3.js
- package-lock.json
- package.json
- registro.html

PUT http://localhost:3000/api/actores/imagen/02 Send

Query Headers² Auth Body¹ Tests Pre Run New

Json Xml Text Form Form-encode GraphQL Binary

Form Fields ☒ Files

	field name	value
<input type="checkbox"/>		

Files

<input checked="" type="checkbox"/>	imagen	Seleccionar archivo 3712.webp
<input type="checkbox"/>	field name	Seleccionar archivo Select file

Status: 200 OK Size: 25 Bytes Time:

Response Headers⁷ Cookies Resu

```
1 {
2   "msj": "imagen Guardada"
3 }
```

PROBLEMAS SALIDA CONSOLA DE DEPURACIÓN TERMINAL JUPYTER

```
Servidor activo, puerto: 3000
Error conec: MongooseServerSelectionError: connect ECONNREFUSED 0.0.0.0:27017
recibiendo imagen ..
application/octet-stream
archivo subido como: [object Object]
cambio realizado
recibiendo imagen ..
image/webp
archivo subido como: [object Object]
cambio realizado
recibiendo imagen ..
image/webp
archivo subido como: [object Object]
cambio realizado
[]
```

> ESQUEMA

> LÍNEA DE TIEMPO

G04I

config

controllers

models

node_modules

public\images

01.jpg

02.jpg

routes

actores.js

api.js

generos.js

items.js

itemsPRUEBA.js

paginas.js

usuarios.js

index.js

index2.js


index3.js

package-lock.json

package.json

registro.html

public > images > 02.jpg



PROBLEMAS

SALIDA

CONSOLA DE DEPURACIÓN

TERMINAL

JUPYTER

Servidor activo, puerto: 3000
Error conec: MongooseServerSelectionError: connect ECONNREFUSED 0.0.0.0:27017
recibiendo imagen ..
application/octet-stream
archivo subido como: [object Object]
cambio realizado
recibiendo imagen ..
image/webp
archivo subido como: [object Object]
cambio realizado
recibiendo imagen ..
image/webp
archivo subido como: [object Object]
cambio realizado

> ESQUEMA

> LÍNEA DE TIEMPO

8

G04I

config

controllers

models

node_modules

public\images

01.jpg

02.jpg

03.jpg

routes

actores.js

api.js

generos.js

items.js

itemsPRUEBA.js

paginas.js

usuarios.js

index.js

index2.js

index3.js

package-lock.json

package.json

registro.html

PUT

http://localhost:3000/api/actores/imagen/03

Send

Query

Headers²

Auth

Body¹

Tests

Pre Run^{New}

Json

Xml

Text

Form

Form-encode

GraphQL

Binary

Form Fields

Files

field name

value

field name

value

imagen

Seleccionar archivo

image-foster-16536017450

field name

Seleccionar archivo

Select file

Status: 200 OK

Size: 25 Bytes

Time: 10 ms

Response

Headers⁷

Cookies

Results

1 {

2 "msj": "imagen Guardada"

3 }

PROBLEMAS

SALIDA

CONSOLA DE DEPURACIÓN

TERMINAL

JUPYTER

archivo subido como: [object Object]

cambio realizado

recibiendo imagen ..

image/webp

archivo subido como: [object Object]

cambio realizado

recibiendo imagen ..

application/octet-stream

archivo subido como: [object Object]

cambio realizado

recibiendo imagen ..

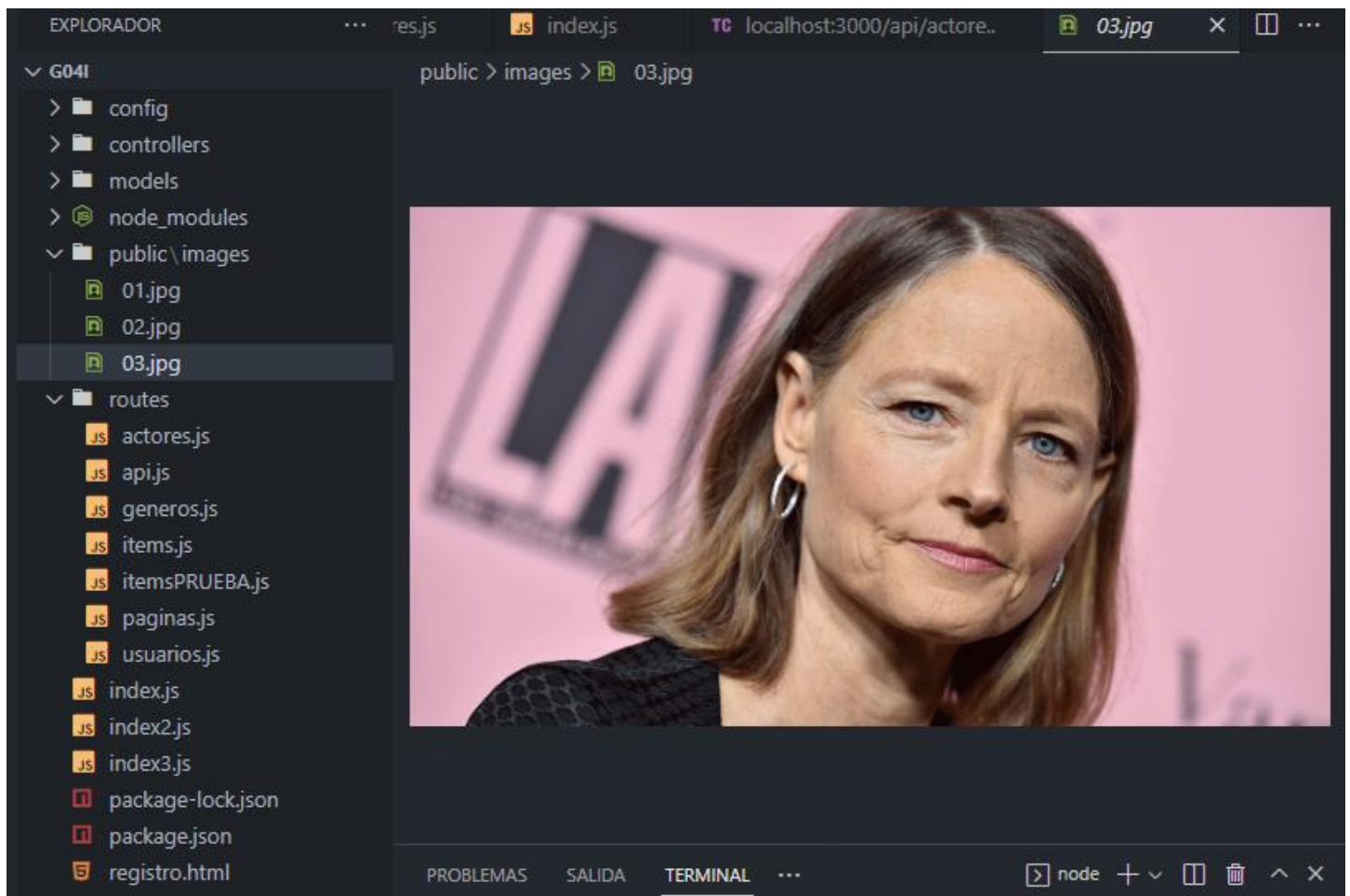
image/gif

archivo subido como: [object Object]

cambio realizado

ESQUEMA

LÍNEA DE TIEMPO



EXPLORADOR

G04I

config

controllers

models

node_modules

public\images

01.jpg

02.jpg

03.jpg

04.jpg

routes

actores.js

api.js

generos.js

items.js

itemsPRUEBA.js

paginas.js

usuarios.js

index.js

index2.js

index3.js

package-lock.json

package.json

registro.html

ESQUEMA

LÍNEA DE TIEMPO

PUT

http://localhost:3000/api/actores/imagen/04

Send

Query

Headers²

Auth

Body¹

Tests

Pre Run^{New}

Json

Xml

Text

Form

Form-encode

GraphQL

Binary

Form Fields

Files

field name

value

Files

imagen

Seleccionar archivo

5729_v9_bc.jpg

field name

Seleccionar archivo

Select file

Status: 200 OK

Size: 25 Bytes

Time:

Response

Headers⁷

Cookies

Resu

1 {

2 "msj": "imagen Guardada"

3 }

PROBLEMAS

SALIDA

CONSOLA DE DEPURACIÓN

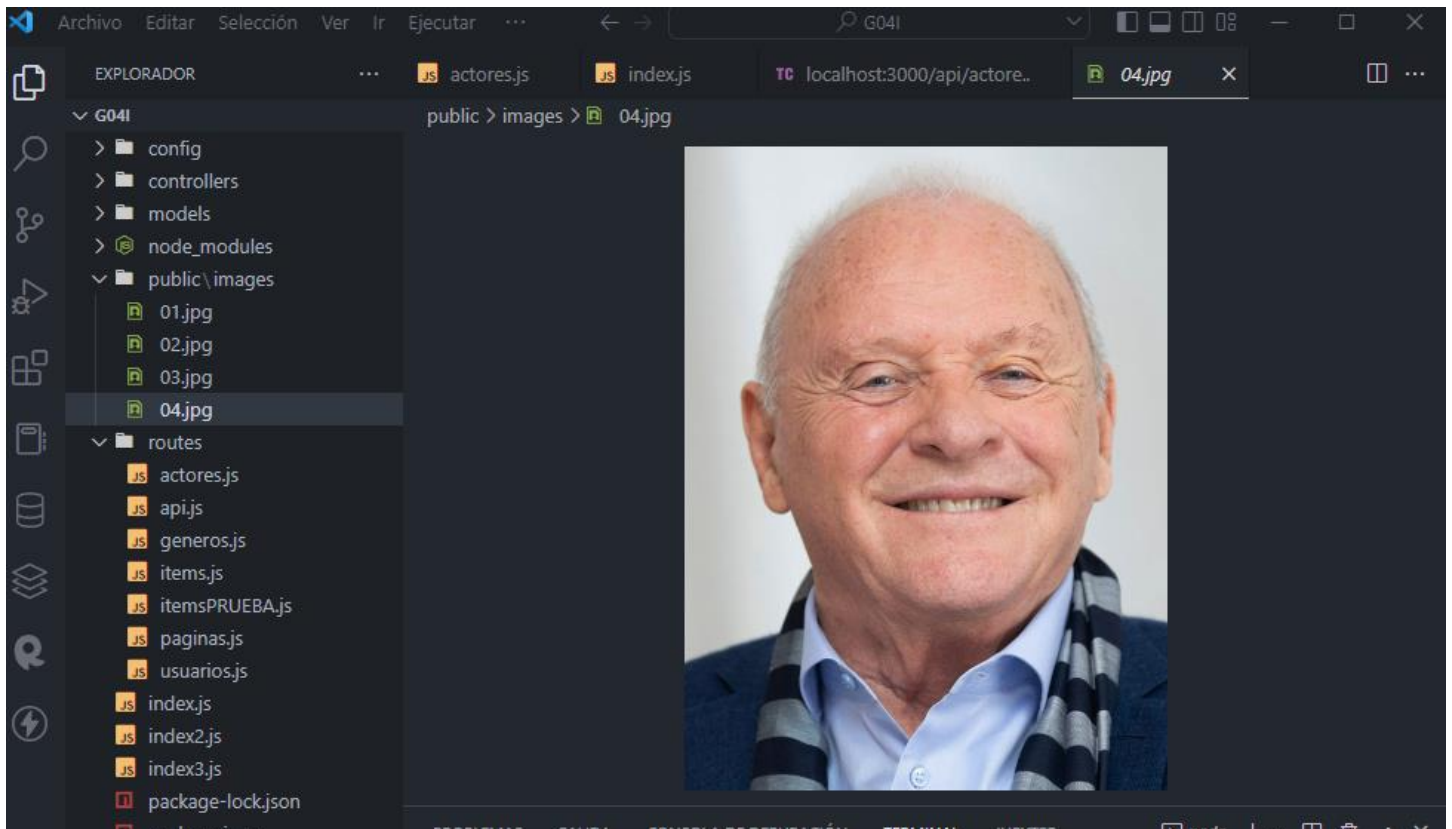
TERMINAL

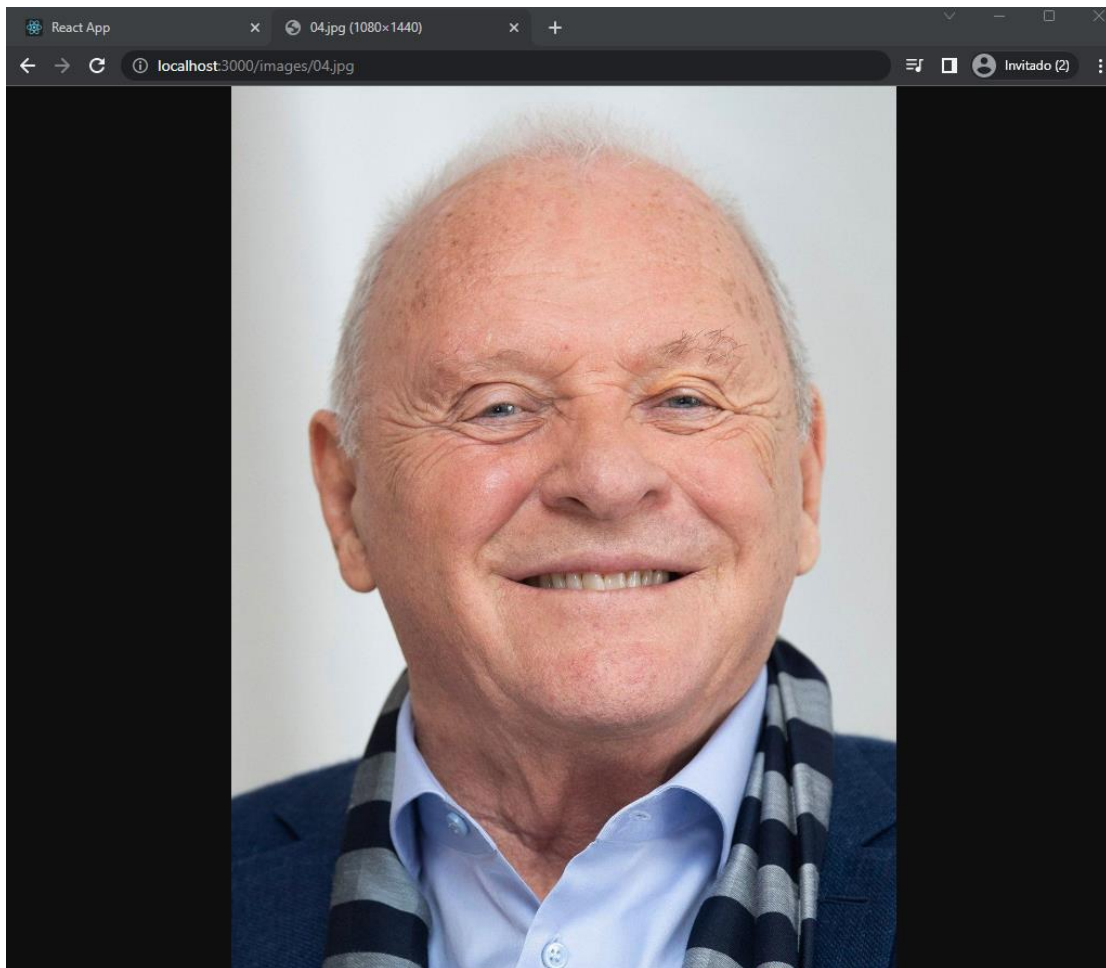
JUPYTER

```

archivo subido como: [object Object]
cambio realizado
recibiendo imagen ..
application/octet-stream
archivo subido como: [object Object]
cambio realizado
recibiendo imagen ..
image/gif
archivo subido como: [object Object]
cambio realizado
recibiendo imagen ..
image/jpeg
archivo subido como: [object Object]
cambio realizado

```







LOGIN DE USUARIO

The screenshot shows the Thunder Client interface. The 'New Request' tab is active, displaying a POST request to `http://localhost:3000/api/usuarios/login`. The request body is a JSON object: `{ "correo": "correoprueba@correo.com", "contrasena": "def456" }`. The response is a 200 OK status with a JSON body: `{ "msj": "OK" }`. The terminal at the bottom shows the server running on port 3000 and the login attempt being successful.

```
POST http://localhost:3000/api/usuarios/login
{
  "correo": "correoprueba@correo.com",
  "contrasena": "def456"
}
```

```
{
  "msj": "OK"
}
```

```
Conexion Satisfactoria!
[nodemon] restarting due to changes...
[nodemon] starting 'node index.js'
Servidor activo, puerto: 3000
Conexion Satisfactoria!
[nodemon] restarting due to changes...
[nodemon] starting 'node index.js'
Servidor activo, puerto: 3000
Conexion Satisfactoria!
intento de login
{ correo: 'correoprueba@correo.com', contrasena: 'def456' }
```

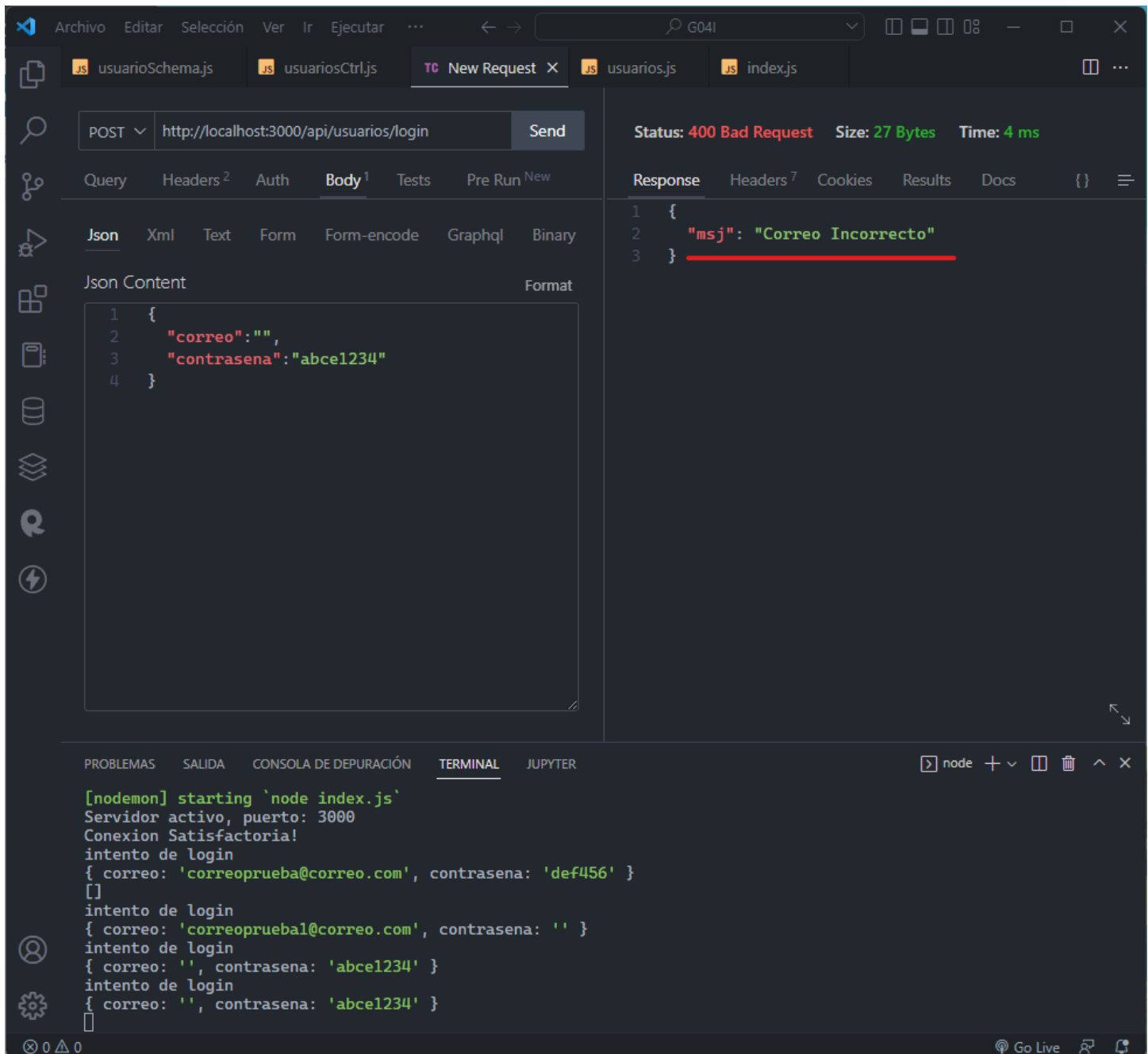
contraseña vacía

The screenshot displays the VS Code interface with a REST client tab. The request is a POST to `http://localhost:3000/api/usuarios/login` with a JSON body: `{ "correo": "correoprueba1@correo.com", "contrasena": "" }`. The response is a 400 Bad Request with a message: `"msj": "Contraseña Incorrecta"`. The terminal shows the server running on port 3000 and the client making two login attempts, the second of which fails due to an incorrect password.

```

[monodroid] starting 'node index.js'
Servidor activo, puerto: 3000
Conexion Satisfactoria!
[monodroid] restarting due to changes...
[monodroid] starting 'node index.js'
Servidor activo, puerto: 3000
Conexion Satisfactoria!
intento de login
{ correo: 'correoprueba@correo.com', contrasena: 'def456' }
[]
intento de login
{ correo: 'correoprueba1@correo.com', contrasena: '' }
[]
  
```

correo vacío



The screenshot shows the VS Code interface with a REST client tab. The request is a POST to `http://localhost:3000/api/usuarios/login`. The body is a JSON object:

```
1 {
2   "correo": "",
3   "contrasena": "abce1234"
4 }
```

The response is a 400 Bad Request with a status of 400, size of 27 bytes, and time of 4 ms. The response body is:

```
1 {
2   "msj": "Correo Incorrecto"
3 }
```

The terminal at the bottom shows the following output:

```
[nodemon] starting 'node index.js'
Servidor activo, puerto: 3000
Conexion Satisfactoria!
intento de login
{ correo: 'correoprueba@correo.com', contrasena: 'def456' }
[]
intento de login
{ correo: 'correopruebal@correo.com', contrasena: '' }
intento de login
{ correo: '', contrasena: 'abce1234' }
intento de login
{ correo: '', contrasena: 'abce1234' }
```

Usuario encontrado en la base de datos

The image shows a VS Code editor with a REST client tab. The request is a POST to `http://localhost:3000/api/usuarios/login` with a JSON body:

```
1 {
2   "correo": "admin@correo.com",
3   "contrasena": "abc123"
4 }
```

The response is a 200 OK status with a 12-byte body:

```
1 {
2   "msj": "OK"
3 }
```

The terminal at the bottom shows the following output:

```
[nodemon] restarting due to changes...
[nodemon] starting 'node index.js'
Servidor activo, puerto: 3000
Conexion Satisfactoria!
intento de login
{ correo: 'admin@correo.com', contrasena: 'abc123' }
{
  _id: new ObjectId("638299210b5d5ee6e47f7b05"),
  correo: 'admin@correo.com',
  contrasena: 'abc123',
  nombre: 'Melissa Giron'
}
```

creación de usuario nuevo



THUNDER CLIENT

New Request

Activity Collections Env

filter activity

POST localhost:3000/api/usuari... 3 mins ago

PUT localhost:3000/api/actores/... 3 hours ago

PUT localhost:3000/api/actores/... 4 hours ago

PUT thunderclient.com/welcome 4 hours ago

POST http://localhost:3000/api/usuarios/ Send

Status: 200 OK Size: 24 Bytes Time: 112 ms

Response Headers Cookies Results Docs

```
1 {
2   "msj": "Usuario Creado"
3 }
```

Json Content

```
1 {
2   "nombre" : "Paula G",
3   "correo" : "nuevopaula@correo.com",
4   "contrasena" : "123456"
5 }
```

Format

```
_id: ObjectId('6382c3e5839c82a4ff42f43d')
correo: "correopaula@correo.com"
contrasena: "$2a$10$wvRerTT0P.kr9hpkGUse.4I.YHg0Xpop0MVV2Z1JiRTZSFLg56d0"
nombre: "Paula G"
__v: 0
```

PROBLEMAS SALIDA CONSOLA DE DEPURACION TERMINAL

```
[nodemon] starting 'node index.js'
Servidor activo, puerto: 3000
Conexion Satisfactoria!
rs
[nodemon] starting 'node index.js'
Servidor activo, puerto: 3000
Conexion Satisfactoria!
{
  nombre: 'Paula G',
  correo: 'nuevopaula@correo.com',
  contrasena: '123456'
}
```


Go Live

Evidencia Frontend

EVIDENCIA DE LA REALIZACIÓN DE LA PÁGINA PRINCIPAL DE NUESTRO PROYECTO

React App

localhost:8000



Titanic

Película


Año: 1997

Duración: 195 minutos

Genero: Romance

Actores:

[Ver](#)



Terminator

Película


Año: 1984

Duración: 108 minutos

Genero: Ciencia Ficción

Actores:

[Ver](#)



La Momia

Película


Año: 1999

Duración: 124 minutos

Genero: Aventura

Actores:

[Ver](#)



El Viaje de Chihiro

Película


Año: 2001

Duración: 125 minutos

Genero: Fantasía

Actores:

[Ver](#)



El diario de Bridget Jones

Película


Año: 2001

Duración: 97 minutos

Genero: comedia

Actores:

[Ver](#)



El silencio de los inocentes

Película

Año: 1991

Duración: 118 minutos

Genero: suspenso

Actores:

[Ver](#)



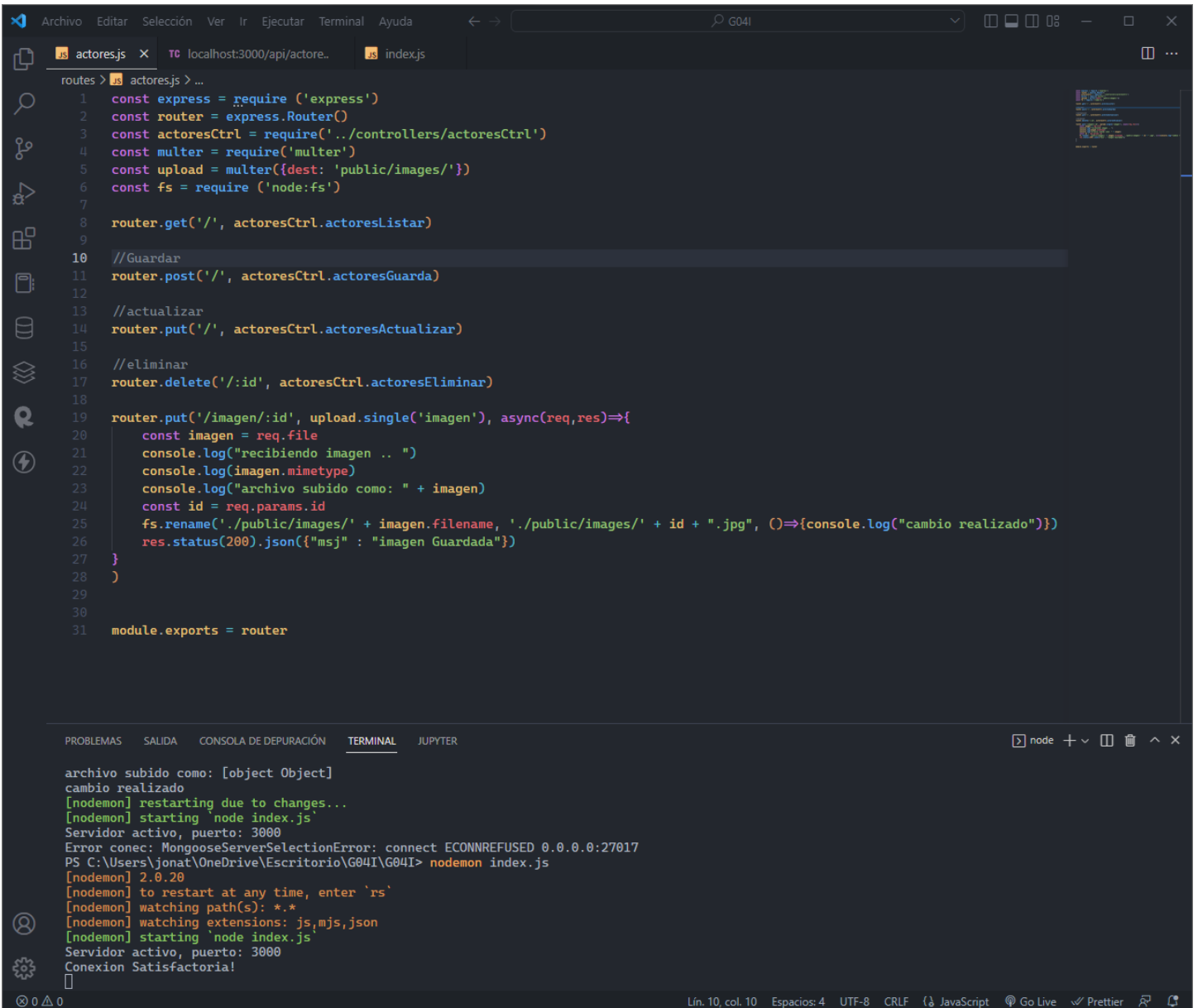
EVIDENCIA DE LA REALIZACIÓN DEL PROCESO DE LOGIN.

The screenshot shows a web application running on `localhost:8000/login`. The application has a blue header with a logo and the text "Inicio". Below the header, there is a login form with two input fields: "Correo Electronico" (Email) and "Contraseña" (Password). The email field contains the text "admin@correo.com". The password field is masked with dots. Below the input fields is a blue button labeled "Acceder" (Access). To the right of the input fields is a yellow button labeled "Login".

The DevTools console is open, showing a warning message: "Warning: Invalid DOM property `for`. Did you mean `htmlFor`?". The warning is from `react-dom.development.js:86`. Below the warning, there is a log message: "email click!!" from `Login.js:56`. Below that, there is a log message: "admin@correo.com" from `Login.js:7`. Below that, there is a log message: "Datos de Acceso Validos" from `Login.js:27`. The log message is an object: `{ "msj": "Datos de Acceso Validos" }`. A green bracket highlights the log messages from `Login.js:7` to `Login.js:27`.

Evidencia Backend

EVIDENCIA DE LA REALIZACIÓN DEL CÓDIGO QUE PERMITE LA CARGA DE IMÁGENES

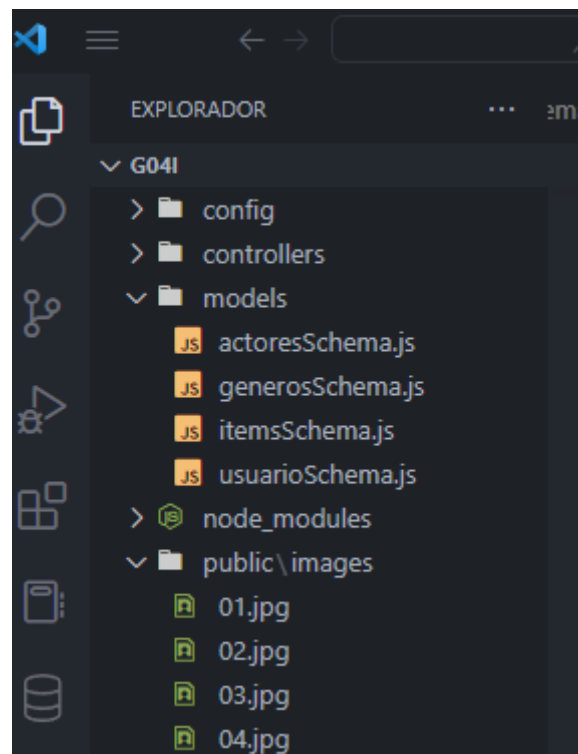


```

JS actores.js  TC localhost:3000/api/actore..  JS index.js  X
JS index.js > ...
22 //app.get('/about',(req,res)⇒{
23 //    res.send("Pagina cerca de ... ")
24 //})
25
26 ///////////////rutas
27 //app.use('/api/items', require('./routes/itemsPRUEBA'))
28 app.use(express.json())
29 app.use(cors())
30 //imagen
31 app.use(express.static('public'))
32

```

EVIDENCIA DEL DIRECTORIO PUBLIC FILES





Evidencia JIRA (Seguimiento del proyecto)

Como evidencia del seguimiento del proyecto con la metodología ágil SCRUM, utilizando el software JIRA, se debe presentar capturas de pantalla donde se visualice la ejecución de los Sprint con las historias de usuario relacionadas con el desarrollo del Backend.

<https://mgr708.atlassian.net/jira/software/projects/GG1PC/boards/5>

Proyectos / G04 Grupo 1 Proyecto Ciclo4

Tablero Sprint 3

⚡ ☆ 🕒 Restantes: 0 días

MR 👤 Epic ▾

AGRUPAR POR Nac

POR HACER

EN CURSO 1 INCIDENCIA

LISTO 2 INCIDENCIAS ✓

Visualización del apartado de Login y su respectiva funcionalidad en consola y encriptación de la contraseña
GG1PC-14

Visualización en la página web de las películas ingresadas en la base de datos
GG1PC-13 ✓

Cargar imágenes de los actores por medio de thunder en el backend
GG1PC-15 ✓

Evidencias de las Reuniones de Equipo

Como evidencia de las reuniones que efectúa el equipo del proyecto, presentar capturas de pantalla de las reuniones efectuadas y si lo consideran pertinente algunas actas de las reuniones.

