Actividad: Paso a Paso Express.js

Objetivo

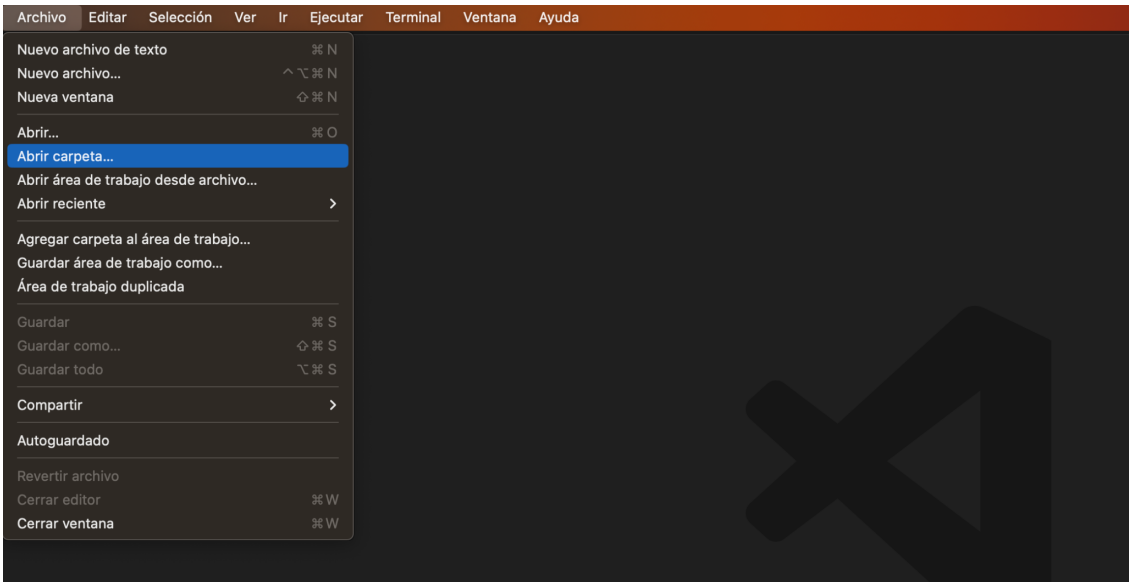
El objetivo de este trabajo práctico es que los alumnos desarrollen una API RESTful utilizando Express.js para la gestión de libros. La API permitirá realizar operaciones básicas como crear, leer, actualizar y eliminar libros, así como obtener una lista de todos los libros disponibles.

Consigna

1. La API debe seguir los principios de arquitectura RESTful y utilizar los métodos HTTP adecuados para cada operación.
2. La API debe implementar las siguientes rutas y funcionalidades:
    a. GET /libros: Devuelve la lista completa de libros.
    b. GET /libros/:id: Devuelve los detalles de un libro específico según su ID.
    c. POST /libros: Crea un nuevo libro con la información proporcionada.
    d. PUT /libros/:id: Actualiza la información de un libro específico según su ID.
    e. DELETE /libros/:id: Elimina un libro específico según su ID.
3. La API debe utilizar una estructura de archivos y carpetas organizada para separar las rutas y los controladores de libros.
4. La API debe manejar adecuadamente los errores y devolver respuestas JSON apropiadas en caso de errores.
5. Se debe implementar un archivo data.js que simule una base de datos y contenga una lista inicial de libros.

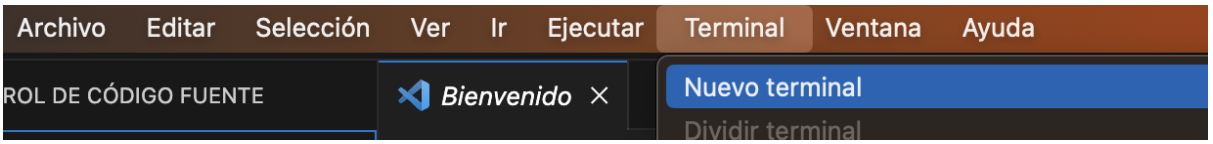
Desarrollo Paso a Paso

1. Iniciamos la aplicación VSCode.
2. Ir al menú Archivo -> Abrir Carpeta

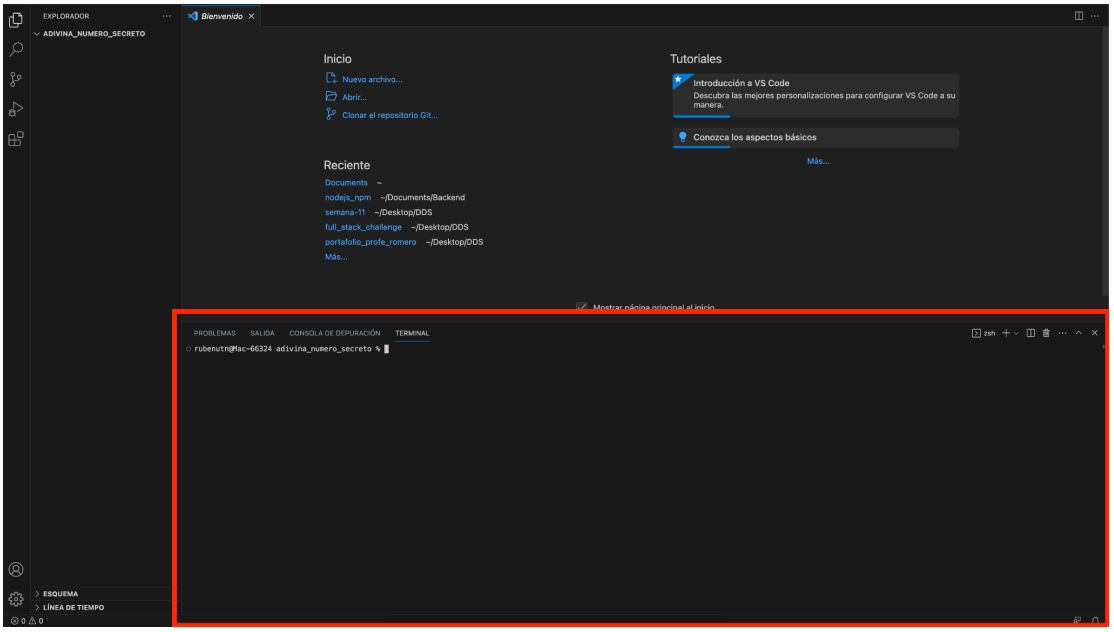


3. Del paso anterior se abre una ventana de búsqueda de carpetas, aquí buscamos la carpeta Documentos, seleccionamos y luego elegimos la opción “Nueva Carpeta”, indicando el nombre de la carpeta **api_libros**.
4. Selecciona la carpeta **api_libros** para que se abra en VSCode.

5. En VSCode buscamos en el menú la opción **Terminal -> Nuevo Terminal:**



6. En la siguiente imagen podemos ver en VSCode la sección de Terminal:



7. En el Terminal vamos a escribir el siguiente comando y luego apretar la tecla enter (para que se ejecute el comando):

```
npm init -y
```

Resultado esperado del comando:

```
> npm init -y
Wrote to /Users/rubenutn/Documents/api_libros/package.json:

{
  "name": "api_libros",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}
```

8. Vamos a instalar la librería **express**, que se utilizará para APIs. En el Terminal vamos a escribir el siguiente comando y luego apretar la tecla enter (para que se ejecute el comando):

```
npm install express
```

9. Seguimos instalando la librería **joi**, para esto en el Terminal vamos a escribir el siguiente comando y luego enter:

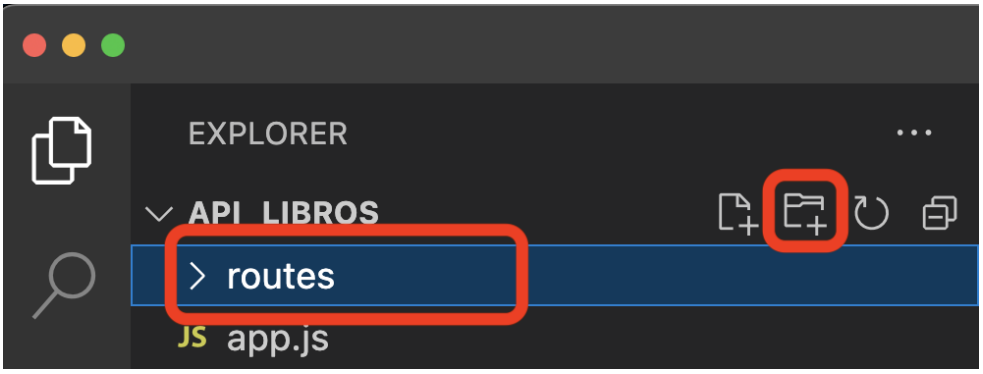
```
npm install joi
```

10. Continuamos creando en VSCode un archivo **app.js** con el siguiente código:

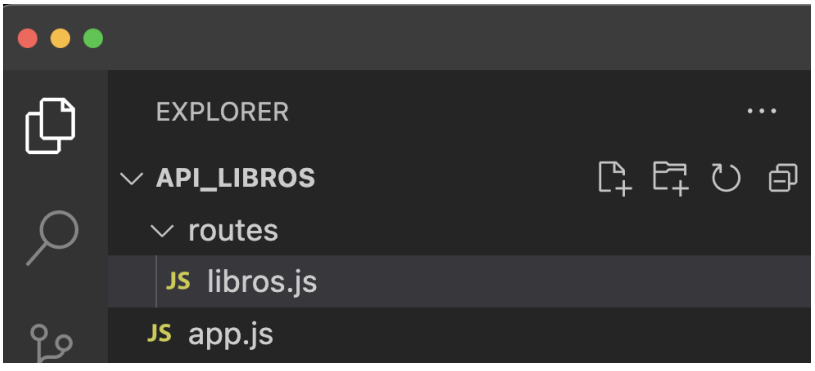
```
const express = require('express');
const app = express();
app.use(express.json());

// Importamos el Router de Libros
const librosRouter = require('./routes/libros');

// Importamos el Middleware Error Handler
const errorHandler = require('./middlewares/errorHandler');
```

```
app.use('/libros', librosRouter);

app.use(errorHandler);

app.listen(3000, () => {
  console.log('Servidor iniciado en el puerto 3000');
});
```

11. Ahora vamos a crear una carpeta **routes** para dejar todos los archivos de enrutamiento:



12. Creamos dentro de la carpeta **routes** el archivo **libros.js**:



13. Agregamos el siguiente código al archivo **libros.js**:

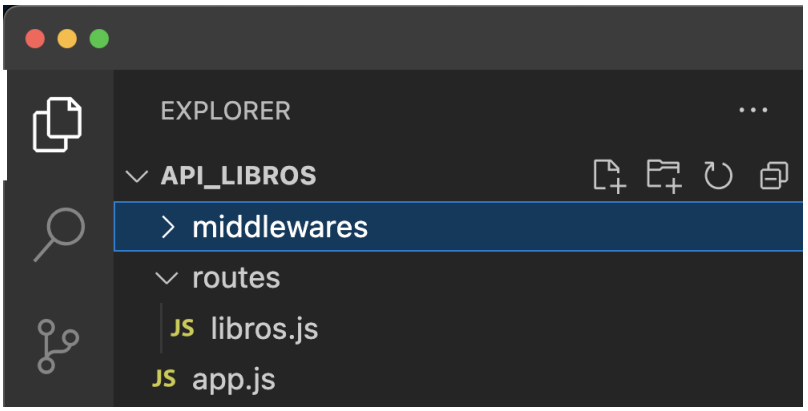
```
const express = require('express');
const router = express.Router();
const libros = require('../data');
const Joi = require('joi');
```

```
const libroSchema = Joi.object({
  titulo: Joi.string().required().label('Título'),
  autor: Joi.string().required().label('Autor')
});

// Obtener todos los libros
router.get('/', (req, res, next) => {
  try {
    res.json(libros);
  } catch (err) {
    next(err);
  }
});

// Obtener un libro por ID
router.get('/:id', (req, res, next) => {
  try {
    const id = req.params.id;
    const libro = libros.find((l) => l.id === id);

    if (!libro) {
      const error = new Error('Libro no encontrado');
      error.status = 404;
      throw error;
    }

    res.json(libro);
  } catch (err) {
    next(err);
  }
});

// Crear un nuevo libro
router.post('/', (req, res, next) => {
  try {
    const { error, value } = libroSchema.validate(req.body);
    if (error) {
      const validationError = new Error('Error de validación');
      validationError.status = 400;
      validationError.details = error.details.map(detail =>
detail.message);
      throw validationError;
    }
```

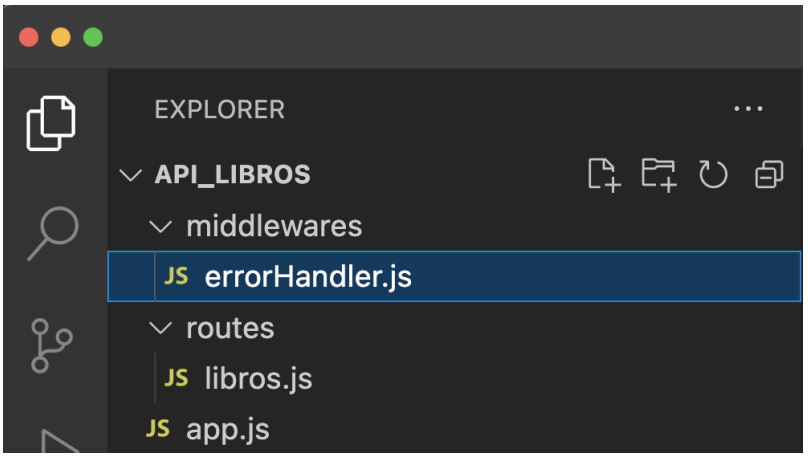
```
    const { titulo, autor } = value;

    const nuevoLibro = {
      id: libros.length + 1,
      titulo,
      autor
    };

    libros.push(nuevoLibro);
    res.status(201).json(nuevoLibro);
  } catch (err) {
    next(err);
  }
});

// Actualizar un libro existente
router.put('/:id', (req, res, next) => {
  try {
    const id = req.params.id;
    const { error, value } = libroSchema.validate(req.body);
    if (error) {
      const validationError = new Error('Error de validación');
      validationError.status = 400;
      validationError.details = error.details.map(detail =>
detail.message);
      throw validationError;
    }

    const { titulo, autor } = value;

    const libro = libros.find((l) => l.id === id);

    if (!libro) {
      const error = new Error('Libro no encontrado');
      error.status = 404;
      throw error;
    }

    libro.titulo = titulo || libro.titulo;
    libro.autor = autor || libro.autor;

    res.json(libro);
  } catch (err) {
```

```
    next(err);
  }
});

// Eliminar un libro
router.delete('/:id', (req, res, next) => {
  try {
    const id = req.params.id;
    const index = libros.findIndex((l) => l.id === id);

    if (index === -1) {
      const error = new Error('Libro no encontrado');
      error.status = 404;
      throw error;
    }

    const libroEliminado = libros.splice(index, 1);
    res.json(libroEliminado[0]);
  } catch (err) {
    next(err);
  }
});

module.exports = router;
```

14. Vamos a crear una carpeta **middlewares** para luego vamos a definir los **middlewares**:



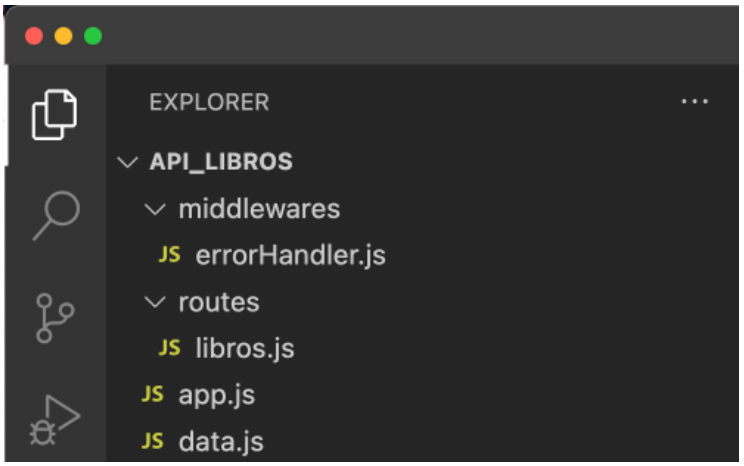
15. Agregamos un middleware para el manejo de errores. Para eso creamos un archivo **errorHandler.js** en la carpeta **middlewares**:



Le agregamos el siguiente código:

```
const errorHandler = (err, req, res, next) => {
  console.error(err);
  res.status(err.status || 500).json({ error: err.message ||
'Error en el servidor' });
};

module.exports = errorHandler;
```

16. Creamos el archivo **data.js** donde vamos a definir una variable con la lista de libros para usar en la API Libros para base de datos:



Le agregamos el siguiente código:

```
const libros = [
  { id: '1', titulo: 'Libro 1', autor: 'Autor 1' },
  { id: '2', titulo: 'Libro 2', autor: 'Autor 2' },
  { id: '3', titulo: 'Libro 3', autor: 'Autor 3' }
];
module.exports = libros;
```

17. Abrimos una terminal en Visual Studio Code. Para hacerlo, selecciona "Terminal" en la barra de menú superior y luego selecciona "Nueva terminal".

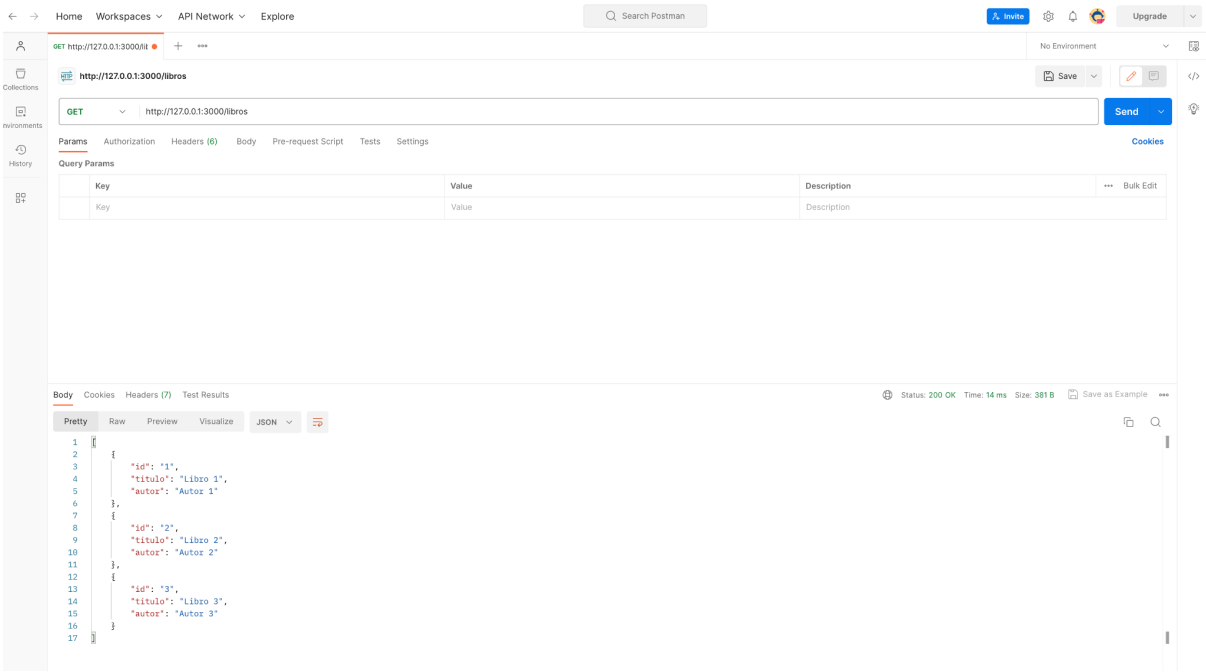
18. Ejecutamos la aplicación utilizando el Terminal de VSCode ejecutando el siguiente comando:

```
node app.js
```

Resultado esperado:

Si todo está bien, vas a ver un mensaje en la consola que dice "Servidor iniciado en el puerto 3000".

19. Abrimos la aplicación Postman y escribimos la dirección "**http://127.0.0.1:3000/libros**" en la barra de Url, hacemos click en **Send**. Deberíamos obtener el siguiente resultado:

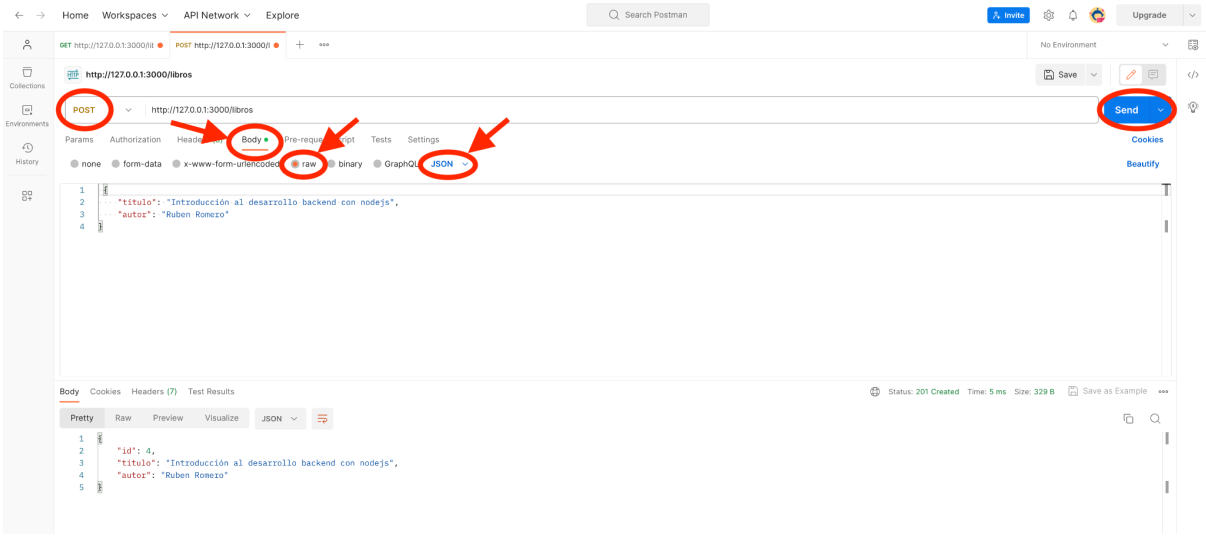


20. Ahora vamos a probar dar de alta un nuevo libro, para eso vamos a:
    a. Utilizar el método HTTP POST
    b. En la sección **Body** vamos a seleccionar la opción **raw**, luego la opción **JSON.**

c. Vamos a copiar y pegar el JSON en el body :

```
{
  "titulo": "Introducción al desarrollo backend con nodejs",
  "autor": "Ruben Romero"
}
```

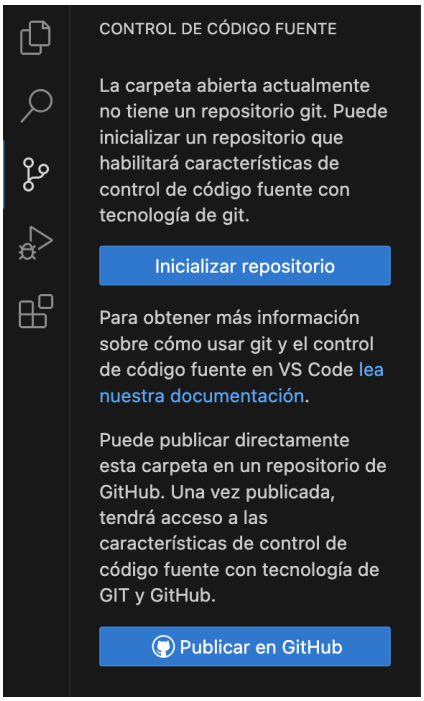
d. y finalmente vamos a enviar la petición con el **botón Send.**



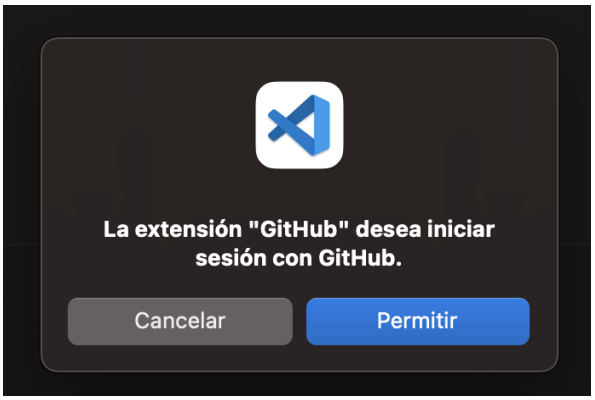
Como vemos el resultado esperado en la respuesta es:

```
{
    "id": 4,
    "titulo": "Introducción al desarrollo backend con nodejs",
    "autor": "Ruben Romero"
}
```

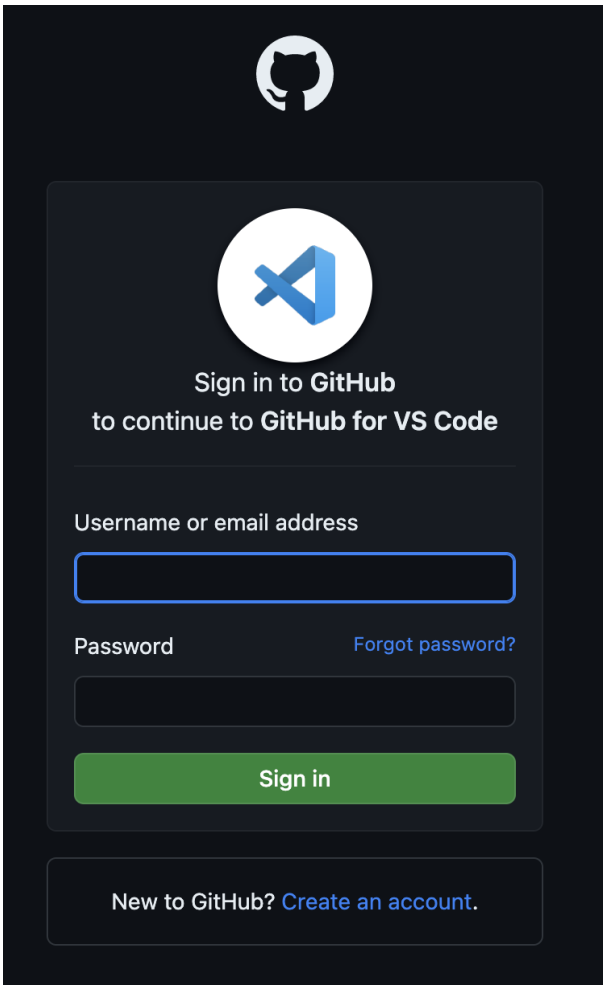
21. Ahora volvemos a VSCode y vamos a llevar a GitHub el código fuente generado usando la herramienta de git de VSCode. Hacemos click en la opción “Publicar en GitHub”:



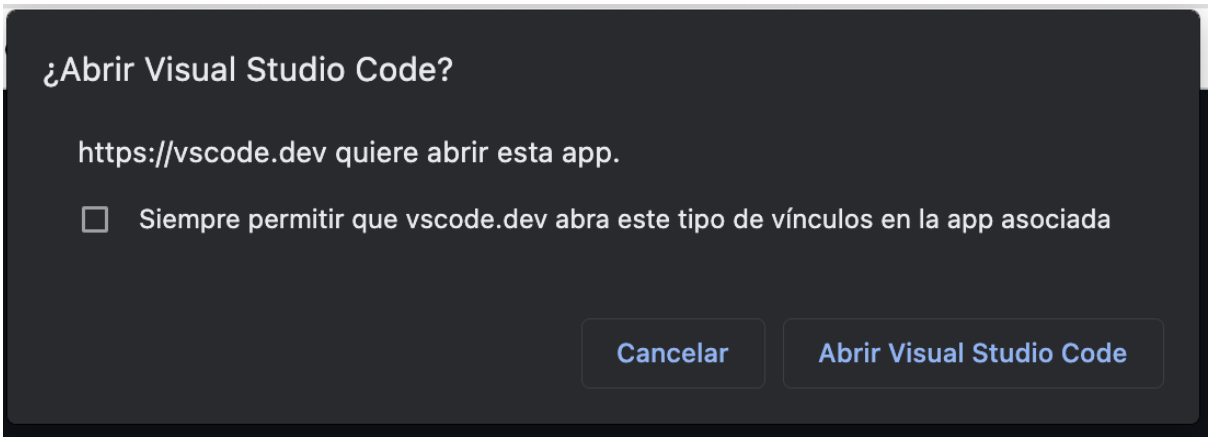
22. VSCode muestra este mensaje para que le darle permisos a iniciar sesión en GitHub a través de un navegador:



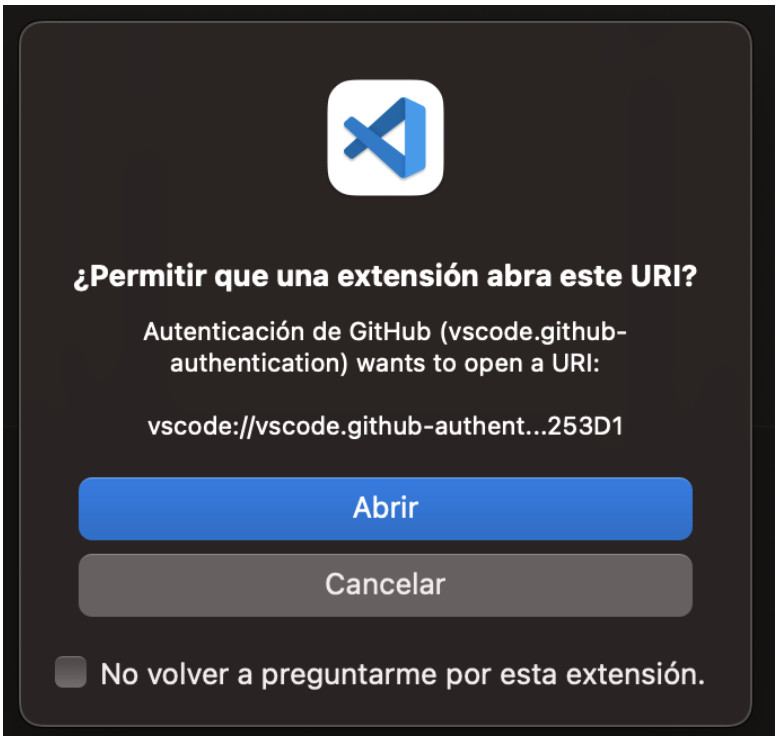
23. VSCode nos lleva al navegador web que tengamos por defecto y nos pide que ingresemos los datos de nuestra cuenta GitHub (el que no tenga cuenta se crea una nueva con su correo electrónico):



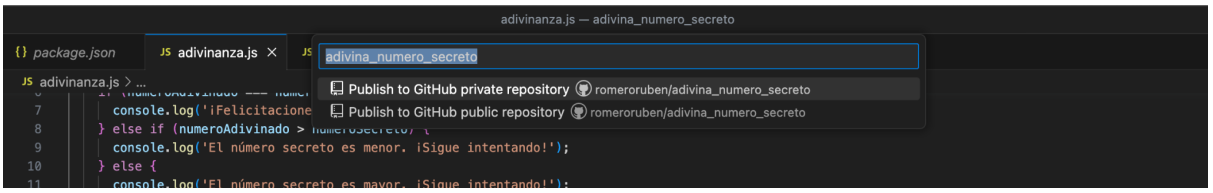
24. El navegador solicita que permitamos abrir VSCode, seleccionamos la opción “Abrir Visual Studio Code”



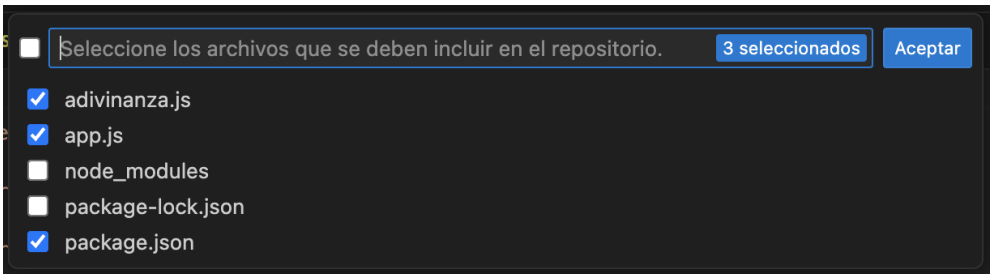
25. VSCode nos pide permisos para abrir una url, hacemos click en la opción “Abrir”:



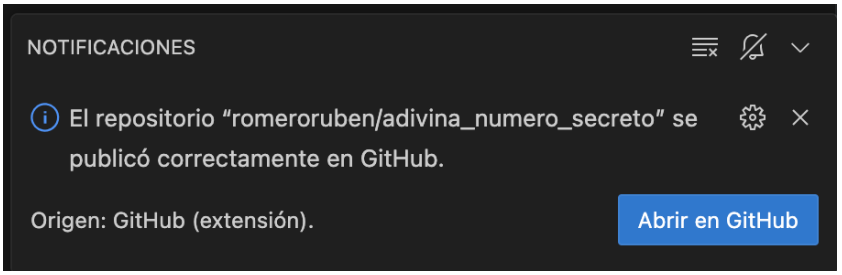
26. Ahora debemos seleccionar que tipo de repositorio queremos crear, seleccionar “Publish to GitHub **public** repository”, esto significa que vamos a publicar nuestro código en forma pública.



27. A continuación seleccionamos los archivos y carpetas que queremos “versionar” y llevar a GitHub.

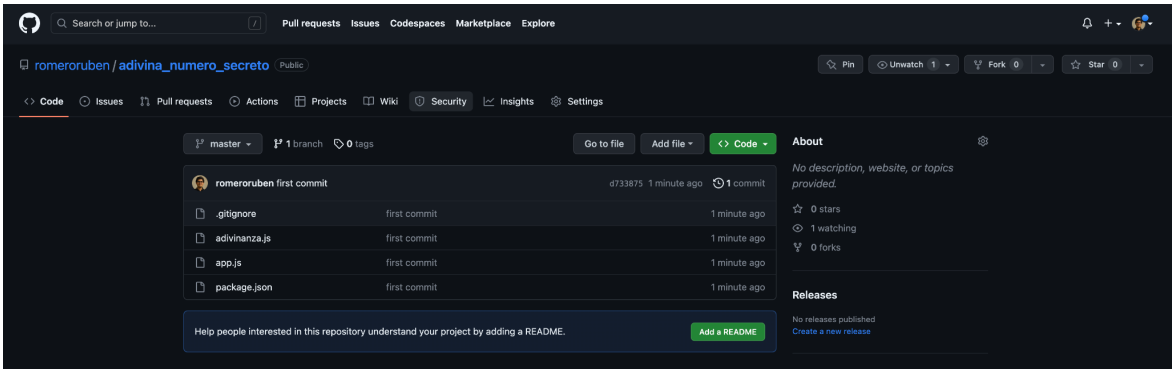


28. Esperamos unos minutos hasta que en VSCode podemos ver la siguiente notificación (abajo a la derecha), hacemos click en “Abrir en GitHub”



:

29. Se abre un navegador con el repositorio de GitHub disponible para que cualquiera lo pueda acceder:



30. IMPORTANTE! Para dar por finalizada la actividad compartir la url de GitHub en la actividad del aula virtual en https://uve.frc.utn.edu.ar/.