**Tarea - TIA-03**

* **Modalidad de Evaluación: En grupo**
* **Peso: 10% (de la nota final)**
* **Metodología**: Aprendizaje Basado en Problemas (ABR)

**EQUIPO: \_\_D\_\_\_\_**

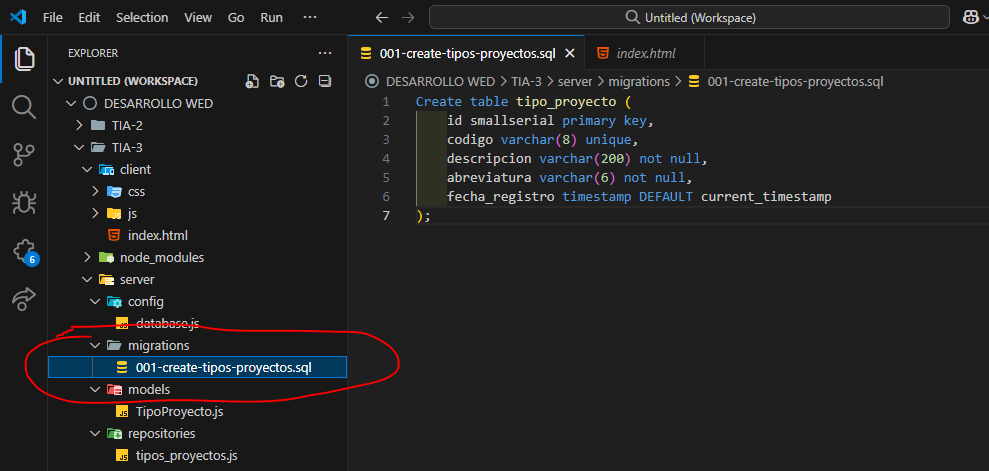
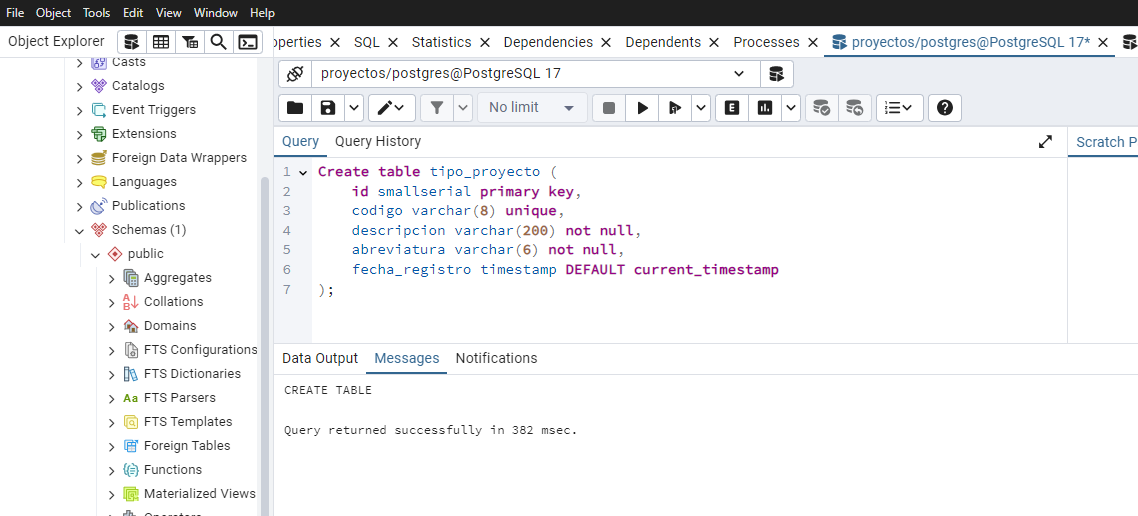
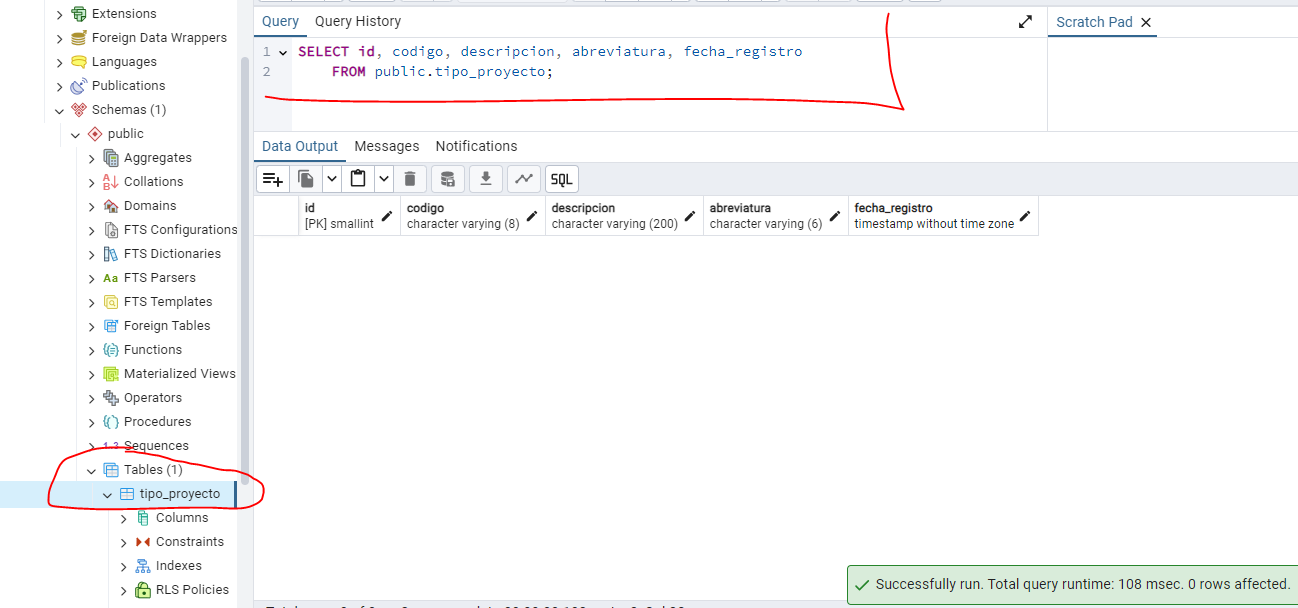
**MIEMBROS DEL EQUIPO:**

* Nicolas Cataño Durango
* [ANDRES FELIPE GUTIERREZ MARTINEZ](mailto:andres.gutierrez409@pascualbravo.edu.co)
* [JULIAN RAMIREZ MOLINA](mailto:julian.ramirez965@pascualbravo.edu.co)
* [EMMANUEL ESPINAL RAMIREZ](mailto:emmanuel.espinal731@pascualbravo.edu.co)
* [KEVIN VILLADA JARAMILLO](mailto:kevin.villada326@pascualbravo.edu.co)

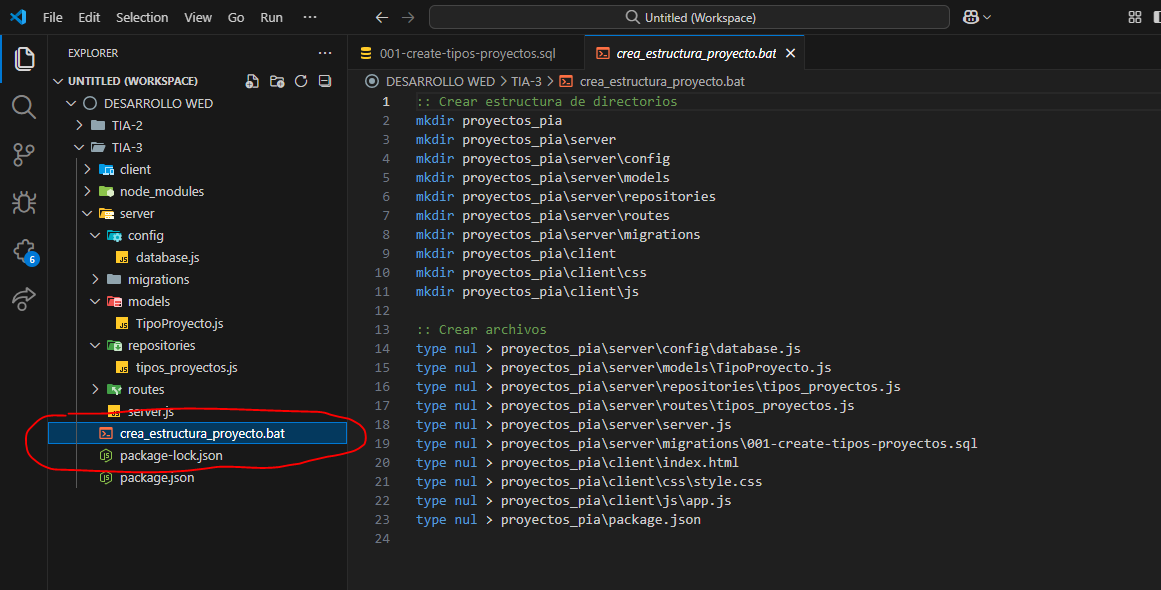
**Caso de Estudio**

**1.- Base de Datos “proyectos” y Tabla de Tipos de Proyecto**

|  | **Tabla** | **tipo\_proyecto** |  | | | |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Descripción** | Tipos de proyecto de Aula que desarrollan los estudiantes durante el semestre | | | | |
| # | **Campo** | **Descripción** | **Tipo Dato** | **Tamaño** | **Restricciones** | **Tablas**  **Relacionada** |
| 1 | **id** | Id de identificación | smallserial |  | Primary Key (PK) |  |
| 2 | **codigo** | Código tipo proyecto | varchar | 8 | Unique Key (UK) |  |
| 3 | **descripcion** | Descripción del tipo de proyecto | varchar |  | Not Null |  |
| 4 | **abreviatura** | Nombre abreviado | varchar | 6 | Not Null |  |
| 5 | **fecha\_registro** | Fecha/hora de inserción |  |  |  |  |

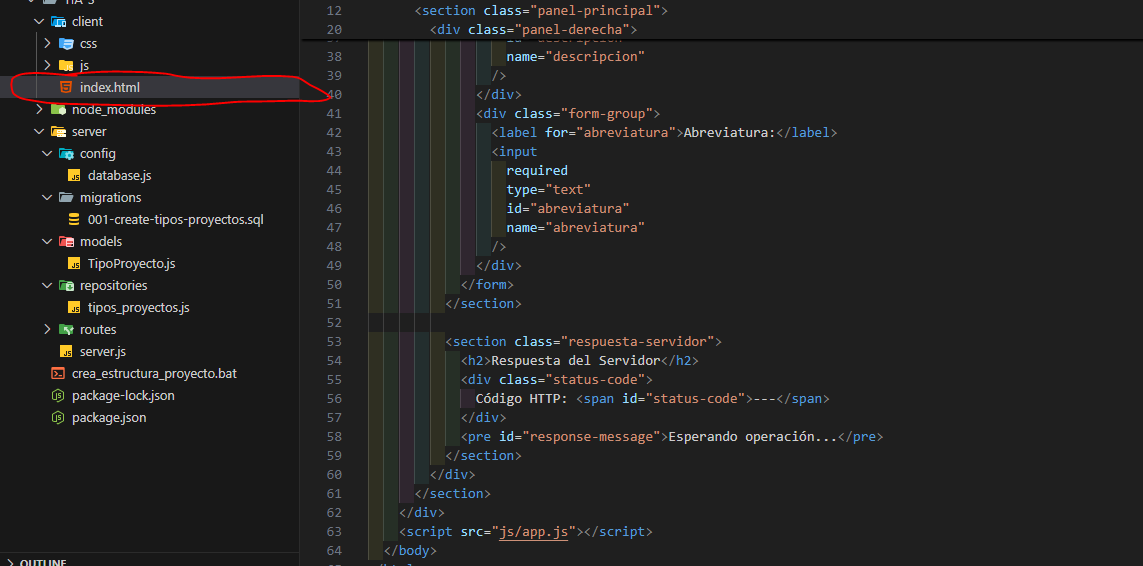
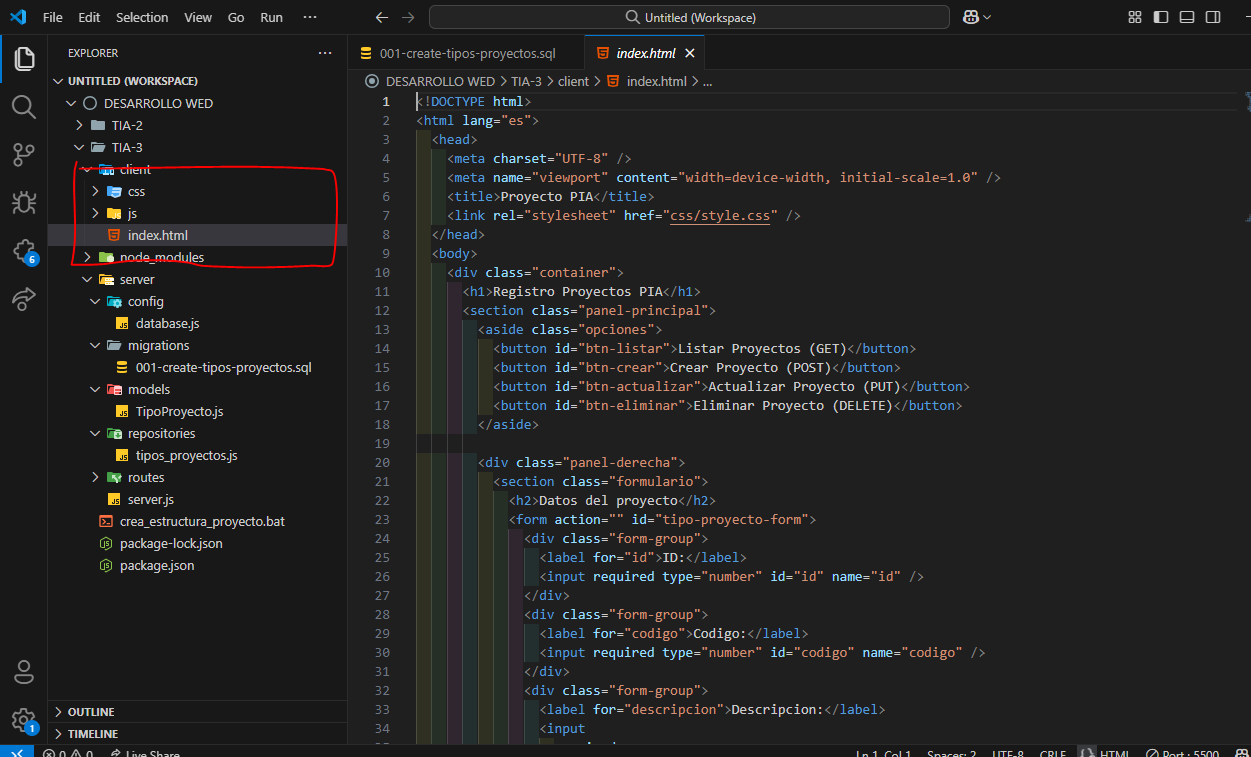
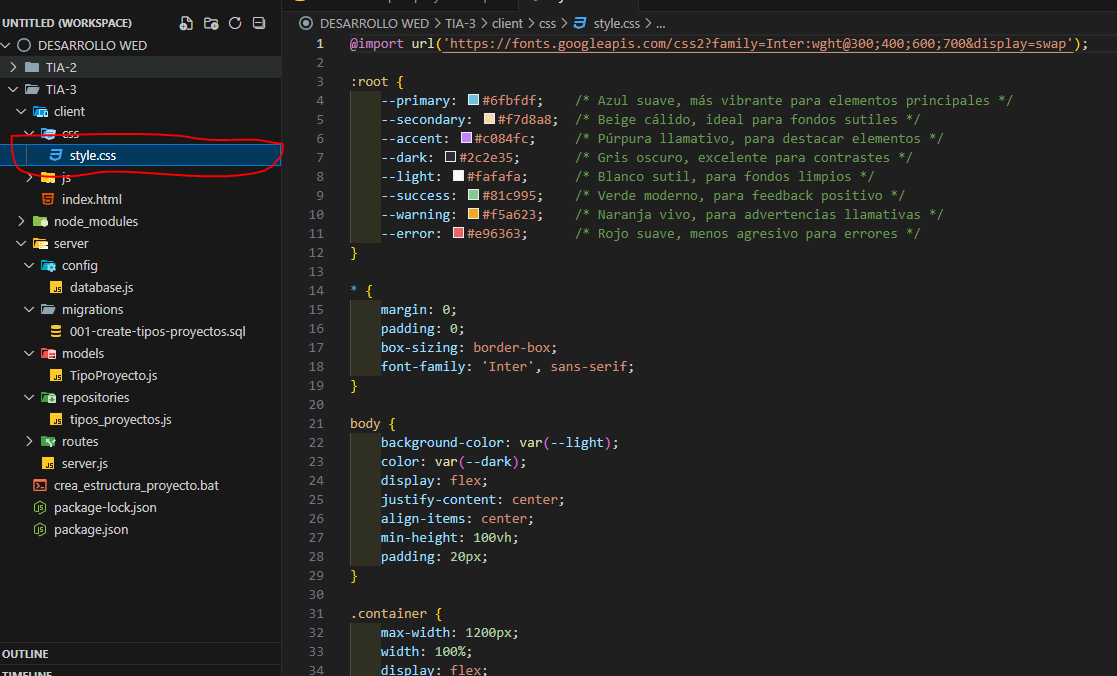
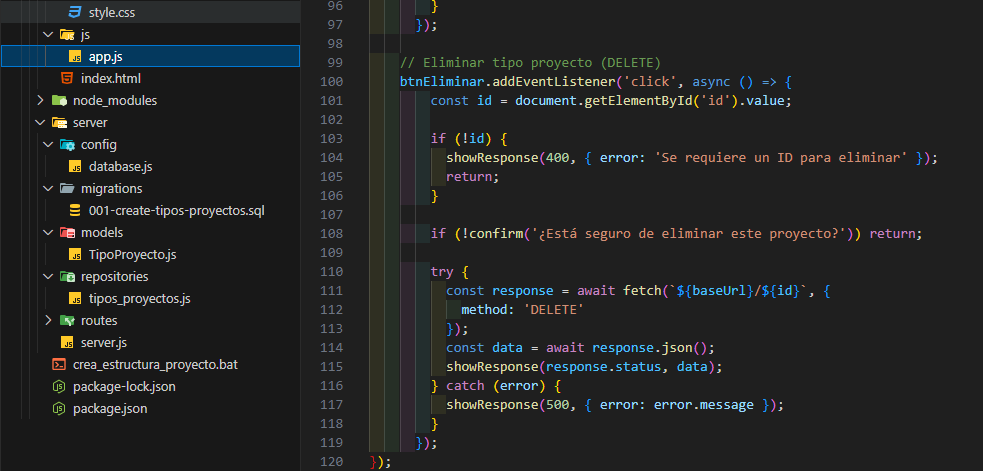
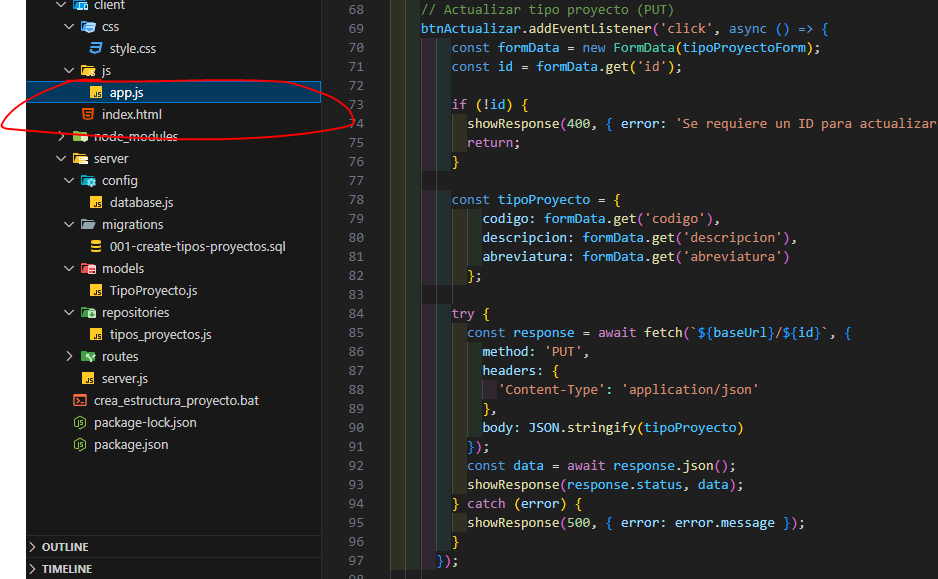
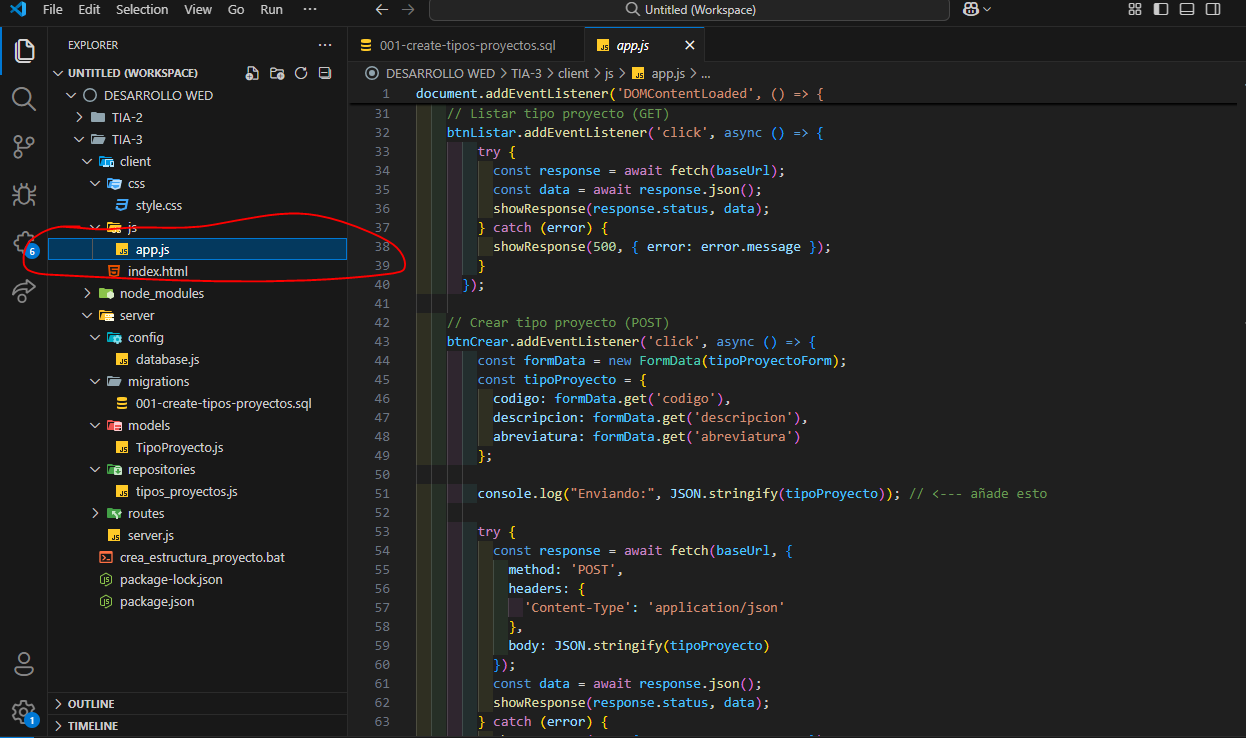
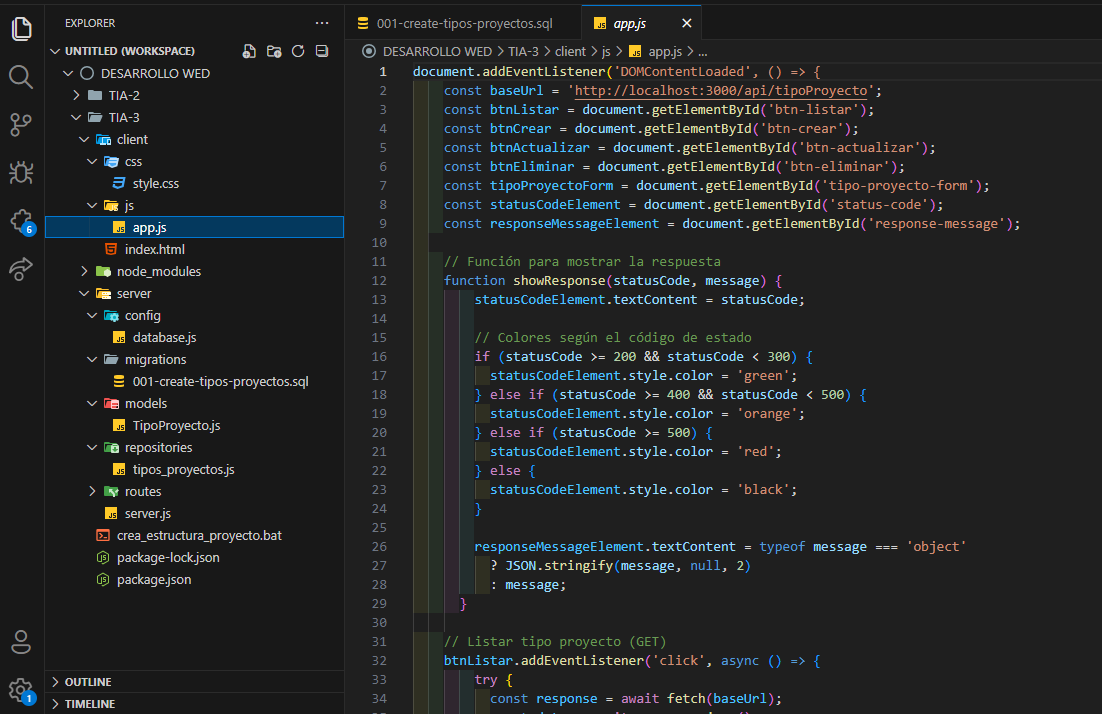
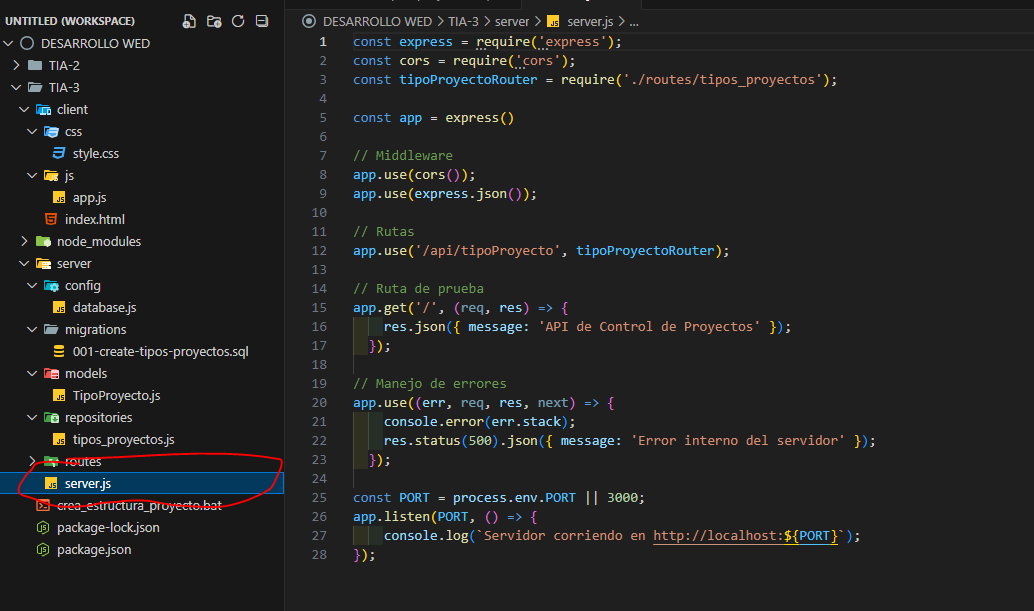
* **
* **
* **

**2.- Estructura del proyecto FrontEnd/BackEnd**

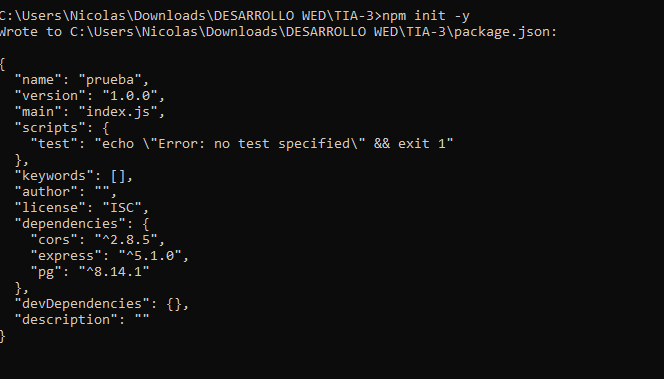
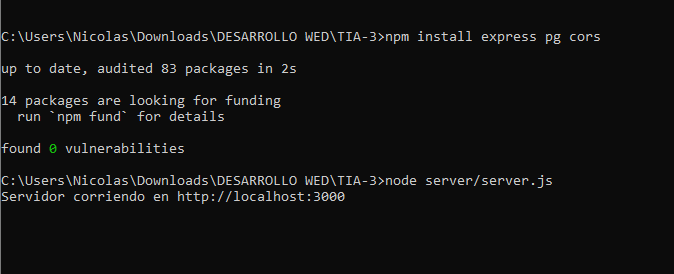
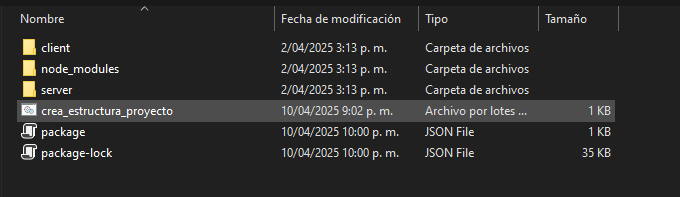
* **
* Este archivo es útil para **ahorrar tiempo y evitar errores humanos** al crear manualmente cada carpeta y archivo. Solo se necesita ejecutar este .bat y la estructura del proyecto queda lista en segundos.

**3.- Código fuente del Proyecto**

**Es una página HTML que sirve como interfaz web para gestionar proyectos PIA. Su función es permitir al usuario registrar, listar, actualizar y eliminar proyectos de forma interactiva. interfaz visual sencilla para probar y manejar un API REST relacionada con proyectos  
  
Este CSS hace que tu página se vea profesional, moderna y sea fácil de usar, gracias a una paleta de colores amigable, diseño limpio y una buena organización de los elementos.  
  
El código JavaScript de forma sencilla. Este archivo (app.js) es el que da vida a tu aplicación, conectando los botones con el backend y mostrando respuestas del servidor. Es como el cerebro del sistema.  
  
Escucha eventos de los botones y hace peticiones HTTP (GET, POST, PUT, DELETE) a un servidor usando fetch(). Luego muestra el resultado en pantalla.  
  
Este archivo permite que tu frontend se comunique con el servidor para crear, leer, actualizar y eliminar proyectos. Es como un mensajero que lleva los datos del formulario y te muestra lo que responde el backend.  
  
Es el servidor backend que recibe las solicitudes desde tu frontend (los botones que creaste en HTML/JS) y las dirige al archivo de rutas que maneja los proyectos.**

* **
* **
* **
* **

**4.- Creación del proyecto Node e instalación de las dependencias iniciales**

* **
* **
* **

Este comando inicializa un nuevo proyecto Node.js dentro de la carpeta proyecto\_pia. Crea automáticamente un archivo package.json con valores por defecto, el cual gestiona la información del proyecto y las dependencias que se van a usar. **npm install express pg cors)**

Este comando instala tres librerías esenciales:

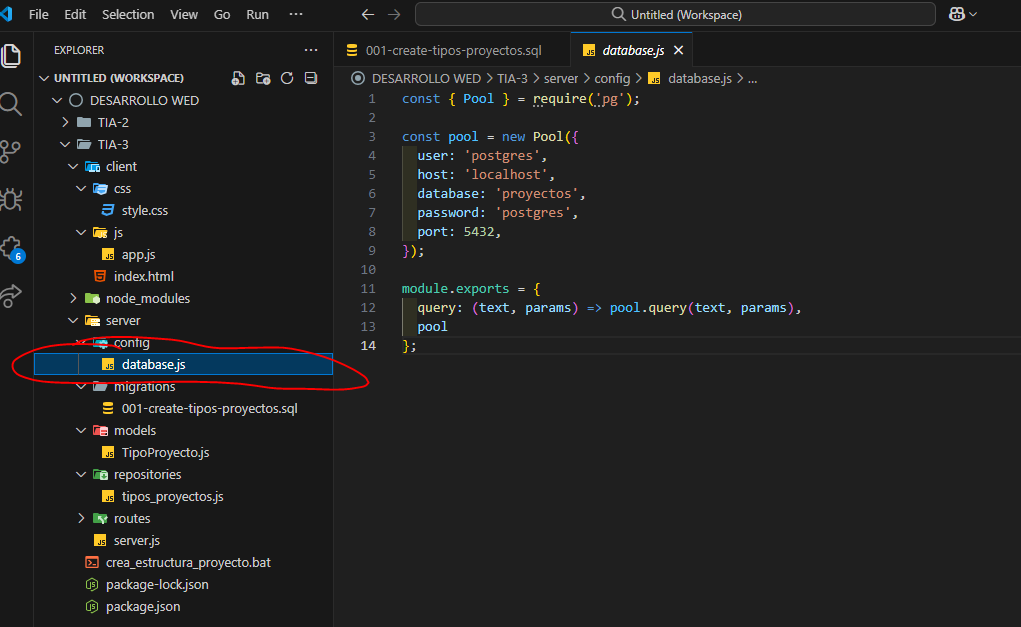
* express: framework minimalista para crear servidores web en Node.js.
* pg: módulo que permite conectarse y trabajar con bases de datos PostgreSQL.
* cors: habilita el intercambio de recursos entre distintos orígenes, permitiendo que el frontend y el backend se comuniquen sin errores de seguridad.

Estas dependencias quedan registradas en el archivo package.json, y el código fuente de las mismas se descarga en la carpeta node\_modules.Después de ejecutar los comandos anteriores, el proyecto contiene:

* package.json: configuración del proyecto.
* package-lock.json: versión exacta de cada dependencia instalada.
* node\_modules/: carpeta donde están todas las librerías que el proyecto necesita.
* Otros archivos del sistema que se generen (como .gitignore, si lo usas más adelante).

Con estas tres etapas completas, ya tienes el ambiente listo para empezar a programar tu API con Express y PostgreSQL

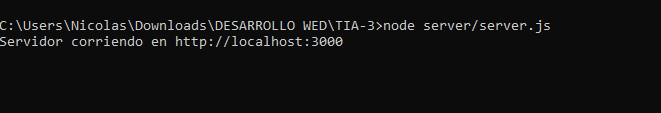
**5.- Configuración de la base de datos**

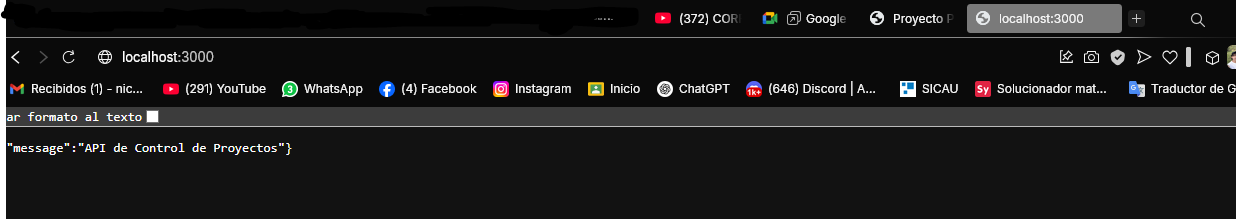
* **

Este archivo es fundamental porque:

* Permite que tu aplicación se comunique con la base de datos.
* Centraliza la configuración de conexión, lo cual facilita el mantenimiento y seguridad.
* Es el puente entre tus rutas de Express (donde están los métodos GET, POST, PUT, DELETE) y los datos reales almacenados en PostgreSQL.

**6.- Arranque del Servidor Node**

* **



Estás iniciando el servidor backend que has construido con Express. En ese momento:

* El servidor se pone a la escucha en el puerto 3000 (o el que definas en PORT).
* Carga los middlewares configurados (como cors y express.json()).
* Activa las rutas que creaste, como /api/tipoProyecto, permitiendo que el frontend pueda hacer peticiones GET, POST, PUT y DELETE.

**Estás iniciando el servidor backend que has construido con Express. En ese momento:**

* **El servidor se pone a la escucha en el puerto 3000 (o el que definas en PORT).**
* **Carga los middlewares configurados (como cors y express.json()).**
* **Activa las rutas que creaste, como /api/tipoProyecto, permitiendo que el frontend pueda hacer peticiones GET, POST, PUT y DELETE.**

**7.- Arrancar la página Web**

* **

Al abrir el archivo index.html desde la carpeta del cliente en el navegador, se carga la interfaz gráfica del proyecto.

En esta página se muestra un formulario que permite interactuar con el backend. El usuario puede:

* **Listar proyectos** existentes (GET).
* **Crear nuevos proyectos** (POST).
* **Actualizar proyectos** (PUT).
* **Eliminar proyectos** (DELETE).

Cada una de estas acciones genera una solicitud al servidor Node.js, que luego se comunica con la base de datos PostgreSQL para procesar la información

**8.- Inserción de registros (POST/INSERT)**



* **

### Registro del tipo de proyecto con código 1000 - "Proyecto de Aula"

En este paso se utilizó el formulario de la página web para registrar un nuevo tipo de proyecto con código **1000** y descripción **"Proyecto de Aula"**.  
 Al hacer clic en el botón **"Crear Proyecto (POST)"**, el sistema envió una solicitud HTTP POST al servidor.

El backend procesó correctamente la petición y devolvió un **código de estado 201 (Created)**, lo que indica que el proyecto fue registrado exitosamente en la base de datos.

### Intento de registrar otro proyecto con código 1000 - "Proyecto Industrial de Aula"

Aquí se intentó registrar otro tipo de proyecto utilizando el mismo **código 1000**, pero con una descripción diferente: **"Proyecto Industrial de Aula"**.

Al ser un código duplicado, el servidor respondió con un **código de estado 400 (Bad Request)** o **409 (Conflict)** (dependiendo de cómo tengas validado el modelo o el SQL), indicando que ya existe un proyecto con ese código.

Este comportamiento es correcto y demuestra que el sistema valida los datos para evitar registros duplicados.

**9.- Modificación de registros (PUT)/UPDATE)**

* **
* **

### ✅ Actualización correcta de un tipo de proyecto (PUT)

En esta prueba, se ingresó el **ID de un proyecto existente** y se modificó su información, por ejemplo, cambiando la descripción o la abreviatura.

Al presionar el botón **"Actualizar Proyecto (PUT)"**, se envió una solicitud HTTP PUT al servidor.  
 La respuesta del servidor fue un **código de estado 200 (OK)**, confirmando que la modificación fue realizada exitosamente.

Esto demuestra que la API está funcionando correctamente y puede actualizar registros en la base de datos.

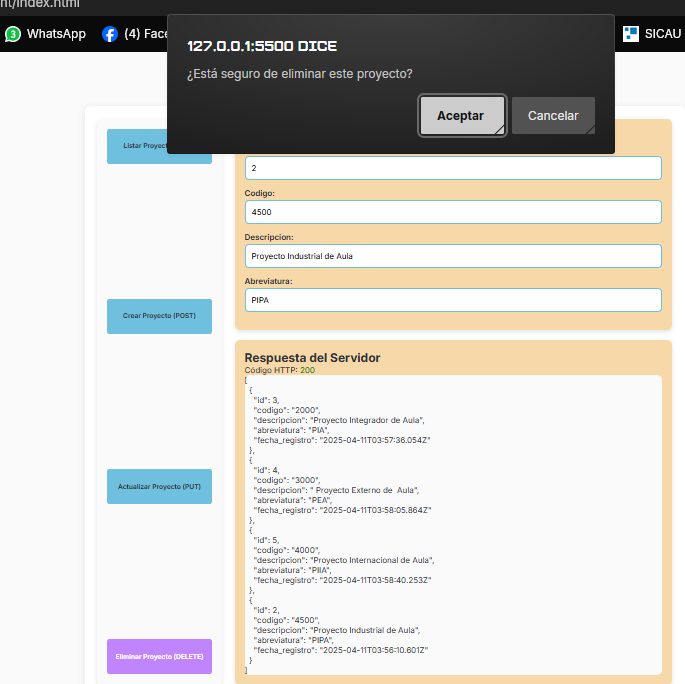
### ❌ Intento fallido de actualización de un tipo de proyecto

En este caso, se intentó modificar un proyecto utilizando un **ID que no existe** en la base de datos o se omitió el ID por completo.

Al ejecutar la acción, el servidor respondió con un **código de estado 400 (Bad Request)** o **404 (Not Found)**, indicando que no se encontró el proyecto que se quería actualizar o que la solicitud era inválida.

Este comportamiento es esperado y correcto, ya que asegura que solo se puedan modificar registros existentes y con datos completos.

**10.- Eliminación de un registro (DELETE/DELETE)**

* **
* **
* **

### ✅ Eliminación exitosa de un tipo de proyecto (DELETE)

En esta operación, se ingresa el **ID de un tipo de proyecto existente** en el formulario y se presiona el botón **"Eliminar Proyecto (DELETE)"**.

La aplicación envió una solicitud HTTP DELETE al servidor y recibió una **respuesta con código 200**, confirmando que el registro fue eliminado correctamente de la base de datos.

Esto indica que la funcionalidad de eliminación funciona correctamente y la API está respondiendo como se espera.

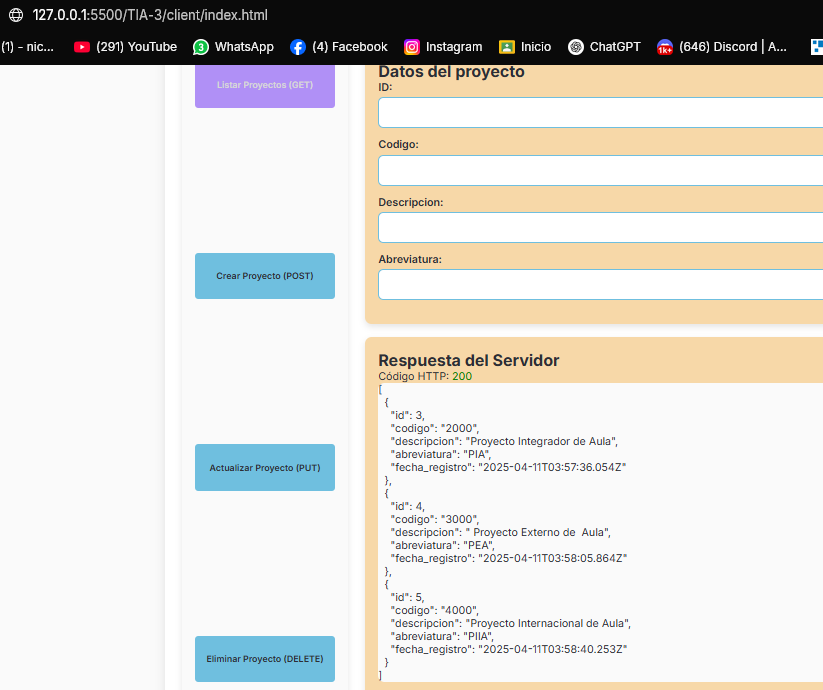
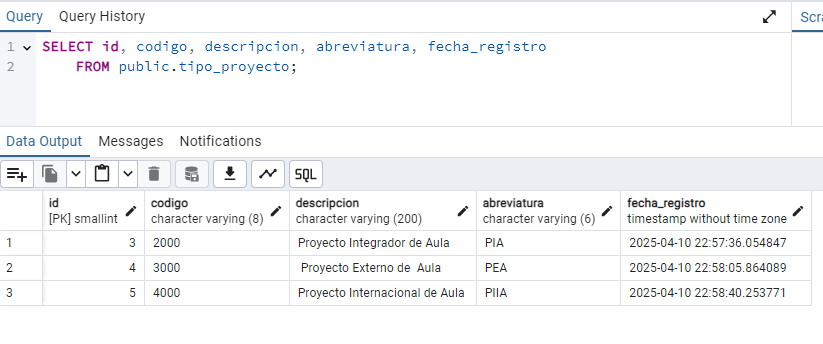
### ❌ Intento fallido de eliminar un tipo de proyecto

Aquí se probó intencionalmente con un **ID inexistente** o se dejó el campo ID vacío antes de presionar el botón de eliminar.

El sistema respondió con un **código de error 400 (Bad Request)** o **404 (Not Found)**, lo que indica que **no se pudo eliminar el registro porque no existía o faltaban datos necesarios**.

Este tipo de control es importante para mantener la integridad del sistema y evitar errores inesperados.

**11.- Consulta de registros (GET/SELECT)**

* **
* **

### ✅ Consulta exitosa desde la página web (HTTP GET)

En esta prueba, se hizo clic en el botón **“Listar Proyectos (GET)”** dentro de la interfaz web. Esto generó una **petición GET** al servidor, el cual respondió con un **código HTTP 200** y devolvió correctamente el listado de todos los tipos de proyecto almacenados en la base de datos.

El sistema mostró los datos en el área de **“Respuesta del Servidor”**, confirmando que la comunicación entre el frontend y la API está funcionando correctamente.

**12.- Conclusiones Individuales**.

### Nicolás Cataño Durango

* Durante el desarrollo del Proyecto Integrador de Aula (PIA), asumí el rol de líder del equipo, con la responsabilidad principal de coordinar y asegurar la integración completa del sistema. Me enfoqué particularmente en el diseño, conexión y gestión de la base de datos en PostgreSQL, así como en la configuración del servidor para que todo el sistema pudiera comunicarse correctamente a través de las rutas HTTP.
* Una de las principales lecciones aprendidas en este proceso fue la importancia de la planificación y la buena comunicación en equipo. Liderar los roles significó mantener a todos informados, revisar los avances, resolver dudas técnicas y ayudar a integrar los diferentes módulos. Esta experiencia fortaleció mis habilidades de liderazgo y mi capacidad de resolver problemas técnicos de manera eficiente.
* En cuanto a los conocimientos técnicos, profundicé en el uso de PostgreSQL y la estructura de las bases de datos relacionales. Comprendí mejor cómo construir sentencias SQL eficientes, cómo manejar las conexiones entre el servidor y la base de datos utilizando Express y cómo organizar las rutas para lograr una arquitectura clara y funcional. Todo esto está directamente relacionado con el propósito del PIA, que busca aplicar conocimientos técnicos reales en un contexto práctico y funcional.
* Además, pude reforzar mis competencias en trabajo colaborativo, pensamiento lógico y toma de decisiones bajo presión. El ejercicio de garantizar que todo funcionara correctamente me exigió revisar código, realizar pruebas de integración, corregir errores y asegurar que el sistema completo se comunicara adecuadamente, desde la interfaz hasta la base de datos.
* Esta experiencia me permitió comprender con mayor claridad el valor de cada rol en un proyecto y cómo se complementan las tareas individuales para lograr un producto funcional. Considero que esta tarea tiene un alto nivel de vinculación con mi futuro profesional, ya que en el ámbito del desarrollo de software es común enfrentarse a la necesidad de coordinar equipos, construir soluciones integrales y garantizar la estabilidad de los sistemas. Me siento más preparado para enfrentar ese tipo de desafíos.

### Andrés Felipe Gutiérrez Martínez

* Durante el desarrollo del PIA, mi función principal fue trabajar junto a mi compañero Julián en la creación de la interfaz gráfica del sistema, utilizando HTML, CSS y JavaScript. Esta tarea me permitió afianzar mis conocimientos sobre diseño web, estructuras semánticas y la lógica del comportamiento de las páginas mediante eventos y manipulaciones del DOM.
* El grado de participación que tuve fue constante a lo largo del proyecto. Desde el inicio estuve involucrado en la planeación de la interfaz, la distribución de las secciones y la implementación de los botones que interactúan con el backend. También ayudé a realizar pruebas visuales y funcionales para asegurarme de que los datos se capturaran correctamente desde los formularios.
* Entre los conocimientos adquiridos destaco el uso de fetch para realizar solicitudes HTTP desde JavaScript, lo que me permitió conectar el frontend con el servidor. Esto me ayudó a entender mejor la dinámica de comunicación cliente-servidor. Igualmente, mejoré mis habilidades en CSS para estilizar las respuestas que se mostraban al usuario, haciéndolas más comprensibles y visualmente agradables.
* Esta experiencia me dejó claro lo importante que es la presentación visual en un sistema de información. Una buena interfaz no solo mejora la experiencia del usuario, sino que también facilita la validación y gestión de datos. Me permitió trabajar en equipo, dividir tareas y aprender de mis compañeros sobre cómo funciona el sistema completo, no solo el frontend.
* Considero que esta tarea tiene una relación directa con mi futuro profesional, especialmente si deseo dedicarme al desarrollo frontend o fullstack. Aprendí cómo estructurar un proyecto, cómo integrar mis componentes con los del resto del equipo y cómo comunicarme eficazmente para resolver problemas técnicos.

### Julián Ramírez Molina

* Mi rol en el Proyecto Integrador de Aula fue el de desarrollar la parte visual del sistema junto a mi compañero Andrés, usando HTML, CSS y JavaScript. Desde el principio trabajamos en la creación de un formulario funcional que permitiera al usuario interactuar con el sistema y gestionar los tipos de proyecto mediante una interfaz clara y dinámica.
* Mi grado de participación fue activo, involucrándome en la estructuración del formulario, los botones de acciones (GET, POST, PUT, DELETE), la conexión con el backend a través de fetch, y el diseño estético para hacer la página amigable. También me encargué de validar algunos datos desde el cliente para facilitar la comunicación con el servidor.
* Gracias a esta experiencia, reforcé mi comprensión sobre cómo crear interfaces web efectivas y cómo conectarlas con servidores mediante peticiones HTTP. Aprendí a manejar respuestas del servidor y a mostrarlas correctamente en la interfaz. Además, adquirí experiencia en organización de código, depuración de errores de JavaScript y trabajo en equipo.
* Esta tarea se relaciona fuertemente con el objetivo del PIA, ya que me permitió aplicar mis conocimientos en un contexto real de desarrollo. A través de la colaboración, pude ver cómo mi trabajo se conectaba con el de mis compañeros, creando un sistema completo y funcional.
* En mi futuro profesional, estas habilidades serán fundamentales si deseo trabajar en el área de desarrollo web. La experiencia práctica que obtuve en este proyecto me dio más confianza para enfrentar nuevos desafíos, colaborar con equipos y pensar de forma lógica para resolver problemas técnicos.

### Emmanuel Espinal Ramírez

* En el desarrollo del PIA, mi función fue trabajar en el enrutamiento HTTP y en la estructura cliente-servidor, lo cual me permitió adquirir una visión más técnica sobre cómo se comunican los distintos componentes de un sistema web. Junto a mi compañero Kevin, nos encargamos de las rutas en Express, gestionando las peticiones GET, POST, PUT y DELETE.
* Mi grado de participación fue constante, aportando en la configuración de las rutas, la validación de los datos en el backend y asegurando que cada petición tuviera una respuesta adecuada. También ayudé a establecer la estructura básica del servidor en Node.js, definir los endpoints y conectar con la lógica del controlador.
* Esta experiencia me ayudó a profundizar en el concepto de API REST, la estructura de los proyectos en Node.js y la importancia de mantener un código limpio y organizado. Además, aprendí a trabajar con controladores y middlewares, elementos clave en cualquier aplicación backend.
* El PIA me permitió desarrollar competencias como el trabajo en equipo, la colaboración técnica y la búsqueda de soluciones eficaces. Esta tarea está altamente relacionada con mi futuro profesional, ya que muchas empresas requieren desarrolladores capaces de estructurar servidores eficientes y seguros.

### Kevin Villada Jaramillo

* Mi participación en el PIA se centró en el diseño del cliente-servidor y en el manejo del enrutamiento HTTP utilizando Express. Trabajé de la mano con Emmanuel para establecer una comunicación eficiente entre el frontend y el backend, estructurando las rutas y asegurándonos de que cada funcionalidad tuviera su correspondiente lógica implementada.
* Estuve involucrado en la configuración del servidor, el manejo de las respuestas HTTP y la conexión con la base de datos a través del archivo de configuración. Gracias a este trabajo, fortalecí mis conocimientos sobre el ciclo de vida de una solicitud web y el uso de herramientas modernas como Express.
* Esta experiencia fue muy enriquecedora. Aprendí cómo trabajar de manera colaborativa, cómo dividir responsabilidades y cómo integrar los diferentes módulos de un proyecto. Me permitió mejorar mi lógica de programación, entender la estructura de una API y profundizar en el desarrollo backend.
* Relaciono este trabajo directamente con mi futuro profesional, ya que me gustaría trabajar como desarrollador backend. Este proyecto me preparó para enfrentar retos reales y me dio la confianza para seguir avanzando en mi formación técnica.

**13.- Calidad del Informe de la Tarea. Rúbrica Informe**

| **#** | **Características del Informe** | **Puntos** | **Calificación** |
| --- | --- | --- | --- |
| 1 | * Presenta informe en la plantilla suministrada de forma correcta * Coloca el informe en la raíz de la carpeta TIA3 en el repositorio GIT * Se presentan todos los miembros del equipo con su nombre * Cubre la totalidad de los ítems de la tarea | **20** |  |
| 2 | * Escribe el nombre del video correctamente (colocando la letra del equipo en la “X”): ***20251-et0179-TIA3-equipo-x-informe.docx*** | **5** |  |
| 3 | * Presenta cuadros centrados, sin “dividirlos” entre saltos de página, sin distorsiones, letra ni tan grande ni tan pequeñita, con colores agradables. * Presenta figuras (pantallazos) centradas y de buena calidad | **50** |  |
| 4 | * Calidad general (tipo de letra, redacción, organización, figuras) * Redacta el informe con buena ortografía, gramática y expresión de ideas | **25** |  |
|  | **Total** | **100** |  |

**14.- Video de Sustentación**. **Rúbrica Video de Sustentación**

| **#** | **Características del Video de Sustentación** | **Puntos** | **Calificación** |
| --- | --- | --- | --- |
| **1** | * Se presentan todos de manera individual mostrando su cara, diciendo su nombre y describiendo la actividad que van a presentar * Se cubre la totalidad de los ítems de la tarea | **10** |  |
| **3** | Muestra de código en ejecución en PostgreSQL. Script SQL (pgAdmin4) | **5** |  |
| **3** | Muestra de código en ejecución del lado del cliente | **25** |  |
| **4** | Muestra de código en ejecución del lado del servidor | **25** |  |
| **5** | Describe la estructura del proyecto Node FrontEnd/BackEnd | **10** |  |
| **6** | Presenta breve conclusión individual sobre la tarea | **10** |  |
| **7** | Presenta video con calidad tanto de sonido como de visualización. Cada participante muestra su rostro claramente, se escucha bien el audio y se ve lo que presenta. | **10** |  |
| **8** | * Presenta enlace de acceso al video en el informe y en el repositorio * Escribe el nombre del video correctamente (colocando la letra del equipo en la “X”) ***20251-et0179-TIA3-equipo-X-video*** | **5** |  |
|  | Total | **100** |  |
|  | *NOTA: El estudiante que no aparezca, no tendrá evaluación de la tarea. ES OBLIGATORIA LA PARTICIPACIÓN EN EL VIDEO. Si por razones de “fuerza mayor” no puede aparecer en el video en conjunto, presentará un video individual exponiendo parte del trabajo.* | **0** | **0** |

**15.- Repositorio GIT. Rúbrica Repositorio GIT**

| **#** | **Características del Video de Sustentación** | **Puntos** | **Calificación** |
| --- | --- | --- | --- |
| **1** | Crea y presenta un repositorio con un nombre que identifique fácilmente el equipo de estudiantes | **20** |  |
| **2** | Presenta breve descripción del contenido del repositorio (Proyectos PA/PIA) | **5** |  |
| **3** | Se presentan los miembros del equipo en la primera página del repositorio | **5** |  |
| **4** | Repositorio organizado por carpetas de Tareas de esta manera  Tarea-1 o TIA-1  Tarea-2 o TIA-2  Tarea-3 o TIA-3  Tarea-4 o TIA-4  Tarea-5 o TIA-5  Tarea-5 o TIA-5 | **10** |  |
| **5** | Tarea 1 (TIA1) subida al repositorio | **10** |  |
| **6** | Tarea 2 (TIA2) subida al repositorio | **10** |  |
| **7** | Tarea 3 (TIA3) subida al repositorio (esta tarea) | **20** |  |
| **8** | Repositorio permite el acceso al público, el docente y todos los estudiantes  (modo lectura) | **5** |  |
| **9** | Cada tarea presenta un enlace al video de sustentación. | **5** |  |
| **10** | La Tarea-3 debe estar organizada en subcarpetas y archivos de esta manera:  Archivos en la raíz de la carpeta Tarea-4   * Informe * Enlace video * Subcarpetas “proyecto\_pia” dentro de la Tarea-4   + Cliente   + Servidor | **10** |  |
|  | **Total** | **100** |  |

**Rúbrica de la Tarea: Criterios de Evaluación**

| **ítem** | **Criterio** | | | **Peso** | **Cal** |
| --- | --- | --- | --- | --- | --- |
| **1** | **Base de Datos “proyectos” y Tabla de Tipos de Proyecto** | | | **20** |  |
| **2** | **Estructura del proyecto FrontEnd/BackEnd** | | | **30** |  |
| **3** | **Código fuente del Proyecto. Cambiar estilo Página Web** | | | **200** |  |
| **4** | **Creación del proyecto Node e instalación de las dependencias iniciales** | | | **10** |  |
| **5** | **Configuración de la base de datos** | | | **10** |  |
| **6** | **Arranque del Servidor Node** | | | **15** |  |
| **7** | **Arrancar la página Web** | | | **15** |  |
| **8** | **Mostrar ejecución operación Rest - Inserción de registros (POST)** | | | **75** |  |
| **9** | **Mostrar ejecución operación Rest- Modificación de registros (PUT)** | | | **75** |  |
| **10** | **Mostrar ejecución operación Rest - Eliminación de un registro (DELETE)** | | | **75** |  |
| **11** | **Mostrar ejecución operación Rest - Consulta de registros (GET)** | | | **75** |  |
| **12** | **Conclusiones Individuales** | | | **50** |  |
| **13** | **Calidad del Informe** | | | **100** |  |
| **14** | **Video de sustentación** | | | **200** |  |
| **15** | **Repositorio** | | | **50** |  |
|  | **NOTA** |  | **TOTAL** | **1000** |  |