**Tarea - TIA-03**

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

**EQUIPO: F**

**MIEMBROS DEL EQUIPO:**

* Juan José Ramos Agudelo
* Ximena Zamudio Mesa
* John Jairo Cañaveral Gutiérrez
* Estiven Toro Henao

**Caso de Estudio**

El Caso de Estudio está relacionado con los Proyectos PA/PIA. Debe tomar en consideración todo el material que se le ha suministrado como el enunciado que se le entregó en la Tarea 2 (TIA-2), el formato de registro de proyecto, la información que socializó la Profesora Vesna Srdanovic en clases, la revisión de los diferentes diccionarios de datos de los otros grupos y toda la información relacionada con los proyectos que Ud. puede recolectar (puede consultar a otros docentes en relación al tema)

**Material de ayuda:**

* Puede utilizar el contenido del ejercicio del Taller Rest-HTTP del 28-03-2025 relacionado con piscinas y adecuarlo a los datos de esa tarea

**Criterios de desempeño / Indicadores de Resultado de Aprendizaje**

* Utilizar los códigos de respuesta HTTP/HTTPS para comprender el estado de las solicitudes y respuestas en una aplicación web.
* Aplicar el uso de APIs RESTful para la comunicación con un servidor backend y el consumo de dichas APIs para la obtención y manipulación de datos.
* Desarrollar habilidades de trabajo en equipo y comunicación efectiva en la resolución de ejercicios prácticos relacionados con tecnologías web.
* El pensamiento sistémico y complejo
* La visión global
* La multi e interdisciplinariedad
* La vinculación solidaria
* Las emociones y los comportamientos
* La autonomía

**Actividad:**

* Analizar el Caso de Estudio suministrado por el Profesor.
* Diseñar e implementar una página Web responsiva que establezca una interacción con un Servidor
* implementar llamadas APIs RESTful para la comunicación con un servidor Backend
* Implementar respuestas con códigos HTTP.
* Conectar y actualizar una base de datos con las cuatro (4) operaciones básicas de un CRUD: INSERT, UPDATE, DELETE, SELECT

**Propósito:**

* Comprender el contexto de un problema o necesidad real y presentar una propuesta de solución
* Comprender cómo funciona un sistema de interacción FrontEnd / BackEnd
* Profundizar los conocimientos de una arquitectura REST
* Practicar con un sistema de Solicitud/Respuesta con el uso de códigos HTTP
* Aprender a conectar y actualizar bases de datos con sistemas de información Web

**Modalidad de entrega:**

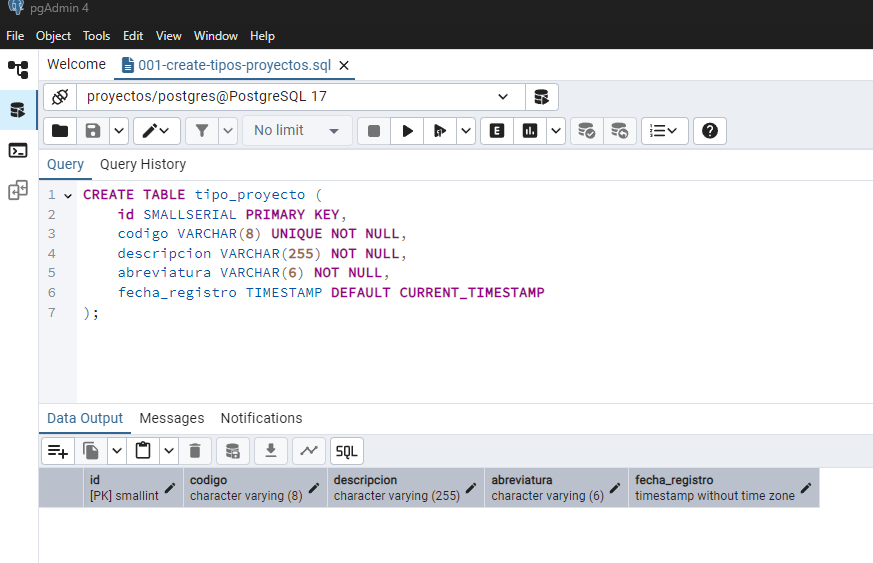
* Los resultados de la Tarea se deben subir al repositorio GIT de tareas de la asignatura en la carpeta Tarea-3 (TIA-3). Nota. Permite el acceso al público y al docente para acceder y ver el contenido.
* En el Classroom solamente colocarán UN SOLO LINK al repositorio. Lo colocará el líder del grupo

**Productos (entregables):**

* Plantilla de Informe con los resultados
* Proyecto Node con conexión y actualización de base de datos (carpetas y sub carpetas)
* Repositorio GIT con las Tareas: TIA-1, TIA-2 y TIA-3
* Equipo de miembros en la página principal del GIT
* Breve descripción del Proyecto PIA en el GIT

**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 | 255 | Not Null |  |
| 4 | **abreviatura** | Nombre abreviado | varchar | 6 | Not Null |  |
| 5 | **fecha\_registro** | Fecha/hora de inserción |  |  |  |  |

******

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

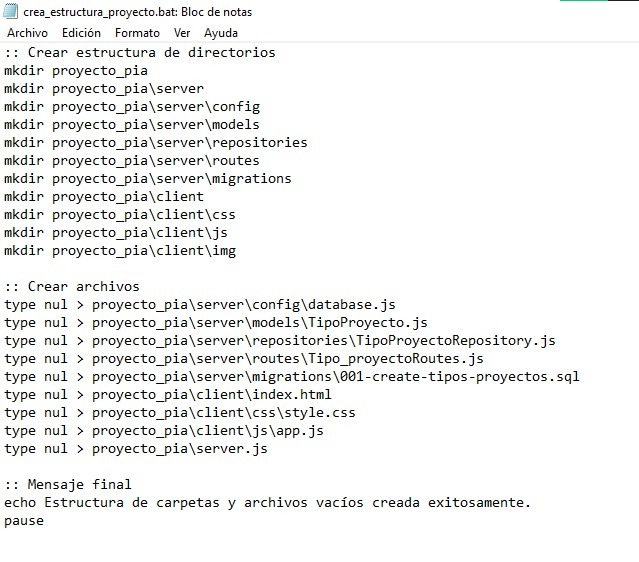
El archivo crea\_estructura\_proyecto.bat permite generar la estructura base del proyecto proyecto\_pia, correspondiente a la TIA-3.

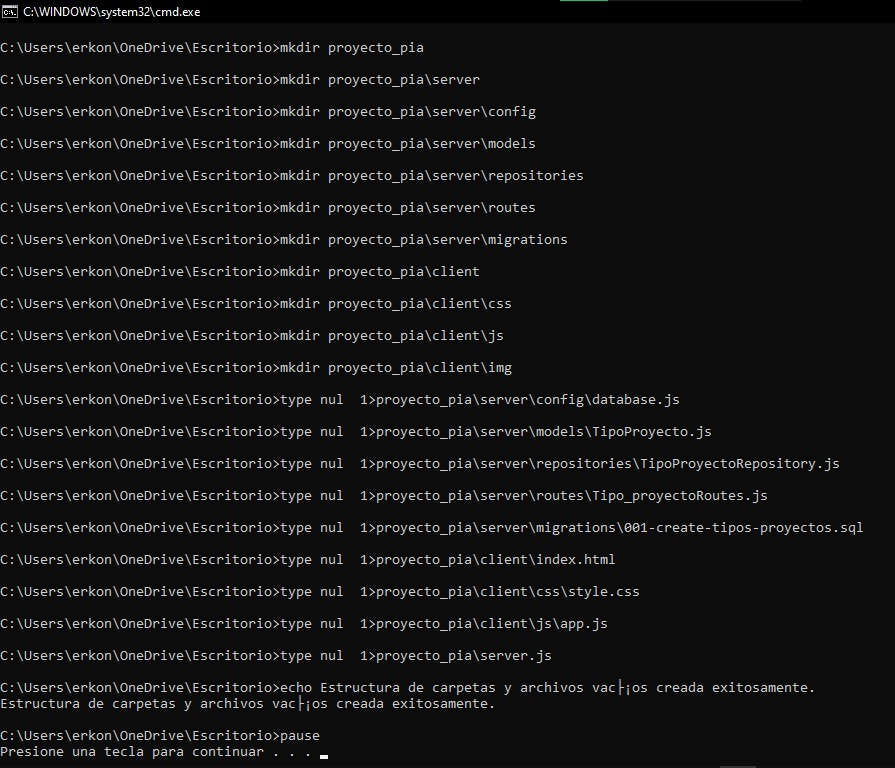
Cuando se ejecuta, este archivo:

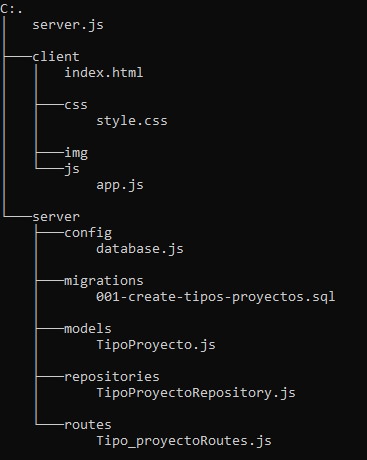
- Crea las carpetas necesarias tanto para el backend (server) como para el frontend (client)

- Dentro de cada carpeta, genera archivos vacíos con los nombres correspondientes al proyecto

- Al finalizar, muestra un mensaje en consola indicando que la estructura se ha creado exitosamente







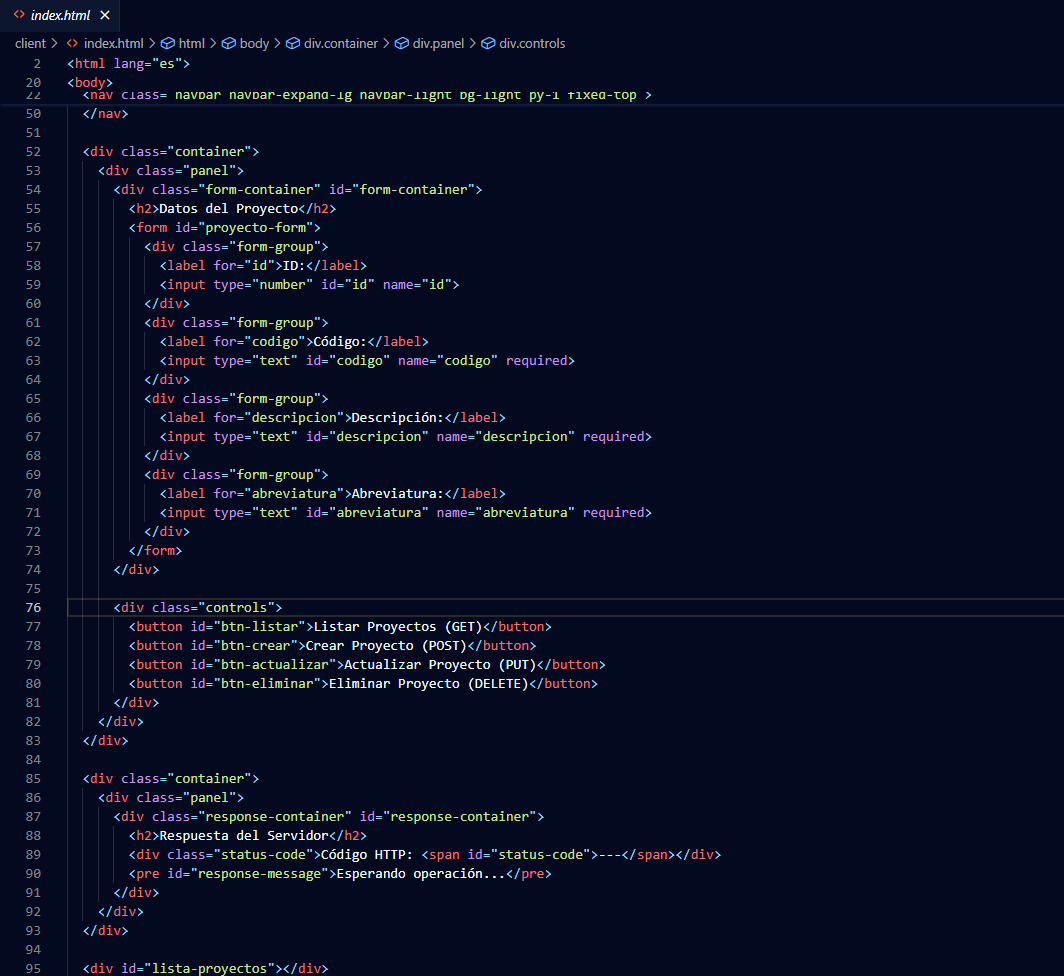
**3.- Código fuente del Proyecto**

**HTML**

Este código HTML crea una **página web responsiva** para el **control de proyectos**, usando **Bootstrap** para el diseño y estilo. Incluye:

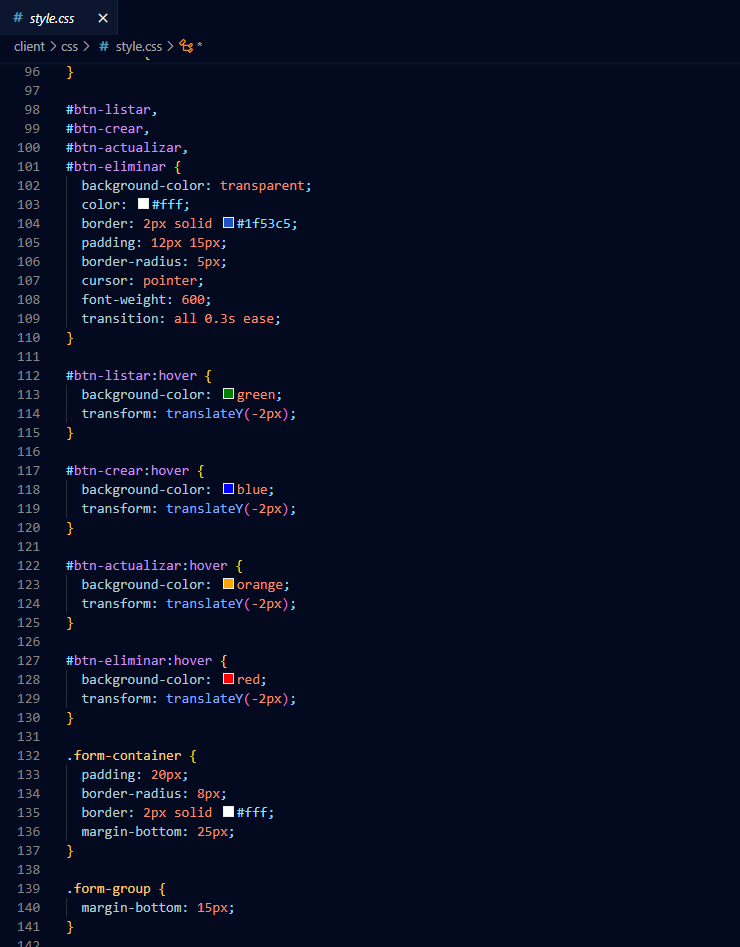
* Una **barra de navegación fija** con enlaces a secciones del sitio.
* Un **formulario** para ingresar datos de un proyecto (ID, código, descripción, abreviatura).
* **Botones** para ejecutar operaciones CRUD: listar, crear, actualizar y eliminar proyectos (GET, POST, PUT, DELETE).
* Un área para mostrar la **respuesta del servidor**.
* Un **footer** con información de contacto y enlaces a GitHub de los autores.

El comportamiento dinámico (peticiones al servidor) se gestiona con **JavaScript** externo (app.js).



**CSS**

Este CSS da estilo moderno y responsivo a la página de control de proyectos. Define colores con variables, aplica transiciones al navbar, botones CRUD con colores distintos al hacer hover, formularios con bordes y campos estilizados, y un footer limpio. Además, adapta el diseño a dispositivos móviles.



**APP.JS**

Este código JavaScript gestiona operaciones CRUD para **tipos de proyectos** con una API REST. Al cargar la página, se configuran los botones para listar, crear, actualizar y eliminar proyectos. Muestra respuestas con estado y mensaje, y adapta el diseño al hacer scroll. Además, mejora la experiencia móvil cerrando el menú al seleccionar una opción.



**SERVER.JS**

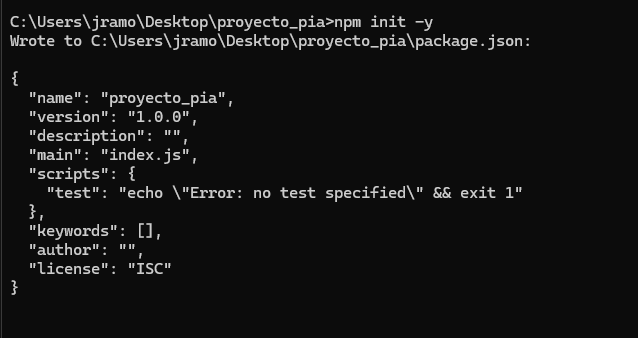
Este archivo configura un servidor con Express que:

* Habilita CORS y el uso de JSON.
* Sirve archivos estáticos desde la carpeta client.
* Define una ruta API en /api/Tipo\_proyectos usando un router externo.
* Muestra index.html al acceder a la raíz (/).
* Maneja errores del servidor.
* Inicia el servidor en el puerto 3000.

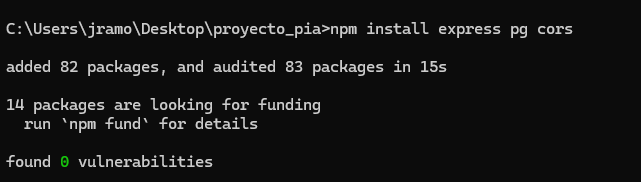


**4.- Creación del proyecto Node e instalación de las dependencias iniciales**

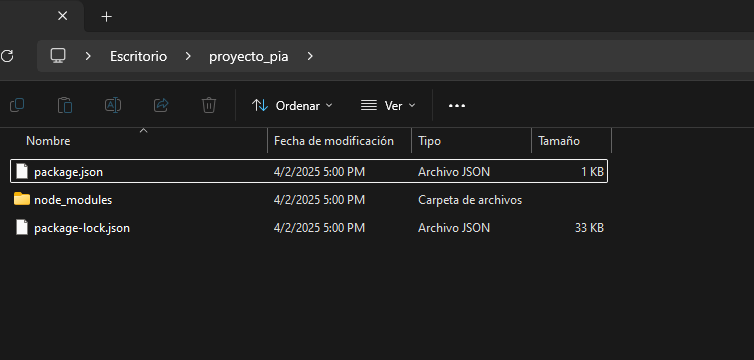
Este codigo inicializa un proyecto Node.js creando automáticamente el archivo package.json con valores por defecto.



este comando instala las dependencias express, pg y cors en el proyecto Node.js para mejorar rutas, conectar con PostgreSQL y permitir solicitudes entre dominios.



Muestra la estructura de un proyecto Node.js después de instalar dependencias. Contiene: package.json, package-lock.json, y la carpeta node\_modules con los módulos descargados.



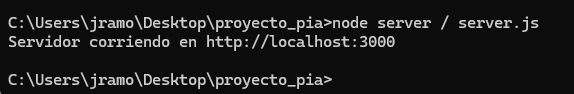
**5.- Configuración de la base de datos**

Este archivo configura la conexión a una base de datos PostgreSQL usando el modulo pg. Define un pool de conexiones con los datos de acceso y exporta una función para ejecutar consultas desde otras partes del proyecto.

****

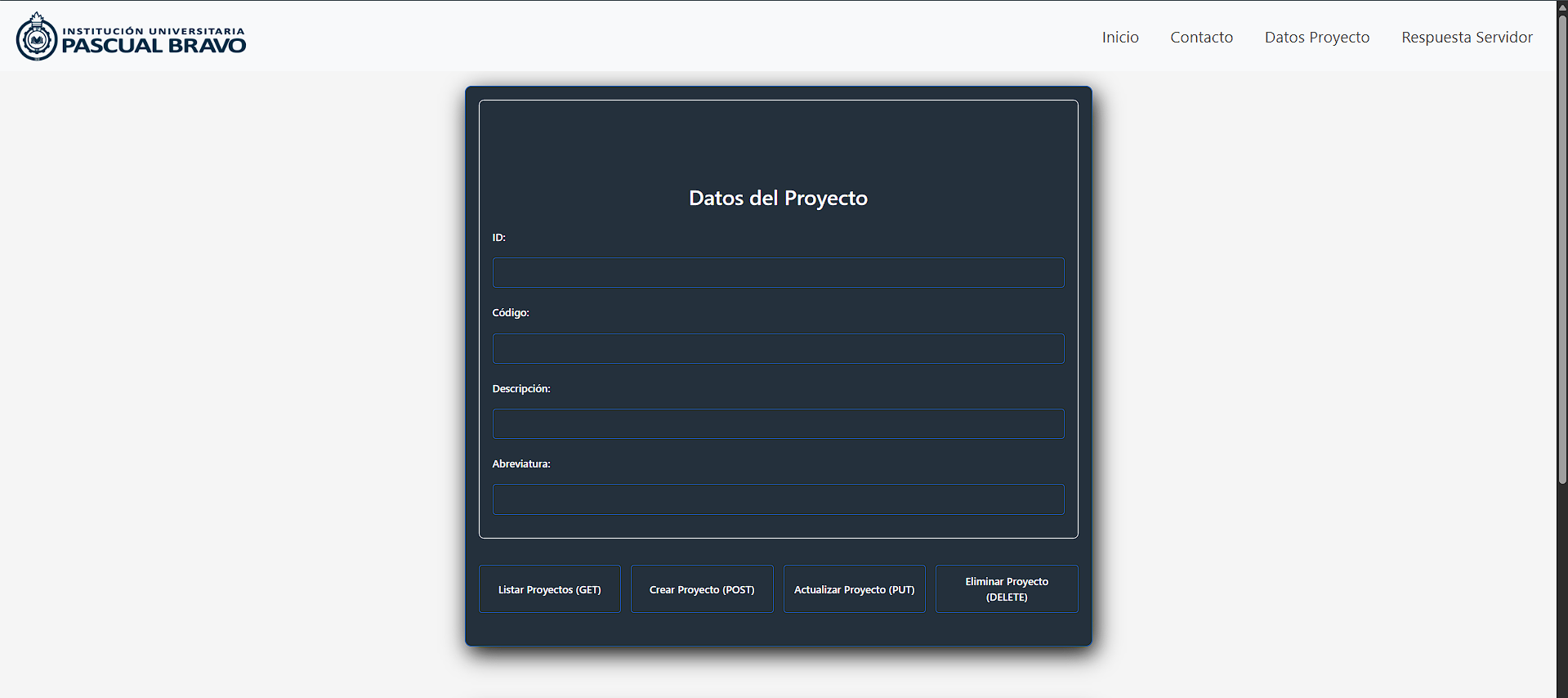
**6.- Arranque del Servidor Node**

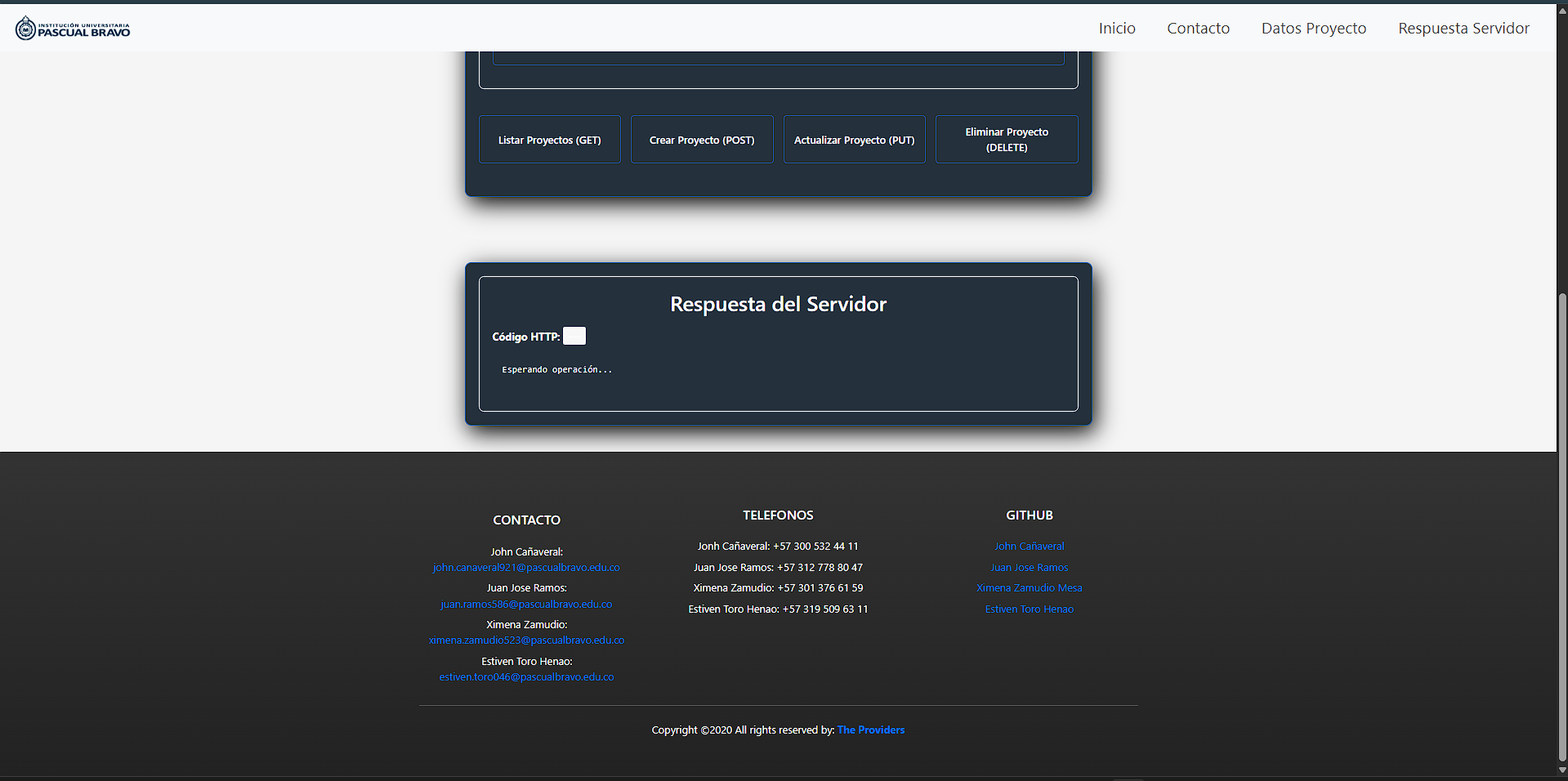
Este comando ejecuta el archivo server.js iniciando el servidor Node.js en http://localhost:3000 , listo para recibir peticiones.

****

**7.- Arrancar la página Web**

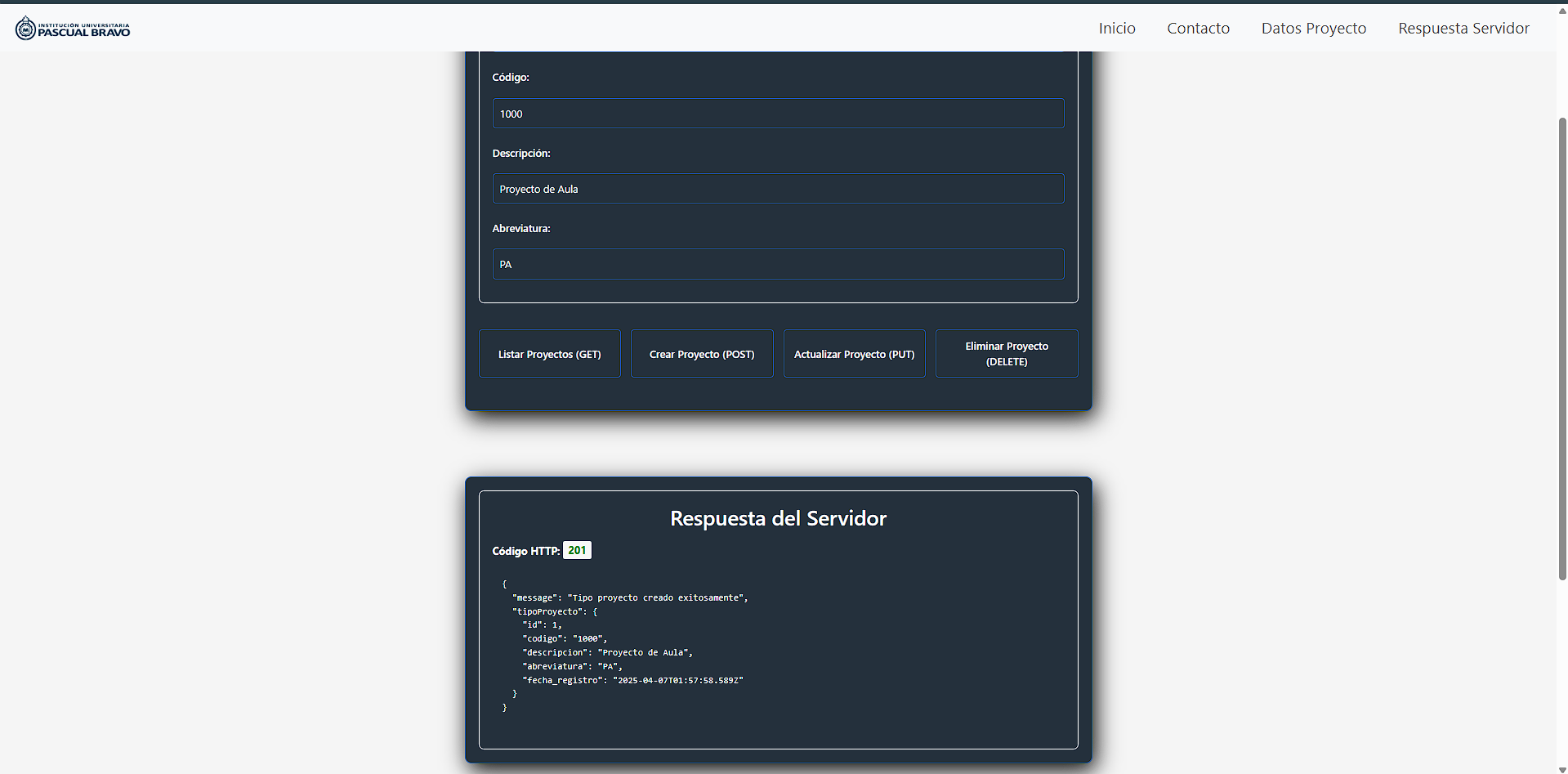
* **Formulario de Datos del Proyecto:**  
  Permite ingresar o editar un proyecto con campos como ID, Código, Descripción y Abreviatura.
* **Botones (CRUD):**
  + **Listar Proyectos (GET):** Muestra todos los proyectos.
  + **Crear Proyecto (POST):** Crea un nuevo proyecto.
  + **Actualizar Proyecto (PUT):** Modifica un proyecto existente.
  + **Eliminar Proyecto (DELETE):** Borra un proyecto.
* **Sección "Respuesta del Servidor":**  
  Muestra el código de respuesta HTTP (como 200, 404, etc.) y mensajes del backend.
* **Footer:**  
  Incluye contactos, teléfonos y enlaces a GitHub de los desarrolladores del proyecto.

****

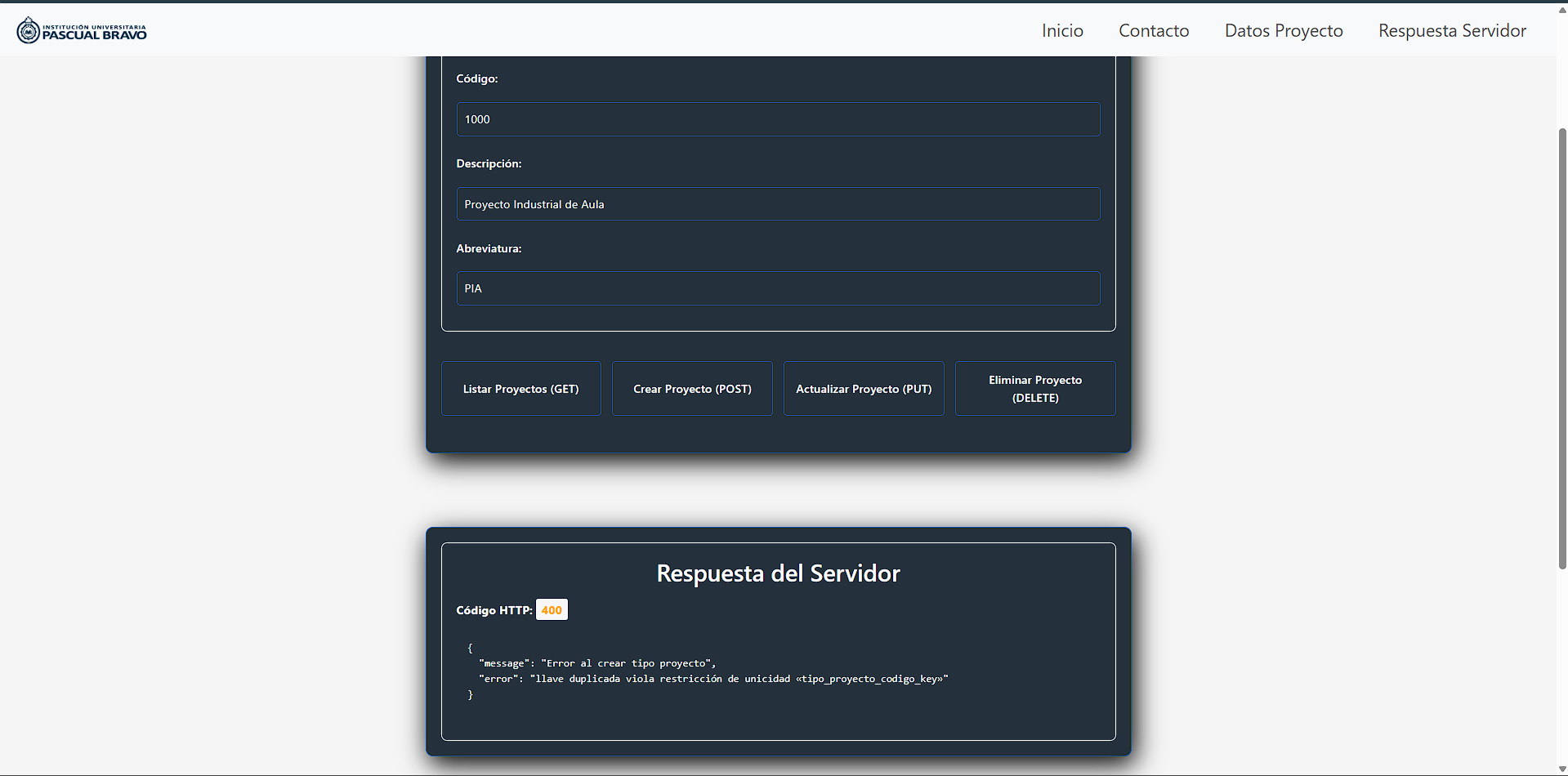
****

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

Se crea exitosamente un proyecto con el codigo: 1000 y con la descripcion: Proyecto de Aula y abreviatura: PA. El servidor responde con codigo http 201 y muestra los datos registrados, incluyendo la fecha de creación.

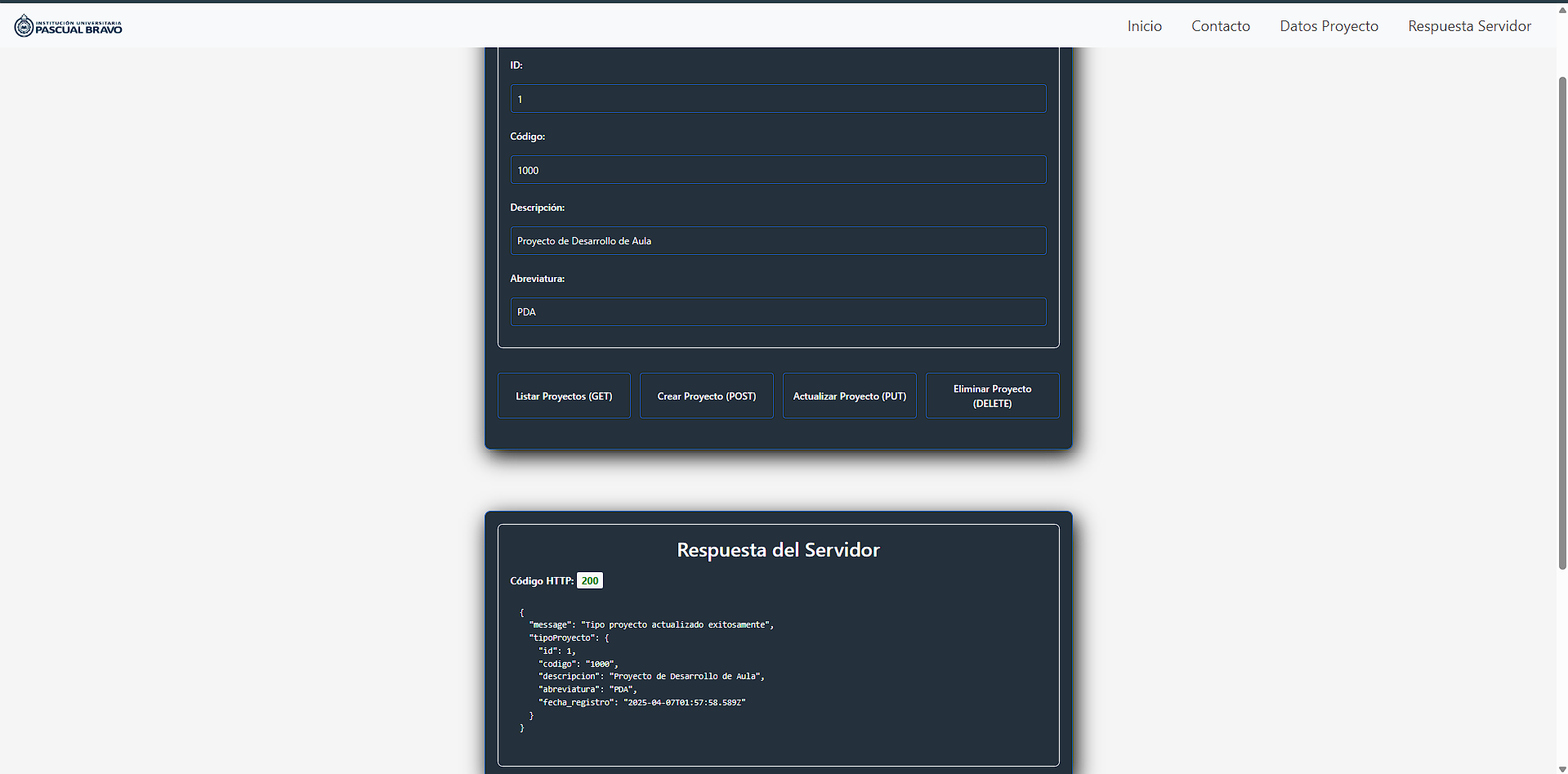


Se intenta crear un proyecto con el codigo: 1000 y con la descripcion: Proyecto Industrial de Aula y la abreviatura: PIA. Pero el servidor responde con error http 400 indicando que ya existe un proyecto con ese mismo codigo (llave duplicada).

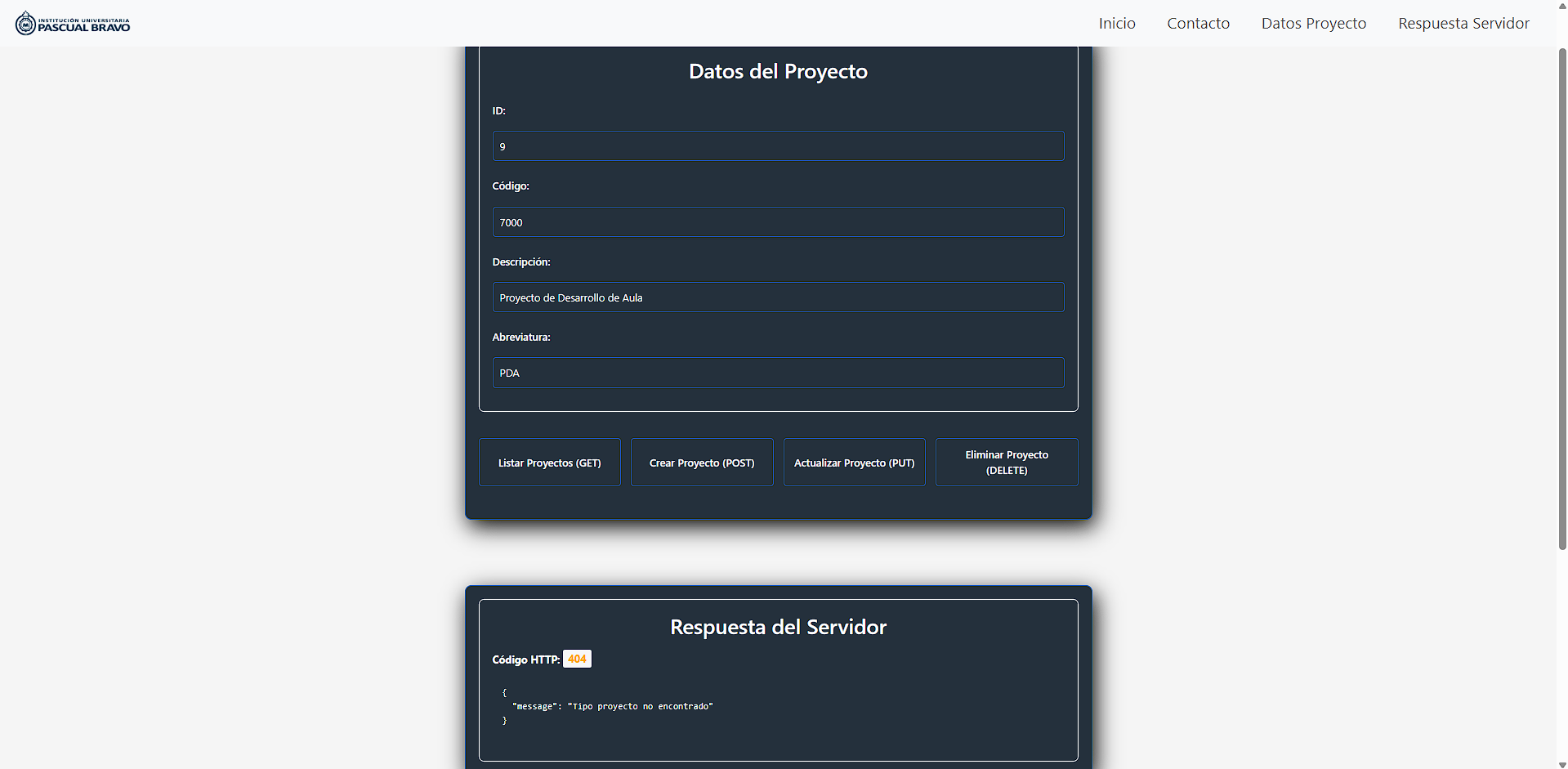
****

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

La actualización del proyecto con ID: 1 se realizo exitosamente. El formulario contenía el Codigo: 1000, la Descripcion: Proyecto de Desarrollo de Aula y la Abreviatura: PDA. Al ejecutar la acción, el servidor respondió con el codigo 200 OK, indicando que el tipo de proyecto fue encontrado y modificado correctamente.

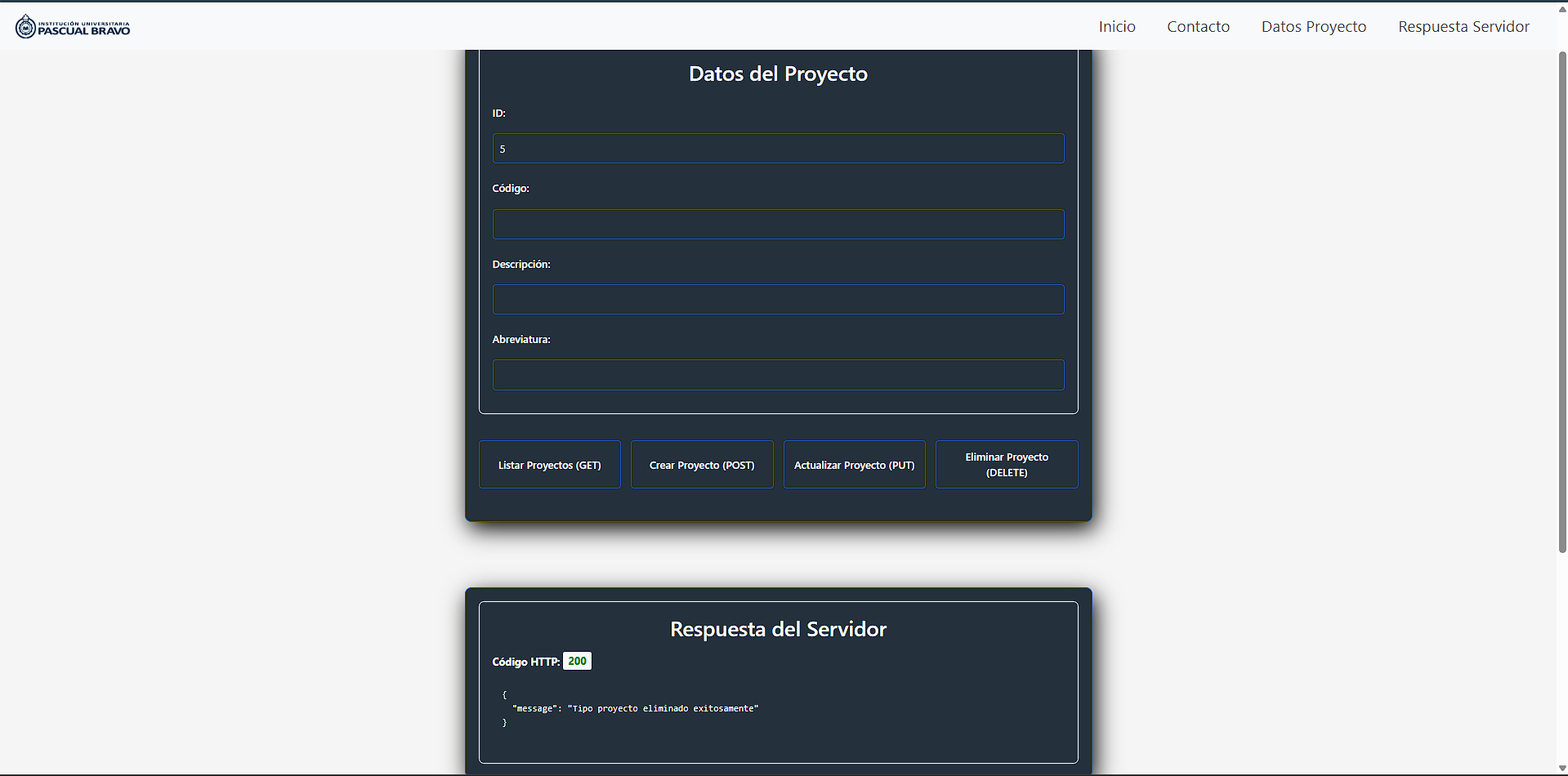
****

se intento actualizar un proyecto con ID: 9, ingresando datos validos en el formulario. Sin embargo, el servidor devolvió un codigo 404 Not Found, lo que indica que no existe un tipo de proyecto con ese ID en la base de datos, por lo tanto, no se pudo completar la actualización.

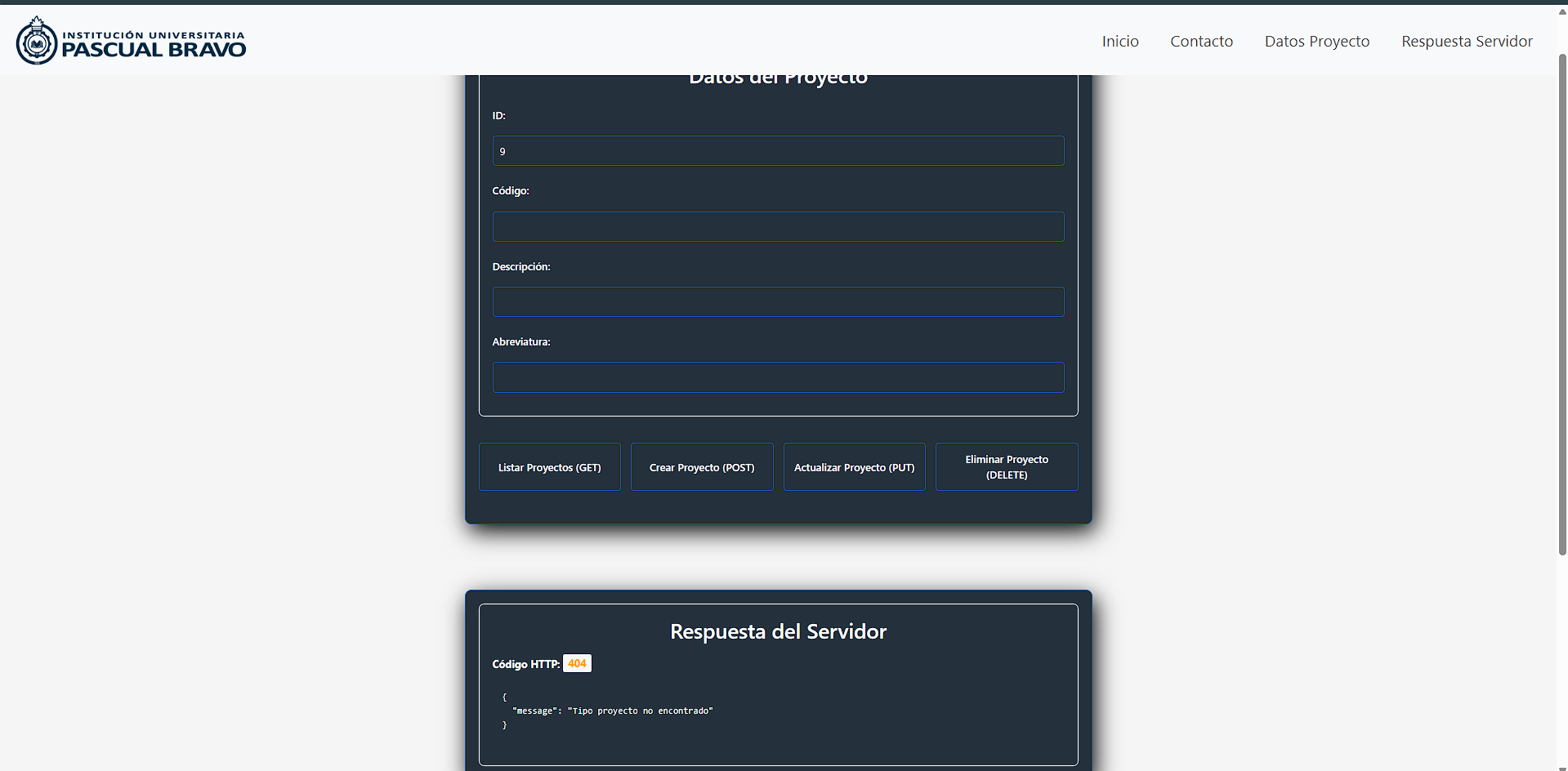
****

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

Se eliminó correctamente el proyecto con ID: 5, el servidor respondió con un codigo 200, confirmando que el tipo de proyecto fue encontrado y eliminado exitosamente.

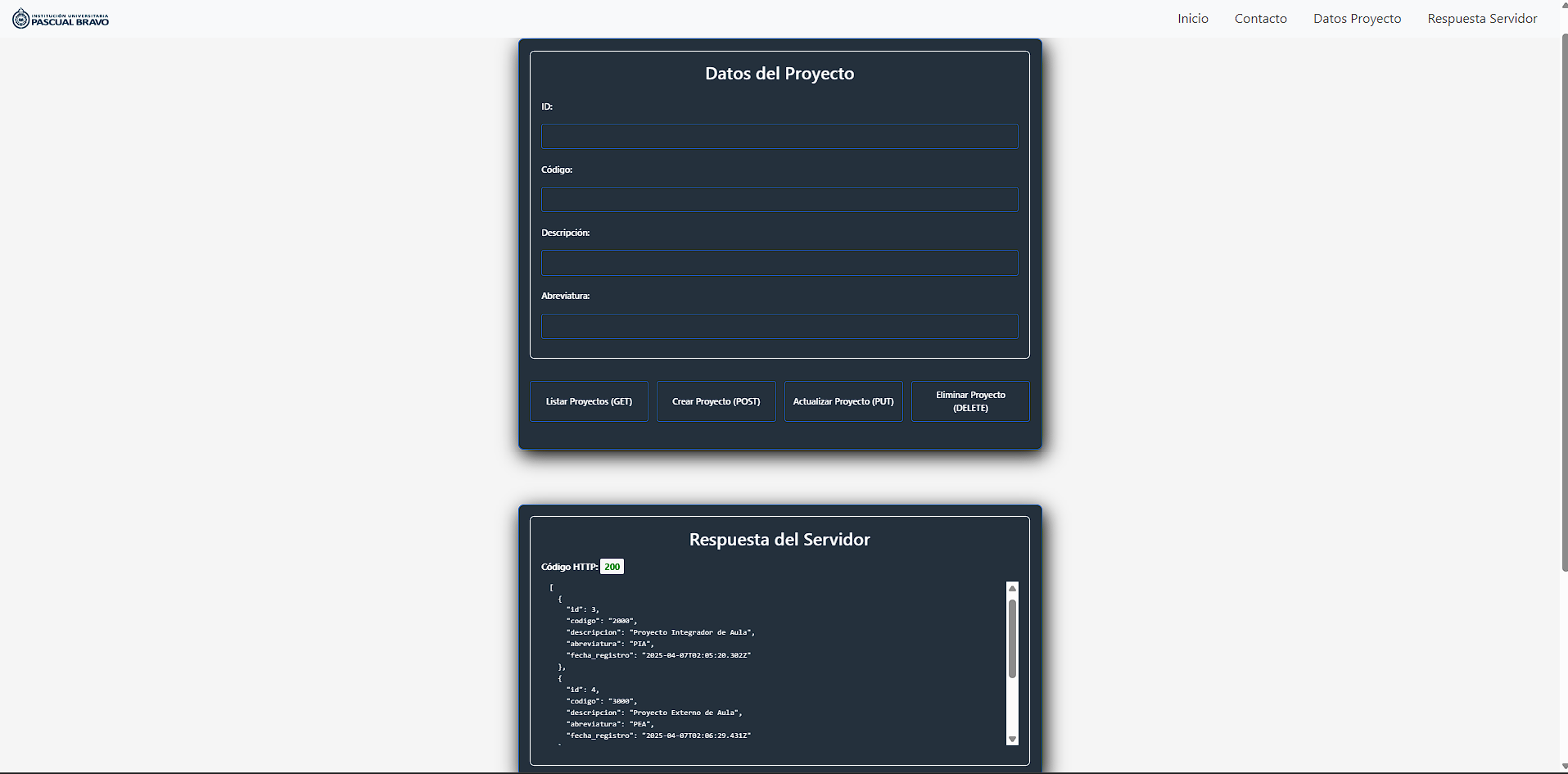
****

Se intento eliminar el proyecto con ID: 9, pero el servidor respondió con un codigo 404, indicando que no existe un tipo de proyecto con ese ID en la base de datos, por lo tanto, no se puede completar la eliminación.

****

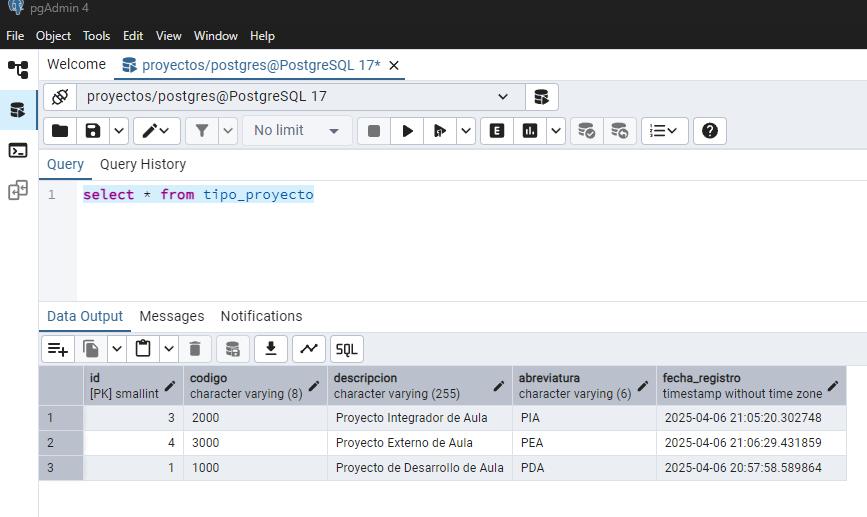
**11.- Consulta de registros (GET/SELECT)**

Al presionar el botón Listar Proyectos (GET). Se ejecuta una solicitud GET a la api. Como los campos de ID y Codigo están vacíos, el sistema interpreta que se desea obtener todos los tipos de datos registrados. En consecuencia, la respuesta del servidor es un codigo http 200 OK y una lista en formato JSON con todos los campos disponibles, la cual se muestra en la sección Respuesta del Servidor.

****

****

muestra el resultado de una consulta SQL ejecutada en PgAdmin sobre la tabla tipo\_proyecto. Se utiliza el comando SELECT \* FROM tipo\_proyecto, lo que permite obtener y visualizar todos los registros almacenados. La tabla contiene campos como ID, codigo, descripcion, abreviatura y fecha\_registro, con tres proyectos registrados: Proyecto de Desarrollo de Aula, Proyecto Integrador de Aula y Proyecto Externo de Aula.

****

**12.- Conclusiones Individuales**.

**Juan jose ramos Agudelo**

Durante el desarrollo del Proyecto Integrador de Aula (PIA), tuve la oportunidad de aplicar de manera práctica muchos de los conocimientos adquiridos a lo largo del semestre en el área de desarrollo web fullstack. Esta experiencia me permitió no solo reforzar conceptos técnicos sobre tecnologías como HTML, CSS, JavaScript, Node.js y PostgreSQL, sino también entender cómo estas herramientas se integran para construir soluciones funcionales, escalables y orientadas al usuario final. La creación de una API REST, junto con una interfaz capaz de consumirla y manejar datos mediante operaciones CRUD, fue un ejercicio valioso para consolidar mi comprensión del flujo completo del desarrollo de una aplicación web.

Uno de los aprendizajes más importantes fue comprender la lógica detrás de la estructura cliente-servidor, desde la solicitud que realiza el frontend hasta el procesamiento que ocurre en el backend y la posterior respuesta al cliente. Este entendimiento me permitió visualizar el sistema como un conjunto de componentes que trabajan de forma coordinada, lo cual es esencial en cualquier entorno profesional. Además, trabajar con bases de datos relacionales como PostgreSQL reforzó mis conocimientos sobre diseño de esquemas, relaciones entre entidades y buenas prácticas de normalización.

La experiencia de desarrollar una solución funcional a partir de requerimientos reales también fortaleció mis competencias en el análisis de problemas, el diseño lógico de soluciones y la toma de decisiones técnicas. Por ejemplo, al decidir cómo organizar los recursos del backend o cómo estructurar el frontend para mantener una experiencia de usuario clara y efectiva, tuve que considerar no solo aspectos funcionales, sino también de rendimiento, mantenibilidad y escalabilidad.

En cuanto a competencias blandas, el proyecto me permitió desarrollar habilidades como la planificación, la organización del trabajo y la comunicación técnica. Al trabajar en fases organizadas (análisis, diseño, codificación y pruebas), aprendí a gestionar el tiempo de manera más eficiente y a priorizar tareas. Además, documentar el proyecto, explicar su funcionamiento y justificar las decisiones tomadas fueron ejercicios útiles para desarrollar mi capacidad de expresión oral y escrita, aspectos fundamentales en el ámbito profesional.

La relación entre el PIA y mi futuro ejercicio profesional es directa. En el mundo laboral, se espera que un desarrollador no solo domine las herramientas técnicas, sino que también sea capaz de comprender requerimientos, proponer soluciones efectivas y trabajar en equipo. Este proyecto fue una simulación realista de ese entorno, donde el aprendizaje no se limitó a lo técnico, sino que involucró la resolución de problemas, el aprendizaje autónomo y la adaptación a nuevas tecnologías. Todo ello me dio una perspectiva más amplia de lo que significa trabajar en el desarrollo de software en el mundo real.

Finalmente, puedo afirmar que este proyecto no solo reforzó mis conocimientos, sino que también me motivó a seguir profundizando en el desarrollo de soluciones web. Siento que ahora tengo una base sólida sobre la cual puedo construir habilidades más avanzadas, y me siento más preparado para enfrentar desafíos profesionales en el área del desarrollo de software. En definitiva, el PIA fue una experiencia valiosa que marcó un antes y un después en mi formación como desarrollador.

**Estiven toro Henao**

Durante este proyecto, tuve la oportunidad de poner en práctica muchos de los conocimientos que adquirimos en clase, y también de aprender cosas nuevas sobre la marcha. Al principio, parecía solo otra tarea técnica, pero a medida que avanzábamos, me di cuenta de que estábamos creando algo mucho más completo: un sistema real, funcional y con su propia lógica.

Trabajar en un proyecto fullstack me ayudó a entender mejor cómo se conectan las diferentes partes de una aplicación web. Por ejemplo, usar HTML, CSS y JavaScript para diseñar la interfaz era algo que ya conocía, pero lo realmente interesante fue ver cómo eso se comunicaba con el backend desarrollado en Node.js, y cómo, a su vez, el backend interactuaba con la base de datos en PostgreSQL. Todo esto se unió a través de una API REST, que al principio me resultó un poco complicada, pero que logré entender mucho mejor gracias a esta experiencia. Me costó al principio, pero luego le fui cogiendo el ritmo.

Uno de los momentos que más disfruté fue observar cómo el sistema respondía al crear, actualizar o eliminar un tipo de proyecto. Me pareció muy valioso implementar validaciones en el formulario para evitar campos vacíos, mostrar mensajes más claros, o incluso algo tan sencillo como reiniciar el formulario después de enviar los datos. Son pequeños detalles, pero que hacen que todo se sienta más cuidado y profesional. Me pareció chévere ver cómo todo se conectaba y empezaba a tomar forma.

En relación al PIA, este proyecto fue una forma muy concreta de aplicar todo lo que aprendimos durante el semestre. No fue una tarea aislada, sino un trabajo que integró frontend, backend, base de datos, validaciones, estructura de carpetas, rutas, respuestas HTTP, control de versiones con Git, entre otras cosas. Me pareció genial que no solo se tratara de que funcionara, sino de que estuviera bien organizado y tuviera sentido como un sistema completo. Incluso la parte del script .bat para crear la estructura del proyecto fue una manera práctica de aplicar algo que uno nunca imagina que va a usar y sí se usa. En serio me sorprendió que algo tan sencillo nos ayudara tanto.

A nivel personal, también aprendí a trabajar mejor en equipo. A veces una piensa que solo con saber programar es suficiente, pero me di cuenta de que la colaboración y la comunicación son igual de importantes.

En general, me siento mucho más preparado para enfrentar proyectos reales y seguir creciendo como desarrollador. Este proyecto me ayudó a ver que lo aprendido en clase realmente se puede aplicar en situaciones concretas, y además me dejó con más confianza para seguir explorando este camino.

**John Jairo Cañaveral Gutiérrez**

Este proyecto fue una gran oportunidad para aplicar conocimientos técnicos en el desarrollo full stack, fortaleciendo habilidades tanto en diseño de interfaces como en lógica de programación y manejo de datos. El uso de HTML, CSS, JavaScript, Bootstrap y respuestas HTTP permitió construir una aplicación responsive, funcional y alineada con las necesidades del usuario. Esta experiencia no solo aportó conocimientos técnicos, sino también habilidades de trabajo en equipo, solución de problemas y adaptación a nuevas tecnologías.

Durante el desarrollo de este proyecto, se logró integrar de manera exitosa tanto el frontend como el backend, permitiendo la creación de una aplicación funcional, interactiva y visualmente atractiva. El uso de tecnologías como HTML, CSS y JavaScript fue fundamental para construir una interfaz clara, organizada y amigable para el usuario. HTML permitió estructurar adecuadamente el contenido, CSS se utilizó para darle estilo y personalización a cada componente visual, y JavaScript permitió añadir dinamismo e interactividad a la aplicación. Además, se implementó Bootstrap, lo que facilitó el diseño responsive y adaptativo, garantizando que la aplicación funcionara correctamente en distintos tamaños de pantalla, desde dispositivos móviles hasta pantallas de escritorio.

En el frontend, se puso especial atención en la experiencia del usuario, utilizando componentes de Bootstrap como botones, formularios y tarjetas, modales. Esto permitió mantener una apariencia coherente en toda la aplicación. La adaptabilidad del diseño responsive fue probada en diferentes dispositivos, comprobando que el contenido se ajustaba correctamente, sin pérdida de funcionalidad ni problemas visuales. Este aspecto fue clave para asegurar que cualquier usuario pudiera interactuar con el sistema desde cualquier lugar.

Por otro lado, el backend fue desarrollado para manejar de forma segura y eficiente las peticiones y respuestas del sistema. Se trabajó con rutas, controladores y lógica de negocio que permitieron recibir solicitudes HTTP (GET, POST, PUT, DELETE), procesarlas, acceder a una base de datos y devolver respuestas acordes a lo solicitado. La comunicación entre frontend y backend fue fluida gracias a estas respuestas HTTP bien estructuradas, lo que permitió funcionalidades como el envío de formularios, validaciones y actualización de datos en tiempo real.

Además, se priorizó la organización del código y la separación de responsabilidades, lo cual facilitó el trabajo colaborativo y el mantenimiento del proyecto a largo plazo. Cada parte del sistema fue diseñada teniendo en cuenta principios básicos del desarrollo web, como la reutilización de componentes y el control de errores.

La integración entre ambas partes (frontend y backend) evidenció la importancia de una buena comunicación cliente-servidor. Al implementar endpoints correctamente definidos y respuestas HTTP claras, el frontend pudo interpretar y mostrar los datos de manera efectiva, mientras que el backend pudo validar, procesar y responder con rapidez.

**Ximena Zamudio Mesa**

En la etapa final de este proyecto de desarrollo de una plataforma integral para la gestión de proyectos, resulta imperativo reflexionar sobre la arquitectura tecnológica implementada y los logros alcanzados. La concepción y materialización de esta herramienta han requerido la sinergia de diversos componentes, cada uno seleccionado estratégicamente para contribuir a la funcionalidad, la eficiencia y la experiencia del usuario.

En el corazón de esta aplicación reside PostgreSQL, un sistema de gestión de bases de datos relacional de código abierto que ha demostrado ser una elección robusta y confiable para el almacenamiento y la organización de la información crítica de los proyectos. Su capacidad para manejar grandes volúmenes de datos con integridad y su extensibilidad mediante diversas funcionalidades avanzadas lo convierten en un pilar fundamental de la infraestructura de datos de nuestra plataforma. La elección de PostgreSQL no solo garantiza la persistencia y la seguridad de la información de los proyectos, sino que también facilita la implementación de consultas complejas y la optimización del rendimiento de la aplicación en general.

Complementando la solidez de la base de datos, el desarrollo de un frontend intuitivo y completamente responsivo ha sido una prioridad central. La interfaz de usuario se ha diseñado meticulosamente para ofrecer una experiencia fluida y accesible desde una amplia gama de dispositivos, ya sean computadoras de escritorio, tabletas o teléfonos inteligentes. Esta adaptabilidad se ha logrado gracias a la implementación de principios de diseño responsivo y a la adopción del framework Bootstrap. Bootstrap, como un conjunto de herramientas de código abierto para el diseño web, ha proporcionado una base sólida de componentes de interfaz de usuario predefinidos y personalizables, acelerando el proceso de desarrollo y garantizando una estética visual coherente y profesional en toda la aplicación. La facilidad de uso y la claridad de la interfaz son factores clave para fomentar la adopción y la eficiencia de la plataforma por parte de los usuarios finales.

En la trastienda de la aplicación, el backend desempeña un papel crucial en la gestión de la lógica de negocio, la manipulación de los datos y la comunicación con la base de datos. Se ha construido con el objetivo de ser eficiente, escalable y capaz de manejar las diversas operaciones requeridas para la gestión de proyectos, desde la creación y asignación de tareas hasta el seguimiento del progreso y la generación de informes. Si bien los detalles técnicos internos, como la gestión de posibles errores que podrían manifestarse en respuestas HTTP específicas, son fundamentales para la robustez del sistema, el enfoque principal para el usuario final radica en la funcionalidad y la fiabilidad del backend para garantizar una experiencia sin interrupciones. La capacidad del backend para procesar las solicitudes de manera eficiente y para mantener la integridad de los datos es esencial para el correcto funcionamiento de la plataforma en su conjunto.

La integración armoniosa de PostgreSQL, el frontend desarrollado con Bootstrap y el backend eficiente ha dado como resultado una plataforma de gestión de proyectos que no solo cumple con los requisitos funcionales iniciales, sino que también ofrece una experiencia de usuario optimizada y adaptable. La robustez de la base de datos asegura la fiabilidad de la información, la interfaz intuitiva facilita la interacción de los usuarios y la eficiencia del backend garantiza un rendimiento adecuado incluso bajo cargas de trabajo significativas.

En conclusión, este proyecto representa un avance significativo en la creación de una herramienta integral para la gestión de proyectos. La cuidadosa selección e integración de tecnologías clave, como PostgreSQL para el almacenamiento de datos, Bootstrap para el diseño de la interfaz y un backend eficiente para la lógica de negocio, han convergido para producir una solución funcional, robusta y centrada en el usuario. La plataforma desarrollada tiene el potencial de optimizar la planificación, la ejecución y el seguimiento de proyectos, contribuyendo a una gestión más eficiente y efectiva en diversos contextos. Los cimientos tecnológicos establecidos proporcionan una base sólida para futuras mejoras y la expansión de las funcionalidades de la plataforma.

**Link Video de Sustentación**

[**https://youtu.be/yebnzFz4zvw**](https://youtu.be/yebnzFz4zvw)

**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** |  |