

INFORME ESCRITO
DISEÑO Y DESARROLLO DE SOFTWARE EDUCATIVO II

CUADRO HERNÁNDEZ JUAN SEBASTIÁN

GALEANO GUERRA GABRIEL

GULLOSO BUELVAS JOSÉ CARLOS

ORTIZ GONZÁLEZ CARLOS DANIEL

RIVAS ÁLVAREZ DANILO SANTOS

MG. FRANKLIN MARTÍNEZ ÁVILA

UNIVERSIDAD DE CÓRDOBA

FACULTAD DE EDUCACIÓN Y CIENCIAS HUMANAS

LICENCIATURA EN INFORMÁTICA CON ÉNFASIS EN MEDIOS

AUDIOVISUALES

MONTERÍA-CÓRDOBA

2023

TABLA DE CONTENIDO

INFORME ESCRITO.....	7
1. ANALISIS DE NECESIDAD	8
2. CRONOGRAMA DE ACTIVIDAD	10
3. TABLA DE ROLES	13
4. DISEÑO DE FINES EDUCATIVOS.....	13
5. FORMATO DE COMPETENCIAS	14
6. DISEÑO DE MATRIZ DE CONTENIDOS	15
7. MODELO PEDAGOGICO EN SOFRWARE EDUCATIVO.....	16
8. DISEÑO DE INTERFAZ	20
9. MARCO TEÓRICO	20
9.1. BACKEND	20
8.1.2. ELEMENTOS QUE CONFORMAN EL BACKEND.....	20
8.1.4. EJEMPLOS DE BACKEND.....	21
10. DESCRIPCIÓN DE VENTANAS	22
11. MODELO E-R.....	31
12. DESARROLLO DE LA BASE DE DATOS	34
12.1. ENTIDAD ACTIVIDADES.	34
12.2. ENTIDAD CONTENIDOS.....	35
12.3. ENTIDAD EVALUACIÓN.	36
12.4. ENTIDAD USUARIOS.	37
12.5. ENTIDAD DATOS DE EVALUACIÓN	38
12.6. ENTIDAD NOTAS.	40
12.7. ENTIDAD PROGRESO.	42
12.8. ENTIDAD PREGUNTAS.....	43
Bases de datos no relacionales (NOSQL).....	45

Modelos de datos flexibles	45
Escalabilidad.....	45
Rendimiento.....	45
Facilidad de uso	45
Aplicaciones web.....	45
Análisis de big data.....	46
Aplicaciones en tiempo real	46
APIs	55
Aplicaciones web.....	55
Aplicaciones móviles.....	55
Sistemas backend.....	55
Beneficios	55
Inconvenientes	56
Ejemplo práctico de la implementación de una API con NodeJs y Express.js.....	56
REFERENCIAS	74

TABLA DE ILUSTRACIONES

ILUSTRACIÓN 1 (PRIMER MODELO E-R)	32
ILUSTRACIÓN 2 (VISTA DISEÑADOR).....	33
ILUSTRACIÓN 3 (ATRIBUTOS DE LAS ACTIVIDADES)	34
ILUSTRACIÓN 4 (DATOS DE LAS ACTIVIDADES).....	35
ILUSTRACIÓN 5 (ATRIBUTOS DE LOS CONTENIDOS)	35
ILUSTRACIÓN 6 (DATOS DE LOS CONTENIDOS).....	36
ILUSTRACIÓN 7 (ATRIBUTOS EVALUACIÓN)	36
ILUSTRACIÓN 8 (DATOS DE LA EVALUACIÓN).....	37
ILUSTRACIÓN 9 (ATRIBUTOS DEL USUARIO).....	37
ILUSTRACIÓN 10 (DATOS DEL USUARIO).....	38
ILUSTRACIÓN 11 (ATRIBUTOS DATOS DE EVALUACIÓN)	38
ILUSTRACIÓN 12 (DATOS DE LAS EVALUACIONES REALIZADAS 25 DE 95)	39
ILUSTRACIÓN 13 (ATRIBUTOS DE LAS NOTAS)	40
ILUSTRACIÓN 14 (DATOS DE LAS NOTAS)	41
ILUSTRACIÓN 15 (ATRIBUTOS DEL PROGRESO)	42
ILUSTRACIÓN 16 (DATOS DE LOS PROGRESOS)	42
ILUSTRACIÓN 17 (ATRIBUTOS DE LAS PREGUNTAS).....	43
ILUSTRACIÓN 18 (DATOS DE LAS PREGUNTAS 25 DE 95).....	44
ILUSTRACIÓN 19 (COLECCIÓN USERS).....	47
ILUSTRACIÓN 20 (COLECCIÓN EVALUATIONS).....	48
ILUSTRACIÓN 21 (COLECCIÓN CONTENTS)	49
ILUSTRACIÓN 22 (COLECCIÓN ACTIVITIES)	50
ILUSTRACIÓN 23 (COLECCIÓN PROGRESS).....	51
ILUSTRACIÓN 24 (COLECCIÓN EVALUATIONDATAS)	52
ILUSTRACIÓN 25 (COLECCIÓN QUESTIONS).....	53
ILUSTRACIÓN 26 (COLECCIÓN NOTES)	54

TABLA DE DATOS

TABLA 1 (RUTAS AMIGABLES).....	73
--------------------------------	----

INFORME ESCRITO

El informe tiene en cuenta los siguientes puntos:

- ✓ Introducción tipo marco teórico sobre qué es Backend y sus elementos.
- ✓ Explicar de acuerdo con los requisitos del sistema qué entidades se identificaron con sus respectivos atributos.
- ✓ Adjuntar capturas de pantallas de las entidades por la paleta estructura de MySQL con los tipos de datos y llaves primarias de cada entidad de dato, también anexar capturas de los datos registrados en todas las tablas (incluyendo las relacionales), al igual que al diseñador.
- ✓ Realizar una descripción explicando cada imagen al igual el archivo NoSql lo pegan allí y explican cada colección y las relaciones que existen en cada colección.
- ✓ Qué es un API, adjuntar las rutas realizadas en clase explicando cada una y su funcionamiento con Node.js y Express.

1. ANALISIS DE NECESIDAD

ANÁLISIS DE LA NECESIDAD EDUCATIVA	
Tipo: Sentida	La necesidad a tratar es de tipo sentida, debido a que es un problema detectado por un docente de la Institución Educativa Rafael Núñez de Córdoba
Identificación del aprendizaje ideal	
<p>Los estudiantes deben conocer e identificar:</p> <ol style="list-style-type: none"> 1. Describo y modelos fenómenos periódicos del mundo real usando relaciones. 2. Identifico características de localización de objetos geométricos en sistemas de representación cartesiana y otros (polares, cilíndricos y esféricos) y en particular de las curvas y figuras cónicas. 3. Resuelvo problemas en los que se usen las propiedades geométricas de figuras cónicas por medio de transformaciones de las representaciones algebraicas de esas figuras. 	
Población	Rango de edad: 16 años en adelante
	Escolaridad: Grado 10° en adelante
	Conocimiento que posee: Deben tener conocimientos básicos sobre figuras geométricas bidimensionales y tridimensionales, y saber el manejo de un computador.
	Intereses y expectativas: (Población) Los estudiantes presentan cierto interés por el uso de las tecnologías como mediadores de los procesos de enseñanza y aprendizaje, por tal razón la expectativa que se tiene es que estos mejoren sus conocimientos en estos temas mediante las TIC.
	Intereses y expectativas: (Creadores) Desarrollar el pensamiento espacial en los estudiantes de educación media de manera creativa de tal forma que se dinamicen los procesos de enseñanza y los estudiantes superen las dificultades de aprendizaje.
Área de formación	Area del saber: Matemáticas
	Area de contenido: Pensamiento espacial y sistemas geométricos.
Estado actual	<p>Diagnóstico:</p> <p>Analizando la situación problemática de la institución en cuanto al área de matemáticas se evidencia la dificultad que presentan en lograr algunos desempeños del pensamiento espacial y los sistemas geométricos en los grados de educación media. Los estudiantes se les dificulta modelar figuras en torno a fenómenos periódicos, desconocen el uso de las funciones trigonométricas, les cuesta localizar objetos o figuras en los planos cartesianos e identificar representaciones algebraicas y curvas, lo cual les dificulta la resolución problemas. La institución posee las herramientas necesarias para el desarrollo de las competencias mediadas por TIC, pero se mantienen bajo las mismas estrategias y métodos de enseñanza sumado a que existen equipos de computación, pero no con conexión a internet.</p>
Necesidad	La necesidad se presenta en torno a los bajos niveles de desempeño de los estudiantes en el área de las matemáticas principalmente en el componente del pensamiento espacial y los sistemas geométricos donde se evidencian las dificultades para representar figuras en los planos cartesianos.

Causas	<ol style="list-style-type: none"> 1. Falta de recursos didácticos para la enseñanza de las matemáticas 2. Uso de estrategias tradicionales por parte de algunos docentes del área de matemáticas 3. Poco interés por parte del estudiante hacia el área
Soluciones	<ol style="list-style-type: none"> 1. Diseñar recursos que ayuden a mediar el proceso de enseñanza de las matemáticas 2. Capacitación a los docentes en el uso de estrategias mediadas por TIC para la enseñanza de las matemáticas. 3. Implementación de un software educativo como estrategia para el mejoramiento en los procesos de enseñanza-aprendizaje en el componente de pensamiento espacial y sistemas geométricos.
Conocimientos y habilidades que debe tener el estudiante	Preconceptos: Que los estudiantes posean conocimientos básicos en geometría.
	Precondiciones: pensamiento espacial, manejo básico del computador, visuales y motrices.
Justificación <p>La enseñanza en las escuelas no ha sido una tarea sencilla y aún más cuando existen profesores que no poseen los suficientes recursos didácticos y estrategias pedagógicas para lograr que todos los estudiantes adquieran los conocimientos sobre las temáticas de las diferentes áreas de la educación básica, más que todo en el área de Matemáticas donde la metodología para llevar a cabo las clases se torna un poco “limitada” puesto que se usan estrategias tradicionales, por ejemplo, el docente realiza una explicación de un ejercicio y resuelve un ejemplo de la explicación de un tema, luego de ello, el profesor les asigna varios problemas a los estudiantes para que apliquen el conocimiento adquirido. Metodologías como la anterior son las que hacen que el estudiante observe la asignatura como algo difícil o imposible de aprender y sea una de sus materias menos favoritas.</p> <p>Las TIC aportan mucho a la mejora de la enseñanza con su amplia gama de herramientas y estrategias tecnológicas que ayudan a los docentes a mejorar su rendimiento en las aulas. Por tal razón se desea desarrollar un software educativo para los estudiantes de grado 10° de la Institución Educativa Rafael Núñez de Córdoba con fin de facilitar el aprendizaje de los sistemas geométricos y el pensamiento espacial, de igual forma brindar una calidad educativa.</p>	

2. CRONOGRAMA DE ACTIVIDAD

[illegible]

[illegible]

[illegible]

3. TABLA DE ROLES

Tabla de roles	
Nombres	Funciones
Carlos Daniel Ortiz González	Diseñador de software
Gabriel Galeano Guerra	Programador
José Carlos Gullos Buevas	Pedagogo
Juan Sebastián Cuadro Hernández	Diseñador grafico
Danilo Santos Rivas Álvarez	Téster o responsable de pruebas.

4. DISEÑO DE FINES EDUCATIVOS

DISEÑO DE FINES EDUCATIVOS	
O b j e t i v o s d e a p r e n d i z a	Objetivo general
	Desarrollar un Software Educativo para el desarrollo del pensamiento espacial y los sistemas geométricos en los estudiantes de grado decimo de la institución educativa Rafael Núñez de la ciudad de Montería, Córdoba
	Objetivos específicos

<i>j</i> <i>e</i>	
----------------------	--

	<ul style="list-style-type: none"> ♦ 1. Identificar las dificultades de los estudiantes en las temáticas que ayudan a desarrollar el pensamiento espacial y los sistemas geométricos. ♦ 2. Elaborar un software educativo que favorezca el desarrollo del pensamiento espacial y los sistemas geométricos en los estudiantes de grado 10°. ♦ 3. Analizar el software educativo por parte de los docentes con unas pruebas escalonadas
Dimensiones	<ul style="list-style-type: none"> ● Capaz de describir y modelar fenómenos periódicos del mundo real ● Capaz de identificar características de localización de objetos geométricos en sistemas de representación cartesiana y otros (polares, cilíndricos y esféricos) y en particular de las curvas y figuras cónicas. ● Capaz de resolver problemas en los que se usen las propiedades geométricas de figuras cónicas por medio de transformaciones de las representaciones algebraicas de esas figuras.
V a l o r e s	<p>El presente proyecto puede fortalecer:</p> <ol style="list-style-type: none"> 1. El trabajo grupal en algunos estudiantes. 2. Creatividad para identificar información 3. Razonamiento lógico-matemático

5. FORMATO DE COMPETENCIAS

Competencia 1	Tipo Cognitiva
Objetivos	Norma
1: Enunciado	1: Contexto
Representa lugares geométricos en el plano cartesiano, a partir de su expresión algebraica.	Situación en que se le solicite a un estudiante graficar una función dentro del plano cartesiano.
	2: Recursos
	<ul style="list-style-type: none"> ● Software educativo de matemáticas ● Contenido multimedia sobre trigonometría

2: Elementos	3: Evidencias
1- Identificar características de localización de objetos geométricos en sistemas de representación cartesiana y otros (polares, cilíndricos y esféricos) y en particular de las curvas y figuras cónicas.	<p>1.1 Hace uso del plano cartesiano para representar características de objetos geométricos.</p> <p>1.2 Reconoce los diferentes sistemas de coordenadas (cartesiano, polares, cilíndricos y esféricos) para utilizar el</p>

	más adecuado.
2- Describir y modelar fenómenos periódicos del mundo real usando relaciones y funciones trigonométricas.	<p>2.1 Describe situaciones de la vida cotidiana haciendo uso de funciones trigonométricas</p> <p>2.2 Relaciona conceptos con su entorno físico y lo pone en práctica.</p>
Conceptos	
<ul style="list-style-type: none"> ● Matemáticas: Pensamiento espacial ● Tecnología e informática: Uso del computador, navegadores, internet 	
Habilidades y destrezas	
<ul style="list-style-type: none"> ● Intelectual: Tener conocimientos previos de aritmética y operaciones con polinomios, manejo básico y utilización del computador. ● Social: capacidad de estar en grupos y de explorar e interactuar escuchando opiniones y sugerencias. ● Físicas: tener unas condiciones físicas y mentales normales. 	

6. DISEÑO DE MATRIZ DE CONTENIDOS

CONCEPTOS DE LAS COMPETENCIAS		
Conceptos	Características	Definición
Pensamiento Espacial	<ul style="list-style-type: none"> ● Establece relaciones entre los objetos ● Reconoce atributos ● Identifica conceptos ● Construcción y manejo de las representaciones mentales del ambiente. 	El conjunto de los procesos cognitivos mediante los cuales se construyen y se manipulan las representaciones mentales de los objetos del espacio, las relaciones entre ellos, sus transformaciones, y sus diversas traducciones o representaciones materiales. MEN (1998)
Internet	<ul style="list-style-type: none"> ● Interactivo y automático ● Universal y global ● Es móvil ● Fácil acceso. 	Es una red masiva de redes, infraestructura de redes que conecta a millones de computadoras unidas de forma global; formando una sola red en la que una computadora puede comunicarse con otra siempre y cuando estén las dos computadoras conectadas al Internet. Snell (1995)

7. MODELO PEDAGOGICO EN SOFTWARE EDUCATIVO

MODELO PEDAGOGICO CONECTIVISTA		
Bases conceptuales	Características	Principios educativos
El conectivismo se trata de explicar el aprendizaje complejo en un mundo social digital en rápida evolución donde el aprendizaje se produce a través de las conexiones dentro de las redes. Según George Siemens el aprendizaje es un proceso que ocurre dentro de los entornos virtuales de aprendizaje en elementos básicos, no enteramente bajo el control del individuo (estudiante).	<ul style="list-style-type: none"> • El aprendizaje y el conocimiento se nutre de la aportación de diferentes opiniones. • El aprendizaje puede producirse en diferentes contextos y no siempre estar bajo el control de la persona pudiendo ser este un dispositivo no humano. • Capacidad de discernir entre la información relevante y la que no lo es. 	<p>CHOQUE COGNITIVO</p> <p>Al estudiante se le presentan una serie de problemas basados en figuras poligonometricas donde le debe dar solución a través de las herramientas que se le brindan dentro del software, llevándolo así a buscar soluciones a estas dificultades.</p>
Enfoque	Metáfora Educativa	
En el enfoque del modelo conectivista es social porque según el modelo este se crear más allá del nivel individual de los participantes humanos y está cambiando	Papel del docente: Es orientar a los estudiantes a elegir fuentes confiables de información y a su vez “seleccionar” la información más importante, es decir, tener la	

<p>constantemente y se considera social por la interacción de los individuos en un medio físico o virtual.</p>	<p>habilidad de discernir entre la información que es importante y la que es trivial.</p> <p>El docente explicara las actividades y el como utilizar el software cumpliendo un papel de mediador.</p> <p>Papel del estudiante: Se centra en adquirir la habilidad para seleccionar entre tantas formas y medios de información y de comunicación.</p> <p>Al estudiante se le brindara dentro del software actividades relacionadas a polígonos con el fin de reforzar los conocimientos, si este presenta dificultades con la resolución de problemas puede acudir a la ayuda del docente a través de un sistema de mensajería o de las pistas que se encontraran dentro del software.</p>	
--	---	--

COMPETENCIA		
Representa lugares geométricos en el plano cartesiano, a partir de su expresión algebraica.		
Elementos	Aplicación del modelo pedagógico	Indicadores
Identificar características de localización de objetos geométricos en sistemas de representación cartesiana y otros (polares, cilíndricos y esféricos) y en particular de las curvas y figuras cónicas.	El aprendizaje es responsabilidad del estudiante y el docente solo es un mediador del proceso.	1.3 Hace uso del plano cartesiano para representar características de objetos geométricos. 1.4 Reconoce los diferentes sistemas de coordenadas (cartesiano, polares, cilíndricos y esféricos) para utilizar el más adecuado.
Secuencia de aprendizaje		
Objetivo: Hacer uso del plano cartesiano para representar características de objetos geométricos. Reconocer los diferentes sistemas de coordenadas (cartesiano, polares, cilíndricos y esféricos) para utilizar el más adecuado.		BARRA DE RECURSOS
		Navegación: Funciones trigonométricas - Contenidos - Actividades - Evaluación - Contenido extra
Problema: Representa gráficamente y determina las coordenadas de los focos, de los vértices y la excentricidad de la siguiente elipse $\frac{x^2}{16} + \frac{y^2}{12} = 1$		Documentación:
		- PDF - Imágenes - Videos - Audios
Estrategias: - Establecer puntos en un plano cartesiano y sus intersecciones. - Identificar los sistemas de coordenadas en cuanto a sus determinaciones. - Localizar figuras en un plano cartesiano junto a su ubicación y ejes que la conforman. - Desarrollar actividades para la ubicación de elementos en el plano cartesiano con todos sus elementos. (los ejes coordenados, las coordenadas, el origen, y los cuadrantes).		Comunicación:
		- La comunicación entre el estudiante y el docente se dará a través de un a mensajería interna y de igual forma entre estudiante y estudiante

COMPETENCIA

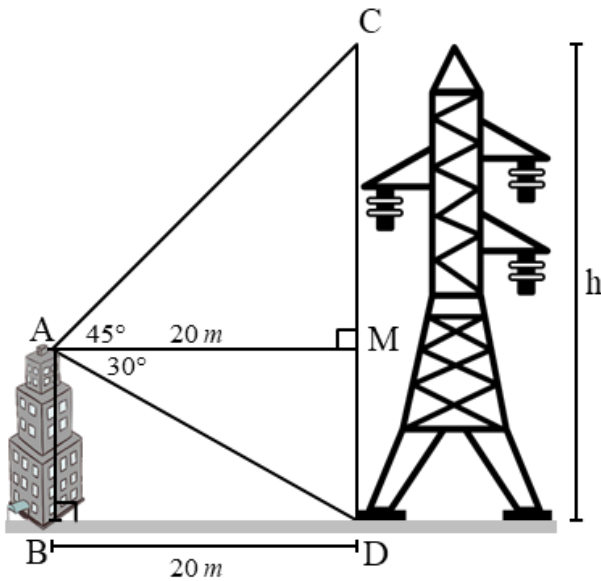
Describir y modelar fenómenos periódicos usando relaciones trigonométricas.

Elementos	Aplicación del modelo pedagógico	Indicadores
Describir y modelar fenómenos periódicos del mundo real usando relaciones y funciones trigonométricas.	El aprendizaje es responsabilidad del estudiante y el docente solo es un mediador del proceso.	<p>2.1 Describe situaciones de la vida cotidiana haciendo uso de funciones trigonométricas.</p> <p>2.2 Relaciona conceptos con su entorno físico y lo pone en práctica.</p>

Secuencia de aprendizaje

Objetivo: Describir situaciones de la vida cotidiana haciendo uso de funciones trigonométricas y relacionar conceptos con su entorno físico y ponerlo en práctica.

Problema:



Desde la azotea de un edificio, Laura observa la parte más alta y la parte más baja de una torre, tal como se muestra en la figura.

Si Laura se encuentra a una distancia de 20 M de la torre, ¿cuál es la altura de la torre?

BARRA DE RECURSOS

Navegación: Funciones trigonométricas

- Contenidos
- Actividades
- Evaluación
- Contenido extra

Documentación:

- PDF
- Imágenes
- Videos
- Audios

Comunicación:

- La comunicación entre el estudiante y el docente se dará a través de un a mensajería interna y de igual forma entre estudiante y estudiante

Estrategias: <ul style="list-style-type: none"> - Revisión de los contenidos (razones trigonométricas) por parte de los estudiantes. - Ejemplo con relaciones de contexto cotidiano y trigonometría para afianzar y poner en práctica los contenidos. - Elaboración de actividades descriptiva y de relación en la sección de actividades del Software <ul style="list-style-type: none"> • La actividad sería relacionar áreas y campos donde se pueden ejecutar las funciones trigonométricas. • Describir y resolver problemas de las áreas y campos ya identificados por ellos. 		8. DISEÑO DE INTERFAZ 9. MARCO TEÓRICO
--	--	---

9.1.BACKEND

El Backend es el encargado de procesar toda la información que alimenta a un Frontend, es decir que es aquella información que el usuario no puede ver ni manipular. Este se compone de marcos, bases de datos o códigos. Para que un sitio web o aplicación opere efectivamente, de tal manera se requiere mucha información y datos que se almacenan en «la parte trasera» de un sistema informático para operar. También es conocido como el lado del servidor, es la parte de una aplicación web que se encarga de procesar las solicitudes del usuario, manejar la lógica de negocio y gestionar la comunicación con la base de datos. Se ocupa de todas las operaciones que no son visibles directamente para el usuario final.

8.1.2. ELEMENTOS QUE CONFORMAN EL BACKEND

- ✓ El Backend se constituye por lenguajes de programación como PHP, Python y C++ y Frameworks.
- ✓ Los servidores controlan cómo los usuarios acceden a los archivos.
- ✓ Las bases de datos son colecciones de datos organizadas y estructuradas.
- ✓ La seguridad es uno de los elementos más importantes dentro de un sitio web, pues garantiza que los visitantes y su información estén seguros. Esto también incluye evitar, en lo posible, ciberataques.

Más específicamente los elementos que conforman el Backend están conformados por:

- **Servidor:** El servidor es un componente esencial del Backend. Puede ser un software o una máquina física que aloja la aplicación web y procesa las solicitudes entrantes. Generalmente, utiliza tecnologías como Apache, Nginx o Microsoft IIS para gestionar las solicitudes y entregar respuestas al cliente.
- **Lenguajes de Programación:** El Backend utiliza lenguajes de programación como Python, Ruby, PHP o Node.js para implementar la lógica de negocio y la funcionalidad de la aplicación. Estos lenguajes permiten el procesamiento de datos, la manipulación de archivos y la interacción con otros componentes del sistema.
- **Frameworks:** Los frameworks del Backend proporcionan un conjunto de herramientas y librerías que simplifican el desarrollo de aplicaciones web. Algunos ejemplos populares son Django (Python), Ruby on Rails (Ruby), Laravel

(PHP) y Express.js (Node.js). Estos frameworks facilitan la creación de rutas, la gestión de bases de datos y la implementación de funciones comunes.

- Base de Datos: El Backend se comunica con una base de datos para almacenar y recuperar información. Puede utilizar diferentes sistemas de gestión de bases de datos como MySQL, PostgreSQL o MongoDB. Estos sistemas permiten la persistencia de datos y su posterior consulta mediante consultas estructuradas.







8.1.3. CARACTERÍSTICAS DE BACKEND



- Seguridad: El Backend es responsable de implementar medidas de seguridad para proteger la aplicación web y los datos sensibles. Esto implica la validación de entradas, el control de acceso, la encriptación de datos y la prevención de ataques como inyecciones SQL y XSS (Cross-Site Scripting).
- Escalabilidad: El Backend debe ser capaz de manejar eficientemente un aumento en la carga de trabajo y el número de usuarios. Debe ser escalable, lo que implica que pueda adaptarse a mayores demandas de recursos y seguir proporcionando un rendimiento óptimo.
- Integración: El Backend se integra con otros servicios y APIs externas para agregar funcionalidades adicionales a la aplicación. Esto puede incluir servicios de pago, servicios de almacenamiento en la nube, servicios de autenticación de terceros, entre otros.
- APIs (Interfaces de Programación de Aplicaciones): El Backend expone APIs que permiten a otras aplicaciones o servicios interactuar con la aplicación web. Estas APIs definen cómo las solicitudes deben ser enviadas y qué respuestas se esperan. Las APIs facilitan la comunicación y la interoperabilidad entre diferentes sistemas.

8.1.4. EJEMPLOS DE BACKEND

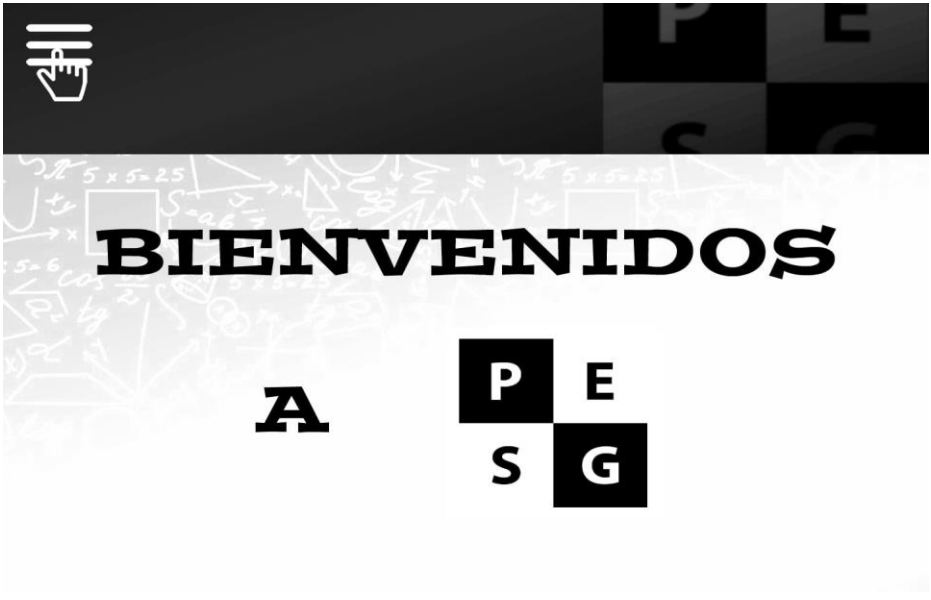
- Inicio de sesión. Cuando una persona accede a un sitio web o aplicación utiliza un correo electrónico y contraseña, esta información es validada y resguardada por el servidor, que consulta su base de datos y así identifica y permite el acceso al usuario.
- Carrito de compras. Este elemento permite la compra de productos en línea y sirve para facilitar la selección de diferentes productos o servicios que algún usuario desee comprar.
- Cookies. Muchos sitios utilizan cookies para realizar un seguimiento de aquello que los usuarios vieron anteriormente, lo que les permite sugerir otros contenidos (o productos) de interés.
- CMS. Un sistema de gestión de contenidos permite al propietario de un sitio web actualizar la información sin tener que modificar el código HTML.
- Formularios de contacto. Si un visitante del sitio web se interesa por recibir más información o ponerse en contacto, se debe contar con un elemento que sea capaz de vincular al usuario con la empresa

10. DESCRIPCIÓN DE VENTANAS

DESCRIPCIÓN DE LAS VENTANAS					
					
Ventana: 01 Inicio de Sesión					
Texto	Imagen	Audio	Videos	Animación	Acciones
-PESG		No	No	Expansión.	
Usuario		No	No	No	En este campo el estudiante ingresa su usuario
Contraseña		No	No	No	En este campo el estudiante ingresa su contraseña
Entrar		No	No	No	Redirecciona al Inicio del software educativo
		No	No	No	Mostrar ayuda

DESCRIPCIÓN DE LAS VENTANAS					
					
Ventana: 02 Cargando					
<i>Texto</i>	<i>Imagen</i>	<i>Audio</i>	<i>Videos</i>	<i>Animación</i>	<i>Acciones</i>
-PESG		No	No	No	
-Cargando	Cargando...	No	No	Los tres puntos aparecen y desaparecen.	Indica al usuario que esta cargando el software educativo.

DESCRIPCIÓN DE LAS VENTANAS








Ventana: 03 Inicio

Texto	Imagen	Audio	Videos	Animación	Acciones
-Menú		No	No	No	Al dar clic despliega la barra lateral del menú.
Bienvenidos a PSEG		No	No	No	

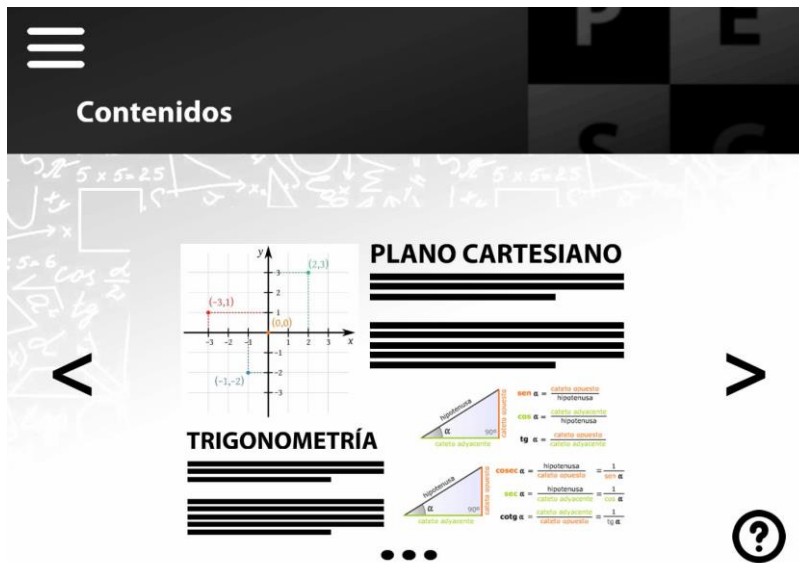
DESCRIPCIÓN DE LAS VENTANAS



Ventana: 03 Despliegue de menú

<i>Texto</i>	<i>Imagen</i>	<i>Audio</i>	<i>Videos</i>	<i>Animación</i>	<i>Acciones</i>
-Inicio	 Inicio	No	No	No	Al dar clic envía al usuario a la ventana de inicio.
-Contenidos	 Contenidos	No	No	No	Al dar clic envía al usuario a la ventana de contenidos.
-Actividades	 Actividades	No	No	No	Al dar clic envía al usuario a la ventana de Actividades.
-Evaluación	 Evaluación	No	No	No	Al dar clic envía al usuario a la ventana de evaluación.
-Créditos	 Créditos	No	No	No	Al dar clic envía al usuario a la ventana de créditos.

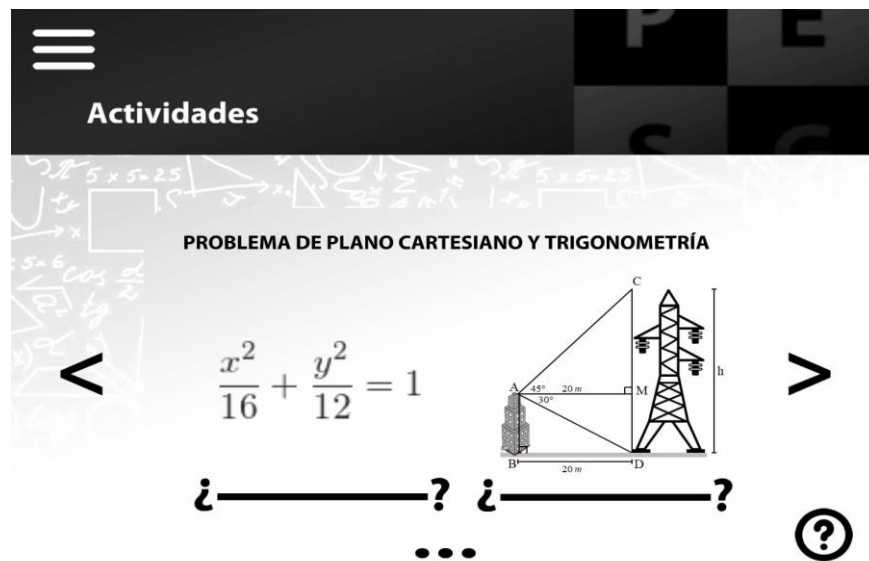
DESCRIPCIÓN DE LAS VENTANAS



Ventana: 04 Contenidos

Texto	Imagen	Audio	Videos	Animación	Acciones
-Contenidos		No	No	No	Indica en la sección que estas del software educativo
PLANO CARTESIANO TRIGONOMETRÍA		No	No	No	
		No	No	No	Al dar clic en las flechas de izquierda o derecha se van mostrando los diferentes contenidos del Software educativo.
		No	No	No	Mostrar ayuda

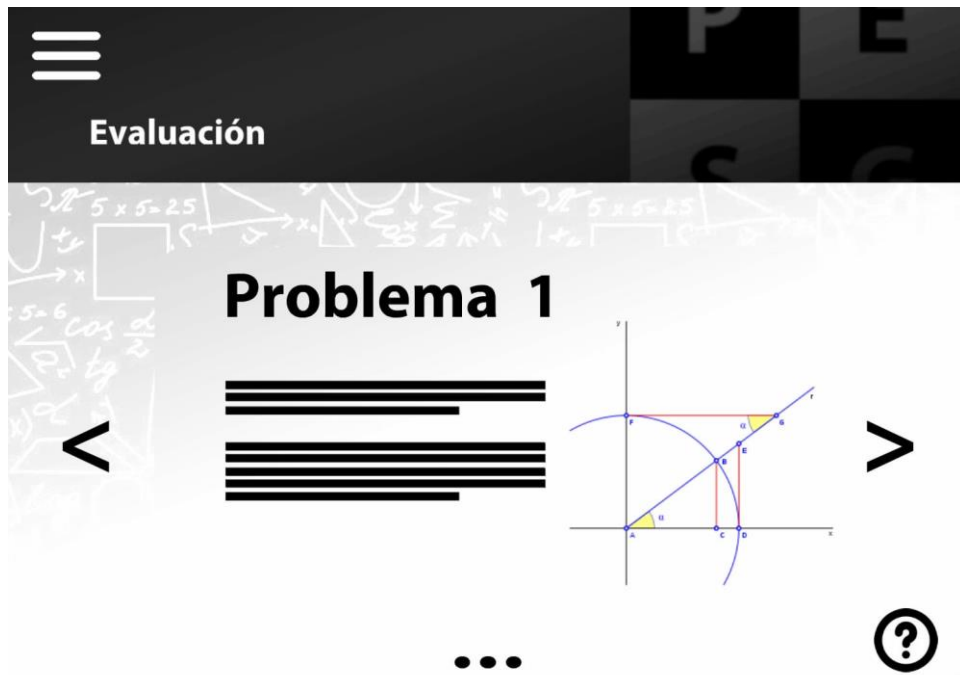
DESCRIPCIÓN DE LAS VENTANAS




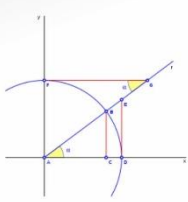
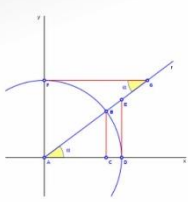


Ventana: 05 Actividades






Texto	Imagen	Audio	Videos	Animación	Acciones
-Actividades		No	No	No	Indica en la sección que estas del software educativo
Problema de plano cartesiano y trigonometría		No	No	No	
	No	No	No	No	
		No	No	No	Al dar clic en las flechas de izquierda o derecha se van mostrando las diferentes actividades del Software educativo.
		No	No	No	Mostrar ayuda










DESCRIPCIÓN DE LAS VENTANAS



Ventana: 06 Evaluación

Texto	Imagen	Audio	Videos	Animación	Acciones
-Evaluación		No	No		Indica en la sección que estas del software educativo
Problema 1 		No	No		
		No	No		Al dar clic en las flechas de izquierda o derecha se van mostrando los diferentes puntos a resolver de la evaluación del Software educativo.
		No	No		Mostrar ayuda

DESCRIPCIÓN DE LAS VENTANAS					
					
Ventana: 07 Créditos					
Texto	Imagen	Audio	Videos	Animación	Acciones
-Créditos		No	No		Indica en la sección que estas del software educativo
					Espacio donde se visualiza a los creadores del software e institución
		No	No		Al dar clic en las flechas de izquierda o derecha se van mostrando los diferentes responsables de la creación del Software educativo.
					Mostrar ayuda

DISEÑO DE GUÍA DE METÁFORAS		
<i>Nombre</i>	<i>Imagen</i>	<i>Descripción</i>
Usuario		Ayuda a identificar el campo de usuario
Contraseña		Ayuda a identificar el campo de contraseña
Menú		Botón de menú, despliega la barra de menú para interactuar con los diferentes ítems de software.
Ayuda		Botón de ayuda para entender el software-.
Inicio		Botón de Inicio. Redirecciona a la ventana de inicio.
Contenidos		Botón de Contenidos. Redirecciona a la ventana de contenido
Actividades		Botón de actividades. Redirecciona a la ventana de actividades
Evaluación		Botón de evaluación. Redirecciona a la ventana de evaluación
Créditos		Botón de créditos. Redirecciona a la ventana de créditos

11. MODELO E-R

Durante el desarrollo del curso de software educativos II se comenzó con la introducción a bases de datos relacionales desarrolladas desde MySQL todo esto con el fin de desarrollar una base para la elaboración del proyecto que se viene trabajando desde software educativos I, el cual aborda el tema de pensamiento espacial y sistemas geométricos enfocados en el grado 10 de la I.E Rafael Núñez. De dicho proyecto se elaboró un diagrama de entidad relación el cual contiene las variables y las relaciones entre estas, mostrando así las llaves primarias de cada variable y sus llaves foráneas ([Ver ilustración 1](#))

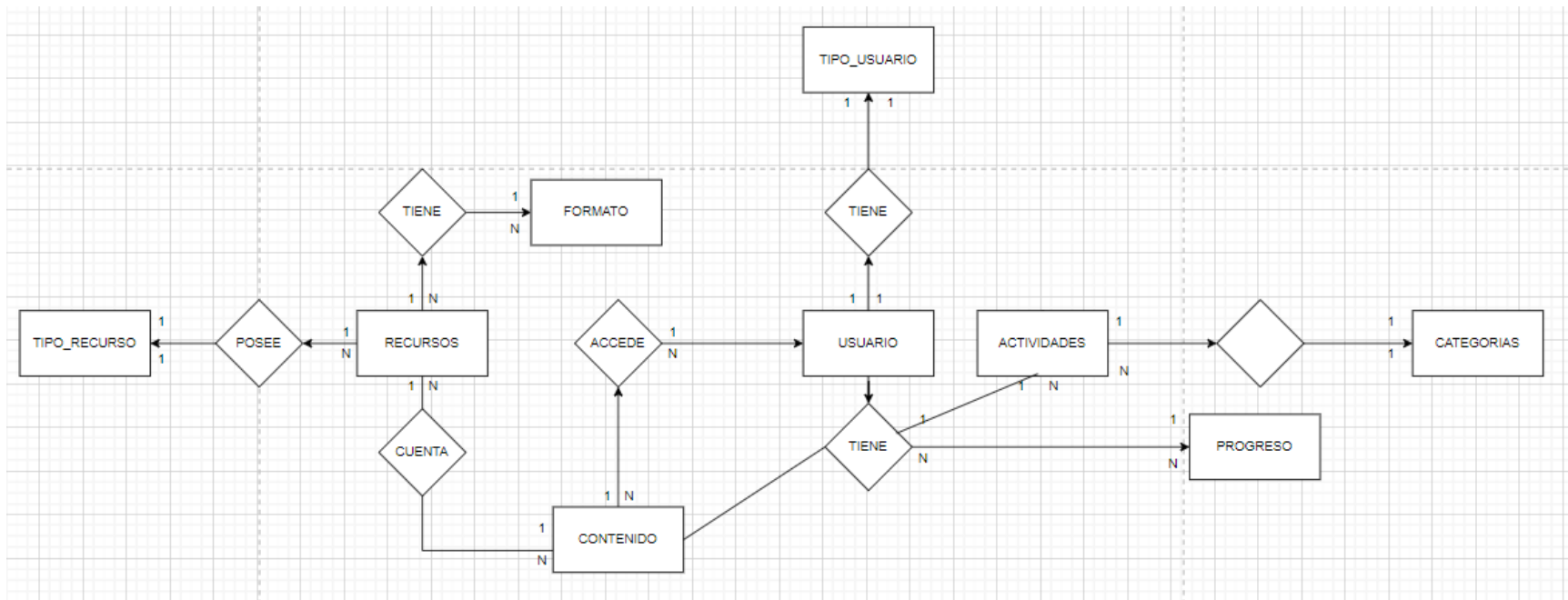


Ilustración 1 (Primer Modelo E-R)

Para el desarrollo de la base de datos se tuvo en cuenta modelo E-R ([Ver ilustración 1](#)). La estructura de la base de datos del proyecto realizada en MySQL es de 8 tablas que contienen de 4 a 8 atributos respectivamente. De las cuales, todas son entidades y representan relaciones entre sí.

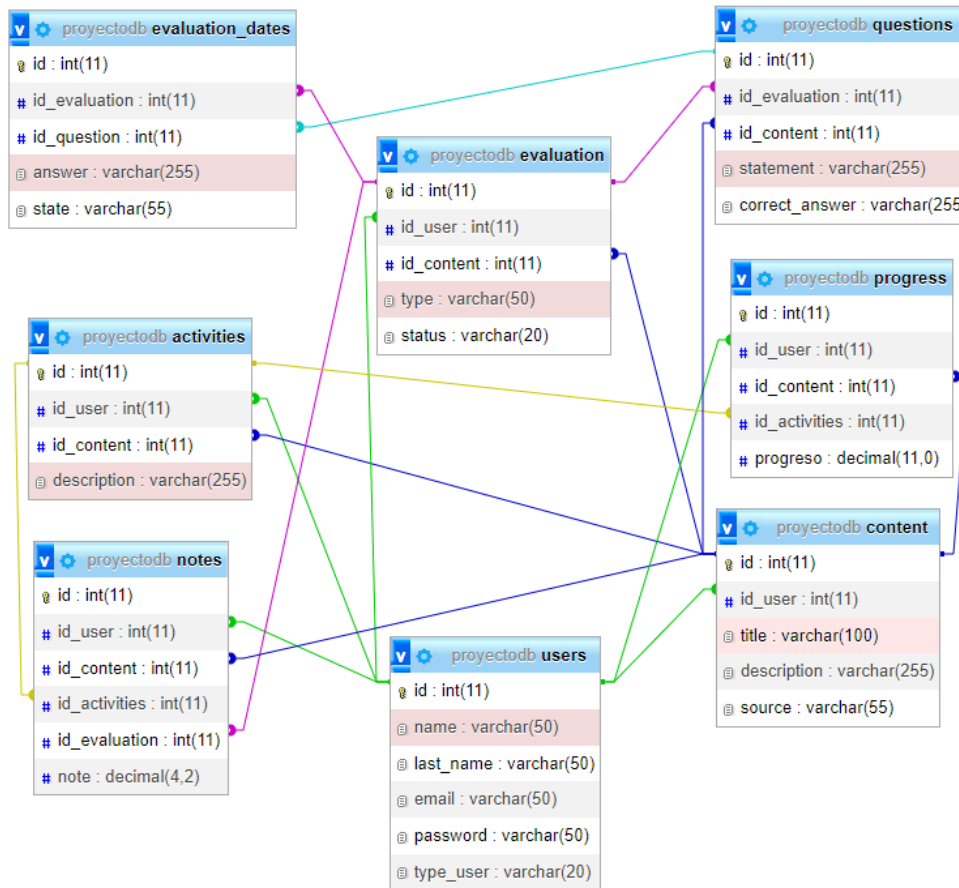


Ilustración 2 (Vista diseñador)

12. DESARROLLO DE LA BASE DE DATOS

La base de datos recibió las modificaciones necesarias para la introducción y relación de los datos, la disminución de entidades y la redundancia, la cual se encuentra distribuida de la siguiente manera:

12.1. ENTIDAD ACTIVIDADES. Es la encargada de almacenar las actividades, contiene un id como llave primaria, una descripción de cada actividad, el id usuario para almacenar los usuarios que realizan las actividades y el id contenido al que pertenece cada actividad, ambas llaves foráneas. Los datos se encuentran en la [ilustración 4](#).




#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
1	id 	int(11)			No	Ninguna
2	id_user 	int(11)			Sí	NULL
3	id_content 	int(11)			Sí	NULL
4	description	varchar(255)	utf8mb4_general_ci		No	Ninguna

Ilustración 3 (Atributos de las actividades)

id	id_user	id_content	description
1	7	7	Conceptos básicos HTML
2	2	2	Aprendiendo a programar con JavaScript
3	3	3	Creación de formularios HTML
4	4	4	Manipulación del DOM con JavaScript
5	5	5	Introducción a CSS para estilizar páginas web
6	6	6	Desarrollo de páginas web responsivas con CSS
7	7	1	Integración de imágenes y multimedia en páginas we...
8	8	8	Uso de APIs y consumo de datos en JavaScript
9	9	9	Implementación de animaciones con CSS y JavaScript
10	10	10	Optimización de rendimiento en aplicaciones web
11	11	11	Pruebas y depuración de aplicaciones web con herra...

Ilustración 4 (Datos de las actividades)

12.2. ENTIDAD CONTENIDOS. Es la encargada de almacenar los contenidos con un id como llave primaria, el título, la descripción, la fuente del contenido, y el respectivo id usuario que accede a dicho contenido como foránea. Los datos se encuentran en la [ilustración 6](#).



#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
1	id 	int(11)			No	Ninguna
2	id_user 	int(11)			Sí	NULL
3	title	varchar(100)	utf8mb4_general_ci		Sí	NULL
4	description	varchar(255)	utf8mb4_general_ci		Sí	NULL
5	source	varchar(55)	utf8mb4_general_ci		No	Ninguna

Ilustración 5 (Atributos de los contenidos)

id	id_user	title	description	source
1	7	Videos Educativos	Son videos creados con el propósito de enseñar o i...	Plataformas de video
2	2	Documentos	Son archivos de texto que contienen información es...	Artículos de revistas científicas
3	3	Presentaciones	Son una serie de diapositivas que contienen inform...	SlideShare, Prezi
4	4	Imágenes	Son representaciones visuales de objetos, personas...	Freepik, Google Images
5	5	Audios	Son grabaciones de sonidos o voces.	Podcast, Plataformas de video
6	6	Archivos PDF	Son un tipo de archivo que permite la visualizació...	Recursos en línea de bibliotecas virtuales
7	7	Enlaces	Son direcciones web que llevan a una página o recu...	World Wide Web
8	8	Ejercicios	Son actividades diseñadas para practicar habilidad...	Educaplay
9	9	Cuestionarios	Son pruebas con preguntas y opciones de respuesta ...	Quizizz
10	10	Simulaciones	Son representaciones interactivas de situaciones o...	Recursos educativos digitales
11	11	Otros Contenidos	Pueden ser cualquier otro tipo de material educati...	Sitios web educativos externos
12	1	Administrador	El profesor que administra los contenidos	admin

Ilustración 6 (Datos de los contenidos)

12.3. ENTIDAD EVALUACIÓN. Encargada de almacenar los datos de las evaluaciones, contiene un id como llave primaria, el tipo de evaluación, el estado que define si fue aprobada o desaprobada, el id de usuario del cual realiza el examen y el id del contenido al que pertenece la evaluación como claves foráneas. Los datos están en la [ilustración 8](#).




#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
1	id 	int(11)			No	Ninguna
2	id_user 	int(11)			Sí	NULL
3	id_content 	int(11)			Sí	NULL
4	type	varchar(50)	utf8mb4_general_ci		Sí	NULL
5	status	varchar(20)	utf8mb4_general_ci		Sí	NULL

Ilustración 7 (Atributos evaluación)

id	id_user	id_content	type	status
1	7	7	Opción múltiple con única respuesta	Aprobado
2	2	2	Preguntas abiertas	Aprobado
3	3	3	Opciones múltiples	Aprobado
4	4	4	Opción múltiple con única respuesta	Aprobado
5	5	5	Preguntas abiertas	Desaprobado
6	6	6	Opciones múltiples	Aprobado
7	7	1	Opción múltiple con única respuesta	Aprobado
8	8	8	Preguntas abiertas	Desaprobado
9	9	9	Opciones múltiples	Aprobado
10	10	10	Opción múltiple con única respuesta	Aprobado
11	11	11	Opción múltiple con única respuesta	Aprobado

Ilustración 8 (Datos de la evaluación)

12.4. ENTIDAD USUARIOS. Es la entidad encargada de almacenar a los usuarios con su respectivo id como llave primaria, el nombre, apellido, email y contraseña, y el tipo de usuario en el cual se define si es un docente o un estudiante. Los datos están en la [ilustración 10](#).


#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
1	id 	int(11)			No	Ninguna
2	name	varchar(50)	utf8mb4_general_ci		Sí	NULL
3	last_name	varchar(50)	utf8mb4_general_ci		Sí	NULL
4	email	varchar(50)	utf8mb4_general_ci		Sí	NULL
5	password	varchar(50)	utf8mb4_general_ci		Sí	NULL
6	type_user	varchar(20)	utf8mb4_general_ci		Sí	NULL

Ilustración 9 (Atributos del usuario)

id	name	last_name	email	password	type_user
1	Juan	Sin Miedo	juansinmiedo@outlook.com	00912	Docente
2	Maria	Varilla	maryvar@hotmail.com	09888	Estudiante
3	Pedro	López	lopezpedro@yopmail.com	34999	Estudiante
4	Ana	González	anag@hotmail.com	39912	Estudiante
5	Carlos	Martínez	carlosm@yahoo.es	99389	Estudiante
6	Laura	Rodríguez	rogrilau@gmail.com	32948	Estudiante
7	Luis	Pérez	luisp2219@gmail.com	98832	Estudiante
8	Sofía	Hernández	sofy1234@yopmail.com	38287	Estudiante
9	Diego	Sánchez	diegosanchez1999@yahoo.es	23482	Estudiante
10	Isabella	Torres	torresisa@outlook.com	39293	Estudiante
11	Camila	Solano	camisolano3000@gmail.com	39938	Estudiante

Ilustración 10 (Datos del usuario)

12.5. ENTIDAD DATOS DE EVALUACIÓN. Es la encargada de almacenar las respuestas de las evaluaciones con su respectiva id como foránea al igual que el id de las preguntas. Esta entidad posee un id como llave primaria, la respuesta y el estado que define si respondió correctamente. Ver datos en la [ilustración 12](#).

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
1	id 	int(11)			No	Ninguna
2	id_evaluation 	int(11)			Sí	NULL
3	id_question 	int(11)			Sí	NULL
4	answer	varchar(255)	utf8mb4_general_ci		Sí	NULL
5	state	varchar(55)	utf8mb4_general_ci		No	Ninguna

Ilustración 11 (Atributos Datos de evaluación)

id	id_evaluation	id_question	answer	state
1	1	1	Opción A	correcto
2	1	2	Opción C	correcto
3	1	3	Opción B	correcto
4	1	4	Opción D	correcto
5	1	5	Opción A	correcto
6	1	6	Opción C	correcto
7	1	7	Opción B	correcto
8	1	8	Opción D	correcto
9	1	9	Opción A	correcto
10	1	10	Opción C	correcto
11	3	11	Opción A	correcto
12	3	12	Opción C	correcto
13	3	13	Opción B	correcto
14	3	14	Opción D	correcto
15	3	15	Opción A	correcto
16	3	16	Opción C	correcto
17	3	17	Opción B	correcto
18	3	18	Opción D	correcto
19	3	19	Opción A	correcto
20	3	20	Opción C	correcto
21	4	21	Opción B	incorrecto
22	4	22	Opción C	correcto
23	4	23	Opción B	incorrecto
24	4	24	Opción D	correcto
25	4	25	Opción A	correcto

Ilustración 12 (Datos de las evaluaciones realizadas 25 de 95)

12.6. ENTIDAD NOTAS. Es la entidad encargada de almacenar las notas, ya sea de las actividades o las evaluaciones, ambas como llaves foráneas, asimismo con el id del usuario y el id del respectivo contenido. Los datos se encuentran en la [ilustración 14](#).

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
1	id 🔑	int(11)			No	<i>Ninguna</i>
2	id_user 🔑	int(11)			Sí	<i>NULL</i>
3	id_content 🔑	int(11)			Sí	<i>NULL</i>
4	id_activities 🔑	int(11)			Sí	<i>NULL</i>
5	id_evaluation 🔑	int(11)			Sí	<i>NULL</i>
6	note	decimal(4,2)			Sí	<i>NULL</i>

Ilustración 13 (Atributos de las notas)

id	id_user	id_content	id_activities	id_evaluation	note
1	7	7	NULL	1	5.00
2	2	2	NULL	2	4.60
3	3	3	NULL	3	5.00
4	4	4	NULL	4	4.00
5	5	5	NULL	5	2.50
6	6	6	NULL	6	3.75
7	7	1	NULL	7	3.90
8	8	8	NULL	8	2.85
9	9	9	NULL	9	3.90
10	10	10	NULL	10	4.20
11	11	11	NULL	11	4.10
12	7	1	1	NULL	5.00
13	2	2	2	NULL	4.75
14	3	3	3	NULL	3.11
15	4	4	4	NULL	4.45
16	5	5	5	NULL	2.10
17	6	6	6	NULL	4.90
18	7	7	7	NULL	3.79
19	8	8	8	NULL	2.75
20	9	9	9	NULL	4.60
21	10	10	10	NULL	4.30
22	11	11	11	NULL	4.85
23	1	12	NULL	NULL	NULL

Ilustración 14 (Datos de las notas)

12.7. ENTIDAD PROGRESO. Es la encargada de almacenar el progreso de las actividades por contenido realizadas por el estudiante todos con sus respectivas id de llaves foráneas, y el respectivo progreso donde se coloca un porcentaje del contenido trabajado.

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
1	id 	int(11)			No	Ninguna
2	id_user 	int(11)			Sí	NULL
3	id_content 	int(11)			Sí	NULL
4	id_activities 	int(11)			Sí	NULL
5	progreso	decimal(11,0)			No	Ninguna

Ilustración 15 (Atributos del progreso)

Aquí los respectivos datos

id	id_user	id_content	id_activities	progreso
1	7	7	7	100
2	2	2	2	73
3	3	3	3	50
4	4	4	4	87
5	5	5	5	20
6	6	6	6	95
7	7	1	1	68
8	8	8	8	33
9	9	9	9	79
10	10	10	10	62
11	11	11	11	91

Ilustración 16 (Datos de los progresos)

12.8. ENTIDAD PREGUNTAS. Es la encargada de almacenar las preguntas que se realizan en la evaluación, esta entidad almacena el enunciado y la respuesta correcta de la pregunta e incluye las llaves foráneas de la evaluación y del respectivo contenido. Los datos se encuentran en la [ilustración 18](#).

#	Nombre	Tipo	Cotejamiento	Atributos	Nulo	Predeterminado
1	id 	int(11)			No	Ninguna
2	id_evaluation 	int(11)			Sí	NULL
3	id_content 	int(11)			Sí	NULL
4	statement	varchar(255)	utf8mb4_general_ci		Sí	NULL
5	correct_answer	varchar(255)	utf8mb4_general_ci		Sí	NULL

Ilustración 17 (Atributos de las preguntas)

id	id_evaluation	id_content	statement	correct_answer
1	1	1	Pregunta 1	Opción A
2	1	1	Pregunta 2	Opción C
3	1	1	Pregunta 3	Opción B
4	1	1	Pregunta 4	Opción D
5	1	1	Pregunta 5	Opción A
6	1	1	Pregunta 6	Opción C
7	1	1	Pregunta 7	Opción B
8	1	1	Pregunta 8	Opción D
9	1	1	Pregunta 9	Opción A
10	1	1	Pregunta 10	Opción C
11	3	3	Pregunta 11	Opción A
12	3	3	Pregunta 12	Opción C
13	3	3	Pregunta 13	Opción B
14	3	3	Pregunta 14	Opción D
15	3	3	Pregunta 15	Opción A
16	3	3	Pregunta 16	Opción C
17	3	3	Pregunta 17	Opción B
18	3	3	Pregunta 18	Opción D
19	3	3	Pregunta 19	Opción A
20	3	3	Pregunta 20	Opción C
21	4	4	Pregunta 21	Opción B
22	4	4	Pregunta 22	Opción C
23	4	4	Pregunta 23	Opción B
24	4	4	Pregunta 24	Opción D
25	4	4	Pregunta 25	Opción A

Ilustración 18 (Datos de las preguntas 25 de 95)

BASES DE DATOS NO RELACIONALES (NOSQL)

Las bases de datos NoSQL son un tipo de base de datos que no utiliza el modelo relacional tradicional. En cambio, las bases de datos NoSQL utilizan una variedad de modelos de datos, como documento, clave-valor, columna ancha y gráfico.

MongoDB es una base de datos de documentos, lo que significa que los datos se almacenan en documentos. Los documentos son similares a los objetos JSON y pueden contener una variedad de tipos de datos, como cadenas, números, matrices y objetos. MongoDB es una base de datos NoSQL popular porque es flexible, escalable y fácil de usar. También es adecuado para una variedad de aplicaciones, como aplicaciones web, análisis de Big data y aplicaciones en tiempo real.

Estos son algunos de los beneficios de usar bases de datos NoSQL:

MODELOS DE DATOS FLEXIBLES

Las bases de datos NoSQL le permiten almacenar datos de diversas formas, lo que puede ser útil para aplicaciones con requisitos de datos complejos.

ESCALABILIDAD

Las bases de datos NoSQL se pueden escalar horizontalmente, lo que significa que puede agregar más servidores para aumentar la capacidad de su base de datos.

RENDIMIENTO

Las bases de datos NoSQL pueden ser muy rápidas, especialmente para ciertos tipos de consultas.

FACILIDAD DE USO

Las bases de datos NoSQL suelen ser más fáciles de usar que las bases de datos relacionales tradicionales.

Aquí hay algunos ejemplos de aplicaciones que usan MongoDB:

APLICACIONES WEB

MongoDB se usa a menudo para almacenar datos para aplicaciones web, como perfiles de usuario, catálogos de productos e información de pedidos.

ANÁLISIS DE BIG DATA

MongoDB se puede utilizar para almacenar y analizar grandes cantidades de datos. Esto puede ser útil para las aplicaciones que necesitan realizar un seguimiento del comportamiento de los usuarios, analizar las tendencias del mercado o realizar otros tipos de análisis de datos.

APLICACIONES EN TIEMPO REAL

MongoDB se puede usar para crear aplicaciones en tiempo real que necesitan procesar datos rápidamente. Esto puede ser útil para aplicaciones como aplicaciones de chat, sistemas de negociación de acciones y aplicaciones de juegos.

A continuación, se apreciará el modelo NOSQL que se desarrolló para las necesidades del sistema. En mi primera instancia, se creó una colección llamada “users”, en la colección "users", el campo "activities" contiene una lista de objetos que tienen una relación con la colección "activities" a través del campo "_id". Cada objeto en la lista representa una actividad relacionada con el usuario. La colección "users" también tiene un campo "evaluation" que contiene una lista de objetos que tienen una relación con la colección "evaluations" a través del campo "_id". Cada objeto en la lista representa una evaluación relacionada con el usuario ([Ver ilustración 19](#)).

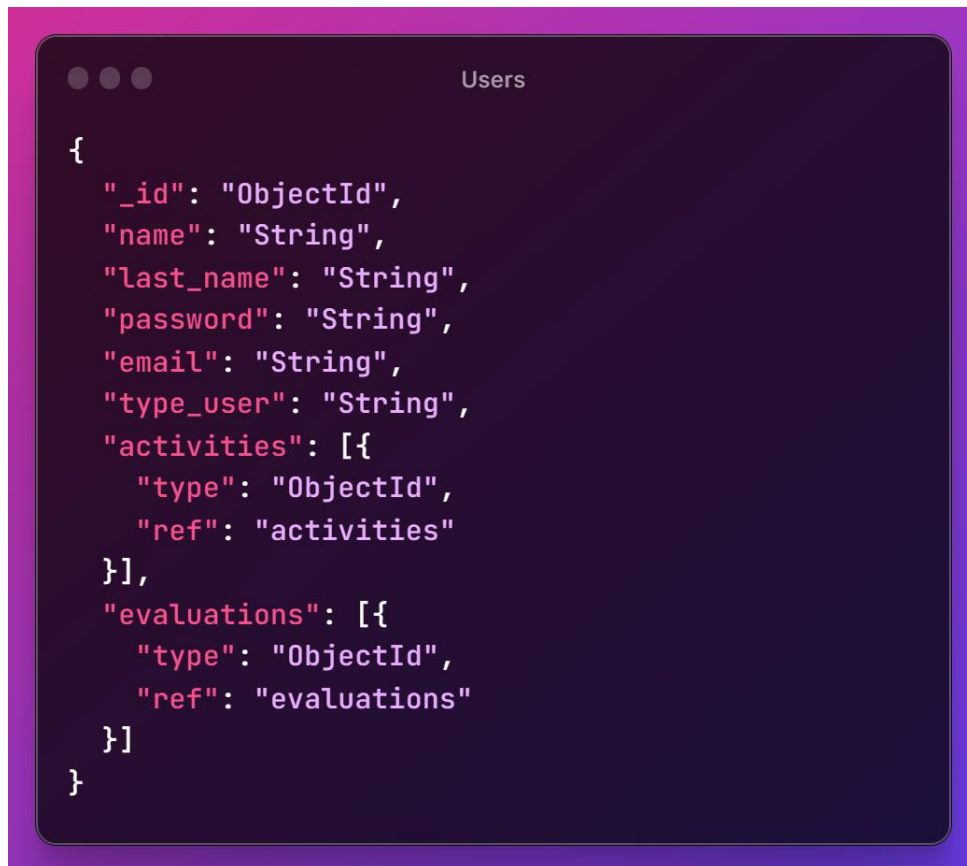


Ilustración 19 (Colección users)

Luego, se encuentra la colección “evaluations”, la colección "evaluations" tiene un campo "id_content" que se relaciona con la colección "contents" a través del campo "_id". Esto indica que cada evaluación está asociada a un contenido específico. La colección "evaluations" también tiene un campo "id_user" que se relaciona con la colección "users" a través del campo "_id". Esto indica que cada evaluación está asociada a un usuario específico ([Ver ilustración 20](#)).

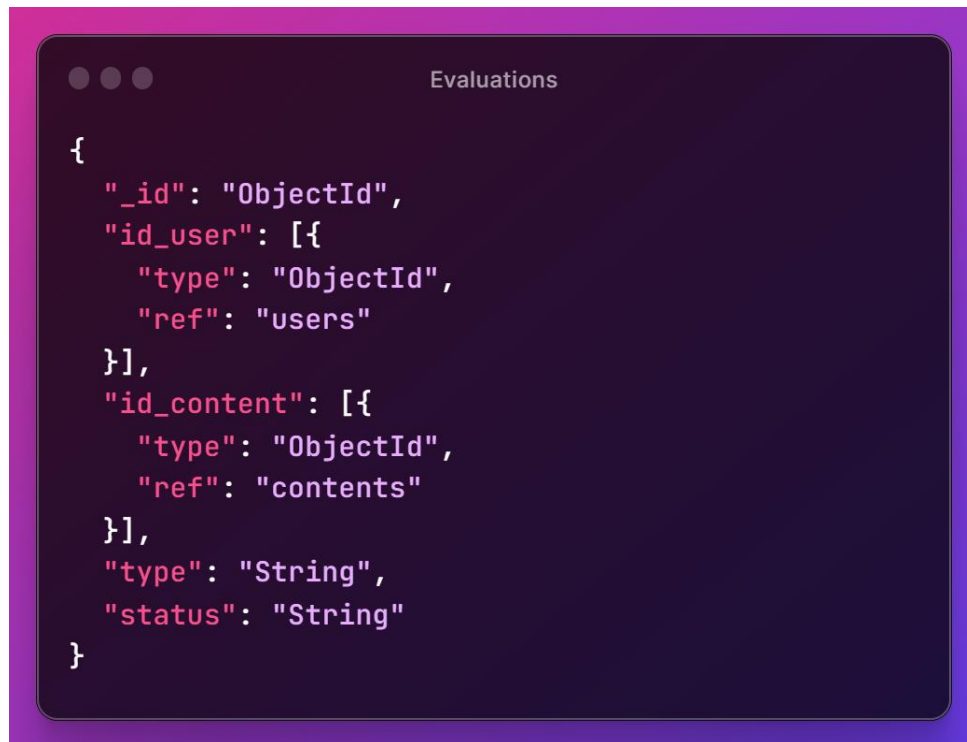


Ilustración 20 (Colección evaluations)

En la colección "contents", el campo "_id" es utilizado como identificador único para cada contenido. Este campo es referenciado en otras colecciones a través de los campos "id_content" ([Ver ilustración 21](#)).

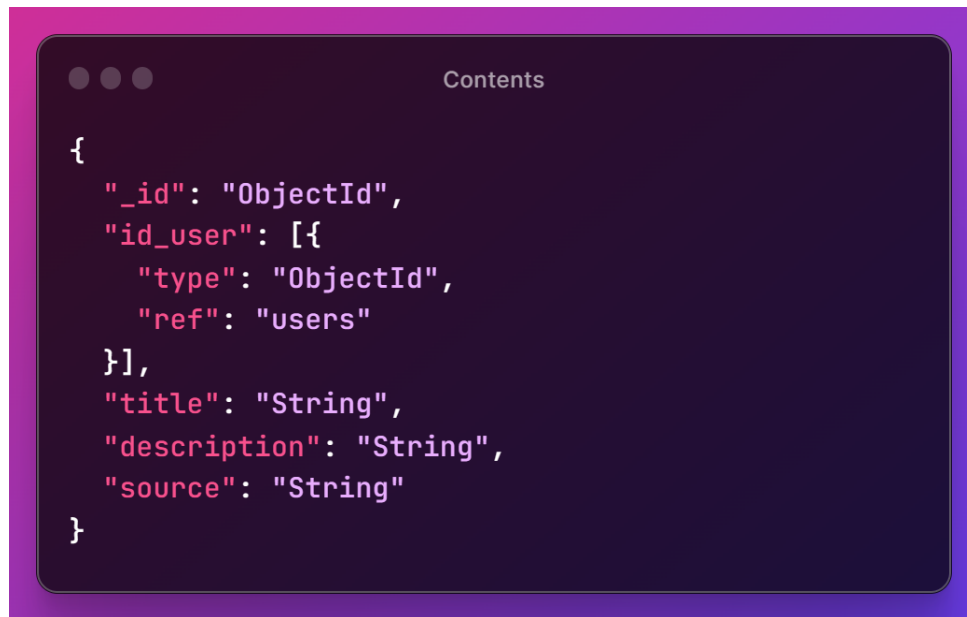


Ilustración 21 (Colección contents)

A continuación, se encuentra la colección “activities”, la colección "activities" tiene un campo "id_user" que se relaciona con la colección "users" a través del campo "_id". Esto indica que cada actividad está asociada a un usuario específico. La colección "activities" también tiene un campo "id_content" que se relaciona con la colección "contents" a través del campo "_id". Esto indica que cada actividad está asociada a un contenido específico ([Ver ilustración 22](#)).

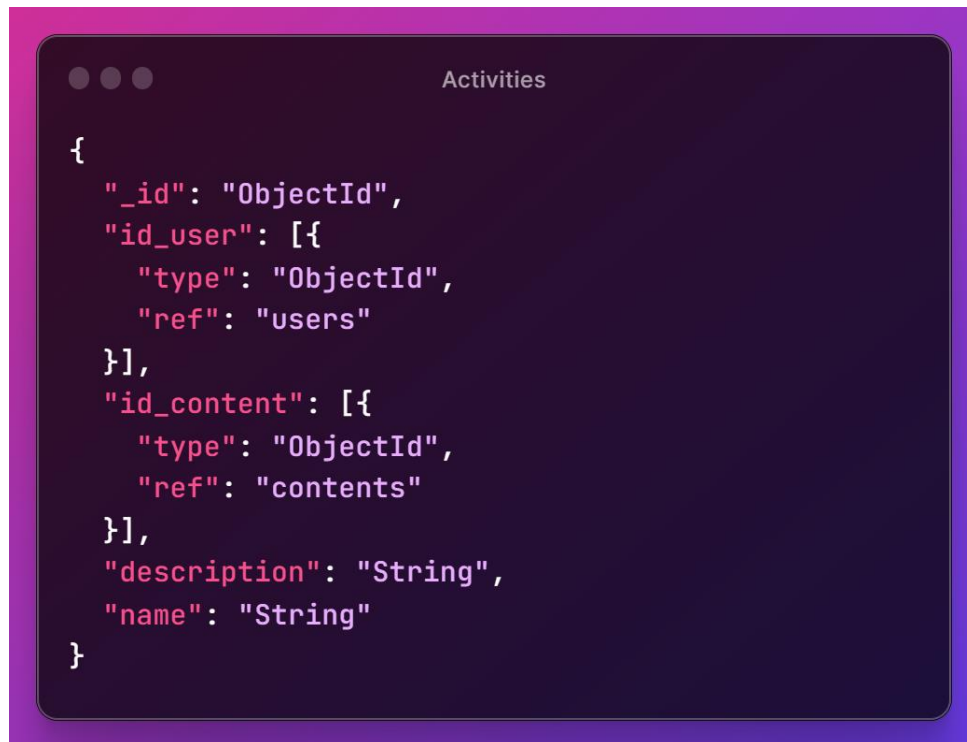


Ilustración 22 (Colección activities)

La colección "progress" almacena los registros de progreso de los contenidos y actividades para usuarios específicos, en las que el "_id" representa el identificador único del registro de progreso. Tiene relaciones con el "id_user" se refiere al identificador del usuario asociado al progreso, "id_content" al identificador del contenido al que se está realizando el progreso e "id_activities" que se refiere al identificador de las actividades relacionadas al progreso, todas estableciendo relación con las colecciones correspondientes a través del campo "_id", lo que indica a qué usuarios, actividades y contenidos está asociado el progreso ([Ver ilustración 23](#)).

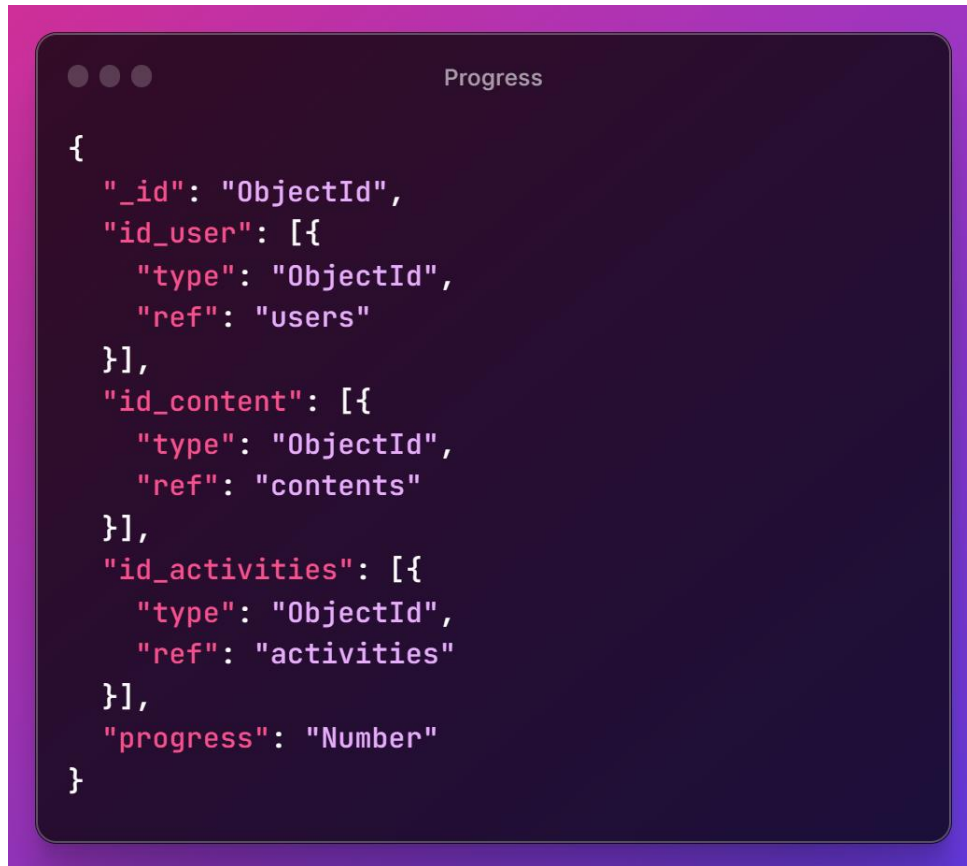


Ilustración 23 (Colección progress)

La colección "evaluationdatas" almacena los datos de evaluaciones, y tiene relaciones con "id_evaluation" en la colección "evaluations" a través del campo "_id" lo que indica que cada registro en "evaluationdatas" está asociado a una evaluación específica en la colección "evaluations", y también "id_question" en la colección "questions" a través del campo "_id" lo cual indica que cada registro en "evaluationdatas" está asociado a una pregunta específica en la colección "questions" ([Ver ilustración 24](#)).

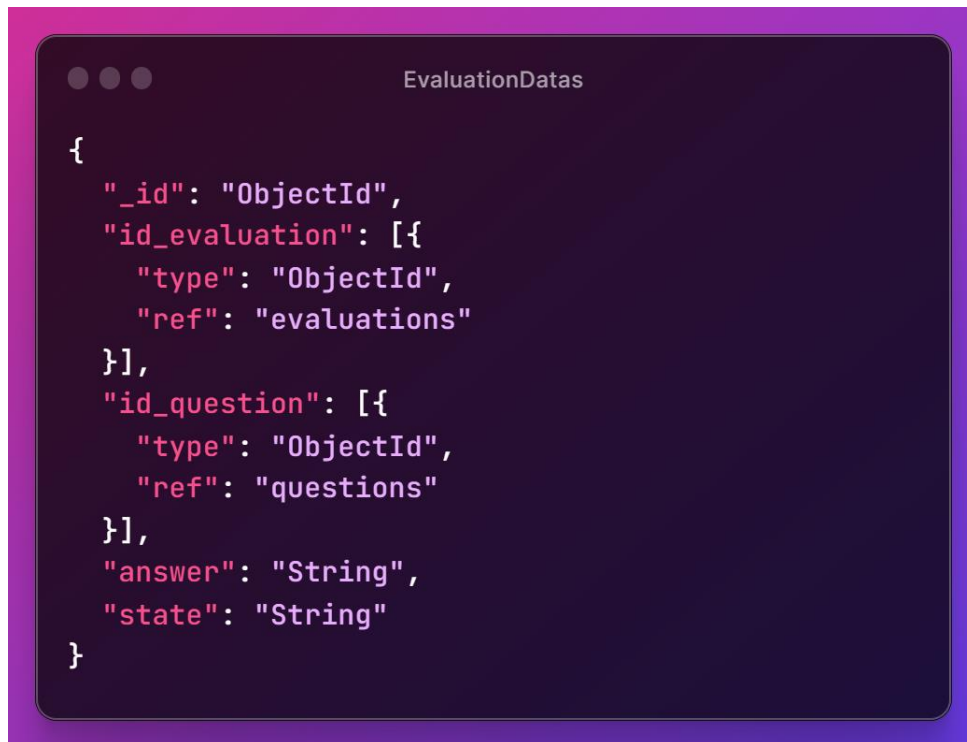


Ilustración 24 (Colección evaluationdatas)

En la colección “questions”, los atributos que representan las relaciones son “id_evaluation” hace referencia a la evaluación a la que pertenece la pregunta, utilizando el campo “_id” de la colección “Evaluations” e “id_content” que referencia al contenido asociado a la pregunta en el campo “_id” de la colección “Contents” ([Ver ilustración 25](#)).

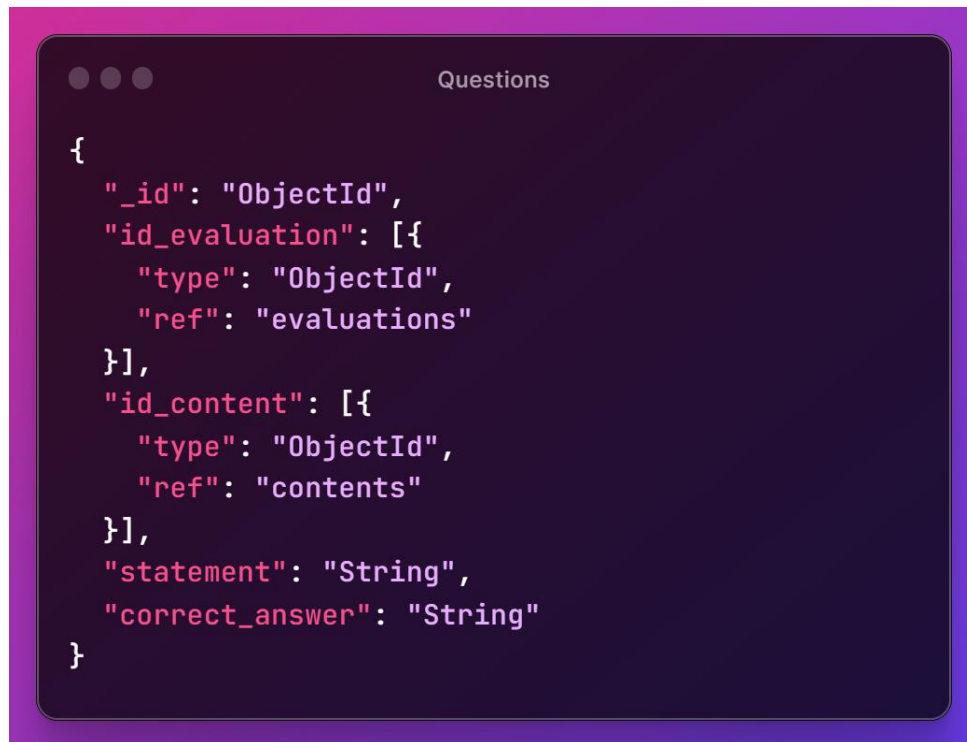


Ilustración 25 (Colección questions)

Finalmente, en la colección “notes”, los campos "id_user", "id_content", "id_activities" y "id_evaluation" contienen objetos con las propiedades "type" y "ref". Esto indica las relaciones entre las notas con las respectivas colecciones de usuarios, contenidos, actividades y evaluaciones ([Ver ilustración 26](#)).



```
{
  "_id": "ObjectId",
  "id_user": [{
    "type": "ObjectId",
    "ref": "users"
  }],
  "id_content": [{
    "type": "ObjectId",
    "ref": "contents"
  }],
  "id_activities": [{
    "type": "ObjectId",
    "ref": "activities"
  }],
  "id_evaluation": [{
    "type": "ObjectId",
    "ref": "evaluations"
  }],
  "note": "Number"
}
```

Ilustración 26 (Colección notes)

APIS

Las API, o interfaces de programación de aplicaciones, son una forma en que dos piezas de software se comunican entre sí. Son un conjunto de definiciones y protocolos que permiten que el software acceda e interactúe entre sí. Las API se utilizan para compartir datos, funciones y servicios entre diferentes aplicaciones.

Estos son algunos ejemplos de API:

La API de Google Maps permite a los desarrolladores acceder y mostrar datos de Google Maps en sus propias aplicaciones.

La API de Twitter permite a los desarrolladores acceder e interactuar con datos de Twitter, como tweets, usuarios y tendencias.

La API de Facebook permite a los desarrolladores acceder e interactuar con datos de Facebook, como perfiles de usuario, fotos y eventos.

Las API se utilizan en una amplia variedad de aplicaciones, que incluyen:

APLICACIONES WEB

Las API se utilizan para potenciar muchas de las funciones de las aplicaciones web, como la búsqueda, los mapas y la integración de redes sociales.

APLICACIONES MÓVILES

Las API se utilizan para potenciar muchas de las funciones de las aplicaciones móviles, como los servicios basados en la ubicación, los pagos y la integración de redes sociales.

SISTEMAS BACKEND

Las API se utilizan para conectar sistemas backend, como bases de datos y servidores, a otras aplicaciones.

BENEFICIOS

Las API son una herramienta poderosa que se puede utilizar para integrar diferentes aplicaciones y servicios. Son una parte clave de la web moderna y son utilizados por muchas de las aplicaciones y servicios más populares.

Estos son algunos de los beneficios de usar API:

Reusabilidad: las API pueden ser reutilizadas por varias aplicaciones, lo que puede ahorrar tiempo y esfuerzo de desarrollo.

Escalabilidad: las API se pueden escalar para manejar más tráfico, lo que puede ayudar a que las aplicaciones crezcan.

Seguridad: las API se pueden usar para proteger datos y servicios, lo que puede ayudar a proteger las aplicaciones del acceso no autorizado.

INCONVENIENTES

Estos son algunos de los inconvenientes de usar API:

Complejidad: Las API pueden ser complejas de desarrollar y usar, lo que puede ser una barrera de entrada para algunos desarrolladores.

Seguridad: las API pueden ser un riesgo de seguridad si no se implementan correctamente.

Costo: Las API pueden ser costosas de usar, si no son gratuitas.

EJEMPLO PRÁCTICO DE LA IMPLEMENTACIÓN DE UNA API CON NODEJS Y EXPRESS.JS

```
const express = require('express');
const app = express();
const users = require('../data.json');
const port = 3000;

// App.use para que el servidor pueda leer el archivo data.json
app.use(express.json());
// Método GET con un texto plano
app.get('/', (req, res) => {
  res.send('Hello World!');
});

// Método GET con un objeto JSON
app.get('/local-users', (req, res) => {
  res.json(users);
});

// Método GET para obtener un usuario por su id
app.get('/local-users/:id', (req, res) => {
  const id = req.params.id;
  const user = users.find(user => user.id == id);
  if (!user) {
```



```

        res.status(404).send('No se puede mostrar el usuario con el id: ' + id +
' porque no existe');
    } else {
        res.json(user);
    }
});
// Método POST para crear un nuevo usuario
app.post('/local-users', (req, res) => {
    const newUser = req.body;
    users.push(newUser);
    res.json(newUser);
});
// Método PUT para actualizar un usuario
app.put('/local-users/:id', (req, res) => {
    const id = req.params.id;
    const updateUser = req.body;
    const index = users.findIndex(user => user.id == id);
    if (index === -1) {
        res.status(404).send('El id ' + id + ' de este usuario no fue encontrado
o no existe');
    } else {
        users[index] = updateUser;
        res.json(updateUser);
    }
});
// Método DELETE para eliminar un usuario
app.delete('/local-users/:id', (req, res) => {
    const id = req.params.id;
    const index = users.findIndex(user => user.id == id);
    if (index === -1) {
        res.status(404).send('No se puede borrar el usuario con el id: ' + id +
' porque ya fue eliminado o no existe');
    } else {
        const deletedUser = users.splice(index, 1);
        res.json(deletedUser[0]);
    }
});
// Listen para que el servidor escuche en el puerto 3000
app.listen(port, () => {
    console.log(`Servidor corriendo en el puerto: ${port}`);
});

```

El código anterior corresponde a una API de manejo de usuarios, la cual permite CREAR, OBTENER, ACTUALIZAR y ELIMINAR usuarios mediante peticiones HTTP, de igual forma, los usuarios son obtenidos desde un archivo local llamado “data.json”, el cual simula una base de datos, a continuación, el código de dicho archivo.

```
[
  {
    "id": 1, "nombre": "Juan", "apellido": "Perez", "edad": 30,
    "fechaNacimiento": "1989-01-01", "sexo": "M"
  },
  {
    "id": 2, "nombre": "Maria", "apellido": "Gomez", "edad": 25,
    "fechaNacimiento": "1994-01-01", "sexo": "F"
  },
  {
    "id": 3, "nombre": "Pedro", "apellido": "Gonzalez", "edad": 35,
    "fechaNacimiento": "1984-01-01", "sexo": "M"
  }
]
```

Este código es un arreglo de objetos, el cual almacena información sobre unos usuarios, como su nombre, apellido, etc.

A continuación, se explicará cada fragmento de código de la API que manipula estos datos.

```
const express = require('express');
const app = express();
const users = require('../data.json');
const port = 3000;
```

const express = require('express'); Importa el módulo 'express' que se encuentra instalado en el proyecto. El módulo 'express' es una biblioteca de Node.js que facilita la creación de servidores web y el manejo de rutas y solicitudes.

const app = express(); Crea una instancia de la aplicación Express. app será utilizado para configurar las rutas y definir el comportamiento de la aplicación web.

const users = require('../data.json'); Importa los datos del archivo 'data.json'. Los datos se asignan a la variable users para su posterior uso en las rutas.

const port = 3000; Define el número de puerto en el que el servidor web escuchará las solicitudes entrantes. En este caso, el puerto se ha configurado como 3000.

```
app.use(express.json());
app.get('/', (req, res) => {
  res.send('Hello World!');
});
app.get('/local-users', (req, res) => {
  res.json(users);
});
app.get('/local-users/:id', (req, res) => {
  const id = req.params.id;
  const user = users.find(user => user.id == id);
  if (!user) {
    res.status(404).send('No se puede mostrar el usuario con el id: ' + id +
' porque no existe');
  } else {
    res.json(user);
  }
});
```

Este nuevo fragmento de código define varias rutas y su correspondiente manejo de solicitudes en la aplicación Express. A continuación, se explica el propósito de cada bloque de código:

app.use(express.json()); Este middleware permite que la aplicación Express pueda analizar los datos enviados en formato JSON en las solicitudes entrantes. Esto es útil cuando se reciben datos JSON en las solicitudes POST, PUT o PATCH.

app.get('/', (req, res) => { res.send('Hello World!'); }); Esta ruta define un manejador para la ruta raíz ("/"). Cuando se recibe una solicitud GET a esta ruta, se envía una respuesta con el texto "Hello World!".

app.get('/local-users', (req, res) => { res.json(users); }); Esta ruta maneja las solicitudes GET a la ruta "/local-users". Cuando se recibe una solicitud GET a esta ruta, se envía una respuesta con los datos users en formato JSON. Los datos provienen del archivo JSON importado anteriormente.

app.get('/local-users/:id', (req, res) => { ... }); Esta ruta maneja las solicitudes GET a la ruta `"/local-users/:id"`, donde `":id"` es un parámetro dinámico que representa el ID de un usuario. Cuando se recibe una solicitud GET a esta ruta, se busca en el array `users` el usuario correspondiente al ID proporcionado. Si se encuentra, se envía una respuesta con los datos del usuario en formato JSON. Si no se encuentra el usuario, se envía una respuesta con un código de estado 404 y un mensaje indicando que el usuario no existe.

```
app.post('/local-users', (req, res) => {  
  const newUser = req.body;  
  users.push(newUser);  
  res.json(newUser);  
});
```

Este bloque de código define una ruta y su manejo para las solicitudes POST a la ruta `"/local-users"` en la aplicación Express. A continuación, se explica lo que hace cada línea:

app.post('/local-users', (req, res) => { ... }); Esta línea establece una ruta para las solicitudes POST a la ruta `"/local-users"`. Cuando se recibe una solicitud POST a esta ruta, se ejecuta la función de manejo especificada.

const newUser = req.body; Esta línea obtiene el cuerpo de la solicitud (request body) utilizando `req.body`. Se asume que la solicitud POST incluye datos en formato JSON que representan un nuevo usuario. El cuerpo de la solicitud se asigna a la variable `newUser`.

users.push(newUser); Esta línea agrega el nuevo usuario (`newUser`) al array `users`. Se asume que `users` es una variable previamente definida que contiene una lista de usuarios.

res.json(newUser); Esta línea envía una respuesta al cliente en formato JSON con los datos del nuevo usuario que se acaba de agregar. La respuesta contiene los mismos datos que se recibieron en la solicitud POST.

```

app.put('/local-users/:id', (req, res) => {
  const id = req.params.id;
  const updateUser = req.body;
  const index = users.findIndex(user => user.id == id);
  if (index === -1) {
    res.status(404).send('El id ' + id + ' de este usuario no fue encontrado
o no existe');
  } else {
    users[index] = updateUser;
    res.json(updateUser);
  }
});

```

Este bloque de código define una ruta y su manejo para las solicitudes PUT a la ruta `"/local-users/:id"` en la aplicación Express. Aquí se explica lo que hace cada línea:

app.put('/local-users/:id', (req, res) => { ... }); Esta línea establece una ruta para las solicitudes PUT a la ruta `"/local-users/:id"`. El parámetro `:id` es un parámetro de ruta dinámico que representa el ID del usuario a actualizar. Cuando se recibe una solicitud PUT a esta ruta, se ejecuta la función de manejo especificada.

const id = req.params.id; Esta línea obtiene el valor del parámetro `id` de la ruta utilizando `req.params.id`. Esto permite acceder al ID del usuario que se desea actualizar.

const updateUser = req.body; Esta línea obtiene el cuerpo de la solicitud (request body) utilizando `req.body`. Se asume que la solicitud PUT incluye datos en formato JSON que representan la actualización de un usuario existente. El cuerpo de la solicitud se asigna a la variable `updateUser`.

const index = users.findIndex(user => user.id == id); Esta línea busca en el array `users` el índice del usuario que tiene el ID coincidente con el valor obtenido en el paso anterior. Utiliza el método `findIndex` de JavaScript para buscar el usuario correspondiente.

if (index === -1) { ... } else { ... } Esta estructura condicional verifica si se encontró el usuario en el array. Si el índice es igual a `-1`, significa que no se encontró el usuario y se envía una respuesta con el código de estado 404 y un mensaje indicando que el usuario no existe. De lo contrario, se procede a actualizar el usuario.

users[index] = updateUser; Esta línea actualiza el usuario existente en el array users con los datos del objeto updateUser. Se reemplaza el usuario existente en el índice encontrado con el objeto de actualización.

res.json(updateUser); Esta línea envía una respuesta al cliente en formato JSON con los datos del usuario actualizado. La respuesta contiene los mismos datos que se recibieron en la solicitud PUT.

```
app.delete('/local-users/:id', (req, res) => {  
  const id = req.params.id;  
  const index = users.findIndex(user => user.id == id);  
  if (index === -1) {  
    res.status(404).send('No se puede borrar el usuario con el id: ' + id +  
    ' porque ya fue eliminado o no existe');  
  } else {  
    const deletedUser = users.splice(index, 1);  
    res.json(deletedUser[0]);  
  }  
});
```

Este bloque de código define una ruta y su manejo para las solicitudes DELETE a la ruta "/local-users/:id" en la aplicación Express. A continuación, se explica lo que hace cada línea:

app.delete('/local-users/:id', (req, res) => { ... }); Esta línea establece una ruta para las solicitudes DELETE a la ruta "/local-users/:id". El parámetro :id es un parámetro de ruta dinámico que representa el ID del usuario a eliminar. Cuando se recibe una solicitud DELETE a esta ruta, se ejecuta la función de manejo especificada.

const id = req.params.id; Esta línea obtiene el valor del parámetro id de la ruta utilizando req.params.id. Esto permite acceder al ID del usuario que se desea eliminar.

const index = users.findIndex(user => user.id == id); Esta línea busca en el array users el índice del usuario que tiene el ID coincidente con el valor obtenido en el paso anterior. Utiliza el método findIndex de JavaScript para buscar el usuario correspondiente.

if (index === -1) { ... } else { ... } Esta estructura condicional verifica si se encontró el usuario en el array. Si el índice es igual a -1, significa que no se encontró el usuario y se envía una respuesta con el código de estado 404 y un mensaje indicando que el usuario ya fue eliminado o no existe. De lo contrario, se procede a eliminar el usuario.

const deletedUser = users.splice(index, 1); Esta línea utiliza el método splice de JavaScript para eliminar el usuario del array users en el índice encontrado. El método splice modifica el array original y devuelve un nuevo array con los elementos eliminados. En este caso, se especifica 1 como segundo parámetro para indicar que se eliminará un único elemento.

res.json(deletedUser[0]); Esta línea envía una respuesta al cliente en formato JSON con los datos del usuario eliminado. La respuesta contiene los datos del usuario eliminado que se obtuvieron del array deletedUser. Dado que splice devuelve un array con los elementos eliminados, se accede al primer elemento [0] para obtener el objeto de usuario eliminado.

```
app.listen(port, () => {  
  console.log(`Servidor corriendo en el puerto: ${port}`);  
});
```

Finalmente, este bloque de código inicia el servidor de la aplicación Express para escuchar las solicitudes entrantes en un puerto específico. A continuación, se explica lo que hace cada línea:

app.listen(port, () => { ... }); Esta línea inicia el servidor y lo pone en modo de escucha en un puerto específico. El parámetro port es el número de puerto en el que se desea que el servidor escuche las solicitudes entrantes. Cuando el servidor comienza a escuchar, se ejecuta la función de devolución de llamada especificada.

console.log(Servidor corriendo en el puerto: \${port}); Esta línea imprime un mensaje en la consola del servidor indicando que el servidor se ha iniciado correctamente y está escuchando en el puerto especificado. El mensaje muestra el número de puerto utilizando una interpolación de cadenas para insertar el valor de la variable port en el mensaje.

TABLA RUTAS AMIGABLES

MÉTODO	RUTA	DATOS DE ENTRADA	DATOS DE SALIDA	DESCRIPCIÓN
GET	/users		[{ "name": "String", "last_name": "String", "password": "String", "email": "String", "type_user": "String", "activities": "Array", "evaluations": "Array", }]	Esta ruta devuelve un array de objetos con la información de todos los usuarios de la base de datos.
GET	/users/:id	id	{ "name": "String", "last_name": "String", "password": "String", "email": "String", "type_user": "String", "activities": "Array", "evaluations": "Array", }	Esta ruta devuelve un objeto con la información de un usuario en específico almacenado en la base de datos.
POST	/users	{ "name": "String", "last_name": "String", "password": "String", "email": "String", "type_user": "String", "activities": "Array", "evaluations": "Array", }	Usuario creado exitosamente	Esta ruta crea un nuevo usuario en la base de datos con la información de entrada y devuelve como salida un mensaje de éxito.

PUT	/users/:id	id	Usuario actualizado exitosamente	Esta ruta actualiza un usuario en la base de datos con la ID correspondiente al usuario que se quiere actualiza y retorna como salida un mensaje de éxito.
DELETE	/users/		Todos los usuarios han sido eliminados	Esta ruta elimina todos los usuarios de la base de datos y trae de vuelta el mensaje de éxito.
DELETE	/users/:id	id	El usuario ha sido eliminado	Esta ruta elimina el usuario en la base de datos con el ID correspondiente al usuario que se quiere eliminar y devuelve el mensaje de eliminado con éxito.
GET	/activities		{ "__id_user": "ObjectId", "__id_content": "ObjectId", "description": "String", "name": "String", }	Esta ruta devuelve un array de objetos con la información de todas las actividades de la base de datos
GET	/activities/:id	id	{ id_user: ObjectId, id_content: ObjectId, description: String, name: String, }	Esta ruta devuelve una actividad en específico de las actividades que están almacenadas en la base de datos.
POST	/activities	{ "__id_user": "ObjectId", "__id_content": "ObjectId", "description": "String", "name": "String", }	Actividad creada exitosamente	Esta ruta crea una nueva actividad en la base de datos con la información de entrada y devuelve como salida un mensaje de éxito.
PUT	/activities/:id	ID	Actividad actualizada exitosamente	Esta ruta actualiza una actividad en la base de datos con la ID correspondiente de la actividad que se quiere actualizar y devuelve como salida un mensaje de éxito.

DELETE	/activities		Todas las actividades han sido eliminadas	Con la ruta anterior se borran todas las actividades de la base de datos.
DELETE	/activities/:id	ID	¡Éxito en eliminarActividad!	Esta ruta elimina una actividad en la base de datos con el ID correspondiente de la actividad que se quiere eliminar y devuelve el mensaje de eliminado con éxito.
GET	/content		{ "__id_user": "ObjectId", "__title": "String", "__description": "String", "__source": "String", }	Esta ruta devuelve un array de objetos con la información del apartado de los contenidos de la base de datos
GET	/content/:id	ID	{ "__id_user": "ObjectId", "__title": "String", "__description": "String", "__source": "String", }	Esta ruta devuelve un contenido en específico de los contenidos que están almacenados en la base de datos y la trae gracias al Id que es el que determina la identificación de los contenidos.
POST	/content	{ "__id_user": "ObjectId", "__title": "String", "__description": "String", "__source": "String", }	¡Éxito en crearContenido!	Esta ruta crea un nuevo contenido en la base de datos con la información de entrada y devuelve como salida un mensaje de éxito.
PUT	/content/:id	ID	¡Éxito en actualizarContenido!	Esta ruta actualiza un contenido en la base de datos con el ID correspondiente del contenido que se quiere

				actualizar y devuelve como salida un mensaje de éxito.
DELETE	/content		Todos los contenidos han sido eliminados	Esta ruta elimina todos los contenidos de la base de datos y devuelve un mensaje de éxito
DELETE	/content/:id	ID	El contenido ha sido eliminado	Esta ruta elimina el contenido en la base de datos con el ID correspondiente al contenido que se quiere eliminar y devuelve el mensaje de eliminado con éxito.
GET	/evaluation		[{ "__id_user": "ObjectId", "__id_content": "ObjectId", "type": "String", "status": "String", }]	Esta ruta devuelve un array de objetos con la información de todas las evaluaciones de la base de datos.
GET	/evaluation/:id	ID	{ "__id_user": "ObjectId", "__id_content": "ObjectId", "type": "String", "status": "String", }	Esta ruta devuelve un objeto con la información de una evaluación en específico.
POST	/evaluation	{ "__id_user": "ObjectId", "__id_content": "ObjectId", "type": "String", "status": "String", }	Evaluación creada exitosamente	Esta ruta crea una evaluación en la base de datos con la información de entrada y devuelve como salida un mensaje de éxito.
PUT	/evaluation/:id	ID	Evaluación actualizada exitosamente	Esta ruta actualiza una evaluación en la base de datos con el ID correspondiente a la evaluación que se quiere actualizar y devuelve como salida un mensaje de éxito.
DELETE	/evaluation		Todas las evaluaciones han sido eliminadas	Esta ruta elimina todas las evaluaciones de la base de datos.

DELETE	/evaluation/:id	ID	La evaluación ha sido eliminada	Esta ruta elimina la evaluación en la base de datos con el ID correspondiente a la evaluación que se quiere eliminar y devuelve el mensaje de eliminado con éxito.
GET	/progress		[{ "id_user": "ObjectId", "id_content": "ObjectId", "id_activities": "ObjectId", "progress": "Number", }]	Esta ruta devuelve un array de objetos con la información de todos los progresos de la base de datos.
GET	/progress/:id	ID	{ "id_user": "ObjectId", "id_content": "ObjectId", "id_activities": "ObjectId", "progress": "Number", }	Esta ruta devuelve un contenido específico del progreso que están almacenados en la base de datos y la trae gracias al ID que es el que determina la identificación del progreso.
POST	/progress	{ "id_user": "ObjectId", "id_content": "ObjectId", "id_activities": "ObjectId", "progress": "Number", }	Progreso creado exitosamente	Esta ruta crea un contenido en la base de datos con el ID correspondiente del progreso que se quiere crear y devuelve como salida un mensaje de éxito.
PUT	/progress/:id	ID	Progreso actualizado exitosamente	Esta ruta actualiza un progreso en la base de datos con el ID correspondiente al progreso que se quiere actualizar y devuelve como salida un mensaje de éxito.

DELETE	/progress		Todos los progresos han sido eliminados	Esta ruta elimina todos los progresos en la base de datos
DELETE	/progress/:id	ID	El progreso ha sido eliminado	Esta ruta elimina el progreso en la base de datos con el ID correspondiente al progreso que se quiere eliminar y devuelve el mensaje de eliminado con éxito.
GET	/note		[{ "id_user": "ObjectId", "id_content": "ObjectId", "id_activities": "ObjectId", "id_evaluation": "ObjectId", "note": "Number", }]	Esta ruta devuelve un array de objetos con la información de todas las notas de la base de datos.
GET	/note/:id	ID	{ "id_user": "ObjectId", "id_content": "ObjectId", "id_activities": "ObjectId", "id_evaluation": "ObjectId", "note": "Number", }	Esta ruta devuelve un objeto con la información de una nota en específico.
POST	/note	{ "id_user": "ObjectId", "id_content": "ObjectId", "id_activities": "ObjectId", "id_evaluation": "ObjectId", "note": "Number", }	Nota creada exitosamente	Esta ruta crea una nota en la base de datos con el ID correspondiente de la nota que se quiere crear y devuelve como salida un mensaje de éxito.

		}		
PUT	/note/:id	ID	Nota actualizada exitosamente	Esta ruta actualiza una nota en la base de datos con el ID correspondiente al progreso que se quiere actualizar y devuelve como salida un mensaje de éxito.
DELETE	/note		Todos las notas han sido eliminadas	Esta ruta elimina todas las notas de la base de datos y trae de vuelta un mensaje de éxito
DELETE	/note/:id	ID	La nota ha sido eliminada	Esta ruta elimina la nota en la base de datos con el ID correspondiente a la nota que se quiere eliminar y devuelve el mensaje de eliminado con éxito.
GET	/question		<pre>[{ "id_evaluation": "ObjectId", "id_content": "ObjectId", "statement": "String", "correct_answer": "String", }]</pre>	Esta ruta devuelve un array de objetos con la información de todas las preguntas de la base de datos.
GET	/question/:id	ID	<pre>{ "id_evaluation": "ObjectId", "id_content": "ObjectId", "statement": "String", "correct_answer": "String", }</pre>	Esta ruta devuelve un objeto con la información de una pregunta en específico.

POST	/question	{ "id_evaluation": "ObjectId", "id_content": "ObjectId", "statement": "String", "correct_answer": "String", }	¡Pregunta creada exitosamente!	Esta ruta crea una pregunta en la base de datos con el ID correspondiente de la pregunta que se quiere crear y devuelve como salida un mensaje de éxito.
PUT	/question/:id	ID	Pregunta actualizada exitosamente	Esta ruta actualiza una pregunta en la base de datos con el ID correspondiente al progreso que se quiere actualizar y devuelve como salida un mensaje de éxito.
DELETE	/question		Todas las preguntas han sido eliminadas	Esta ruta elimina todas las preguntas almacenadas en la base de datos.
DELETE	/question/:id	ID	Pregunta eliminada exitosamente	Esta ruta elimina la pregunta en la base de datos con el ID correspondiente a la pregunta que se quiere eliminar y devuelve el mensaje de respuesta.
GET	/evaluationData		[{ "id_evaluation": "ObjectId", "id_question": "ObjectId", "answer": "String", "state": "String", }	Esta ruta devuelve un array de objetos con la información de todos los datos de la evaluación en la base de datos.

			}}	
GET	/evaluationData/:id	ID	{ "id_evaluation": "ObjectId", "id_question": "ObjectId", "answer": "String", "state": "String", }	Esta ruta devuelve un objeto con la información de los datos de una evaluación en específico.
POST	/evaluationData	{ "id_evaluation": "ObjectId", "id_question": "ObjectId", "answer": "String", "state": "String", }	Dato de evaluación creado exitosamente	Esta ruta crea un dato específico de una evaluación en la base de datos con el ID correspondiente de la dato de la evaluación que se quiere crear y devuelve como salida un mensaje de éxito.
PUT	/evaluationData/:id	ID	Dato de evaluación actualizado exitosamente	Esta ruta actualiza un dato de la evaluación en la base de datos con el ID correspondiente a la evaluación que se quiere actualizar y devuelve como salida un mensaje de éxito.
DELETE	/evaluationData		Todos los datos de evaluación han sido eliminados	Esta ruta elimina todos los datos de la evaluación y trae un mensaje de éxito

DELETE	/evaluationData/id	ID	Dato de evaluación eliminado exitosamente	Esta ruta elimina un dato de una evaluación en la base de datos con el ID correspondiente a la evaluación que se quiere eliminar y devuelve el mensaje de eliminado con éxito.
---------------	--------------------	----	--	--

Tabla 1 (Rutas Amigables)

REFERENCIAS

Coppola, M. (2022, Sep 19). *Frontend y backend: qué son, en qué se diferencian y ejemplos*. Hubspot.es. <https://blog.hubspot.es/website/frontend-y-backend#:~:text=referimos%20al%20backend.-,Qu%C3%A9%20es%20backend,trasera%C2%BB%20de%20un%20sistema%20inform%C3%A1tico.>

Introduction to the server side - Learn web development | MDN. (2023, May 10).

Mozilla.org. https://developer.mozilla.org/en-US/docs/Learn/Server-side/First_steps/Introduction

Russo, B., & Webber, D. (2019). *NoSQL databases: A practical introduction*. Sebastopol, CA: O'Reilly Media.

Tatbul, N., & Özsu, M. T. (2013). *NoSQL databases: A survey*. *ACM Computing Surveys*, *45*(3), 15.

MongoDB. (2023). Retrieved from <https://www.mongodb.com/>

Fowler, M. (2018). *APIs: A modern guide to RESTful web services*. Sebastopol, CA: O'Reilly Media.

Kulkarni, S. (2014). *APIs: A brief history and overview*. *ACM Queue*, *12*(3), 24-34.

API World. (2023). Retrieved from <https://apiworld.co/>