

# Documentación del OVA - Aprendizaje de Bases de Datos con Nuxt.js

**Autor:** Luis alejandro arrieta milanés

Valentina luna Villadiego

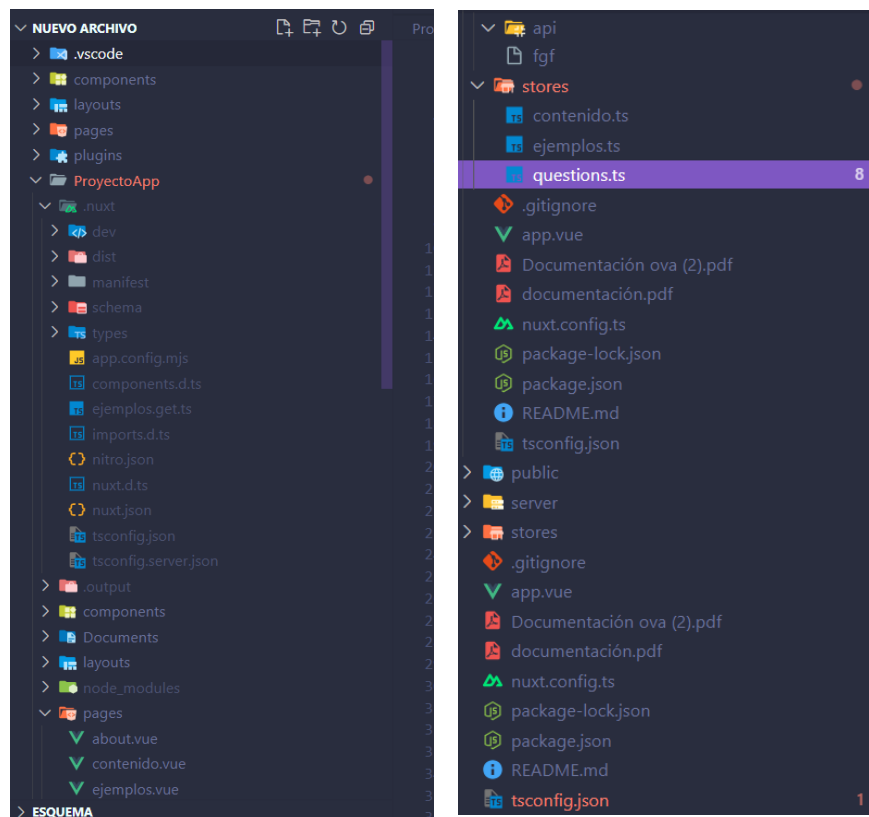
Luis David blanchar

## Introducción

Este documento describe un Objeto Virtual de Aprendizaje (OVA) desarrollado con Nuxt.js, diseñado para enseñar conceptos de bases de datos de forma interactiva. El OVA incluye secciones para contenido educativo, ejemplos prácticos y un quiz que ayuda a evaluar el aprendizaje. Utiliza TypeScript para un desarrollo más organizado y sigue las reglas de Nuxt.js, integrando un servidor simulado y un sistema para guardar el estado del usuario.

## Estructura del Proyecto

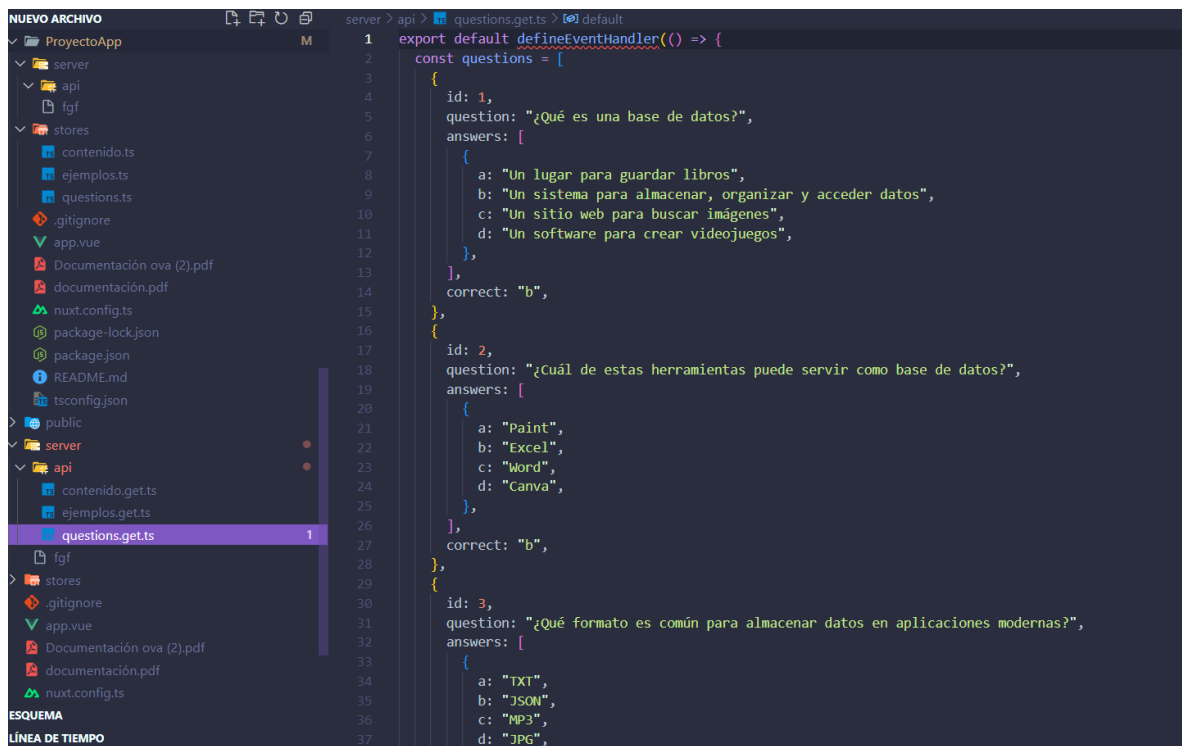
El proyecto está organizado de la siguiente manera:



## Desglose de Componentes Clave

### Conexión con el Servidor (server/api/)

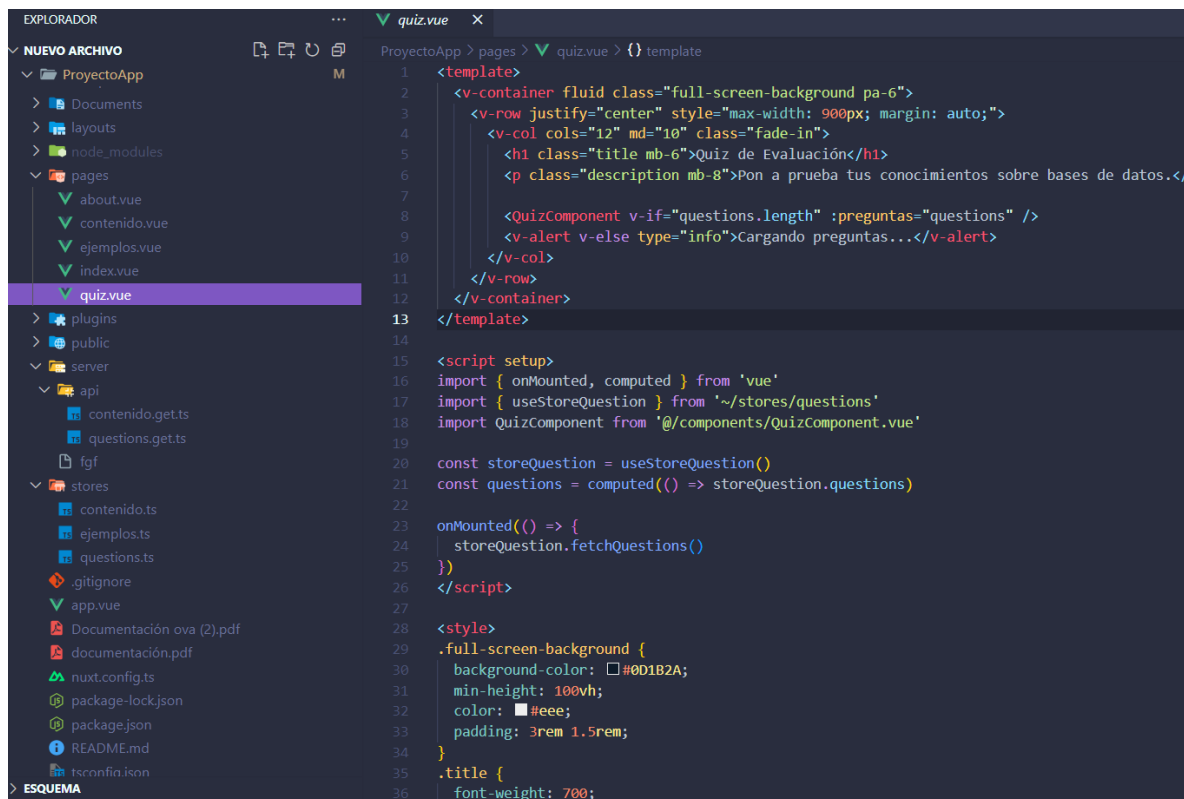
La carpeta **server/api/** contiene programas (endpoints) en TypeScript que simulan una base de datos y envían datos al OVA. El archivo **questions.get.ts** es clave para el quiz funciona así:



```
1 export default defineEventHandler(() => {
2   const questions = [
3     {
4       id: 1,
5       question: "¿Qué es una base de datos?",
6       answers: [
7         {
8           a: "Un lugar para guardar libros",
9           b: "Un sistema para almacenar, organizar y acceder datos",
10          c: "Un sitio web para buscar imágenes",
11          d: "Un software para crear videojuegos",
12        },
13      ],
14      correct: "b",
15    },
16    {
17      id: 2,
18      question: "¿Cuál de estas herramientas puede servir como base de datos?",
19      answers: [
20        {
21          a: "Paint",
22          b: "Excel",
23          c: "Word",
24          d: "Canva",
25        },
26      ],
27      correct: "b",
28    },
29    {
30      id: 3,
31      question: "¿Qué formato es común para almacenar datos en aplicaciones modernas?",
32      answers: [
33        {
34          a: "TXT",
35          b: "JSON",
36          c: "MP3",
37          d: "JPG",
```

- **Qué hace:** Este archivo crea una ruta en la web llamada `/api/questions` que envía un listado de preguntas sobre bases de datos. Cada pregunta tiene un número (id), el texto de la pregunta, varias opciones de respuesta, y la letra de la respuesta correcta (por ejemplo, "b" para la segunda opción).
- **Propósito en el OVA:** Proporciona las preguntas que los estudiantes ven en el quiz, ayudando a evaluar su entendimiento de conceptos como qué es una base de datos o qué herramientas se pueden usar.

## Páginas (pages/)



The screenshot shows a code editor with a file explorer on the left and a code editor on the right. The file explorer shows a project structure with a 'pages' folder containing 'quiz.vue'. The code editor shows the content of 'quiz.vue'.

```
1 <template>
2   <v-container fluid class="full-screen-background pa-6">
3     <v-row justify="center" style="max-width: 900px; margin: auto;">
4       <v-col cols="12" md="10" class="fade-in">
5         <h1 class="title mb-6">Quiz de Evaluación</h1>
6         <p class="description mb-8">Pon a prueba tus conocimientos sobre bases de datos.</p>
7
8         <QuizComponent v-if="questions.length" :preguntas="questions" />
9         <v-alert v-else type="info">Cargando preguntas...</v-alert>
10      </v-col>
11    </v-row>
12  </v-container>
13 </template>
14
15 <script setup>
16   import { onMounted, computed } from 'vue'
17   import { useStoreQuestion } from '~/stores/questions'
18   import QuizComponent from '@components/QuizComponent.vue'
19
20   const storeQuestion = useStoreQuestion()
21   const questions = computed(() => storeQuestion.questions)
22
23   onMounted(() => {
24     storeQuestion.fetchQuestions()
25   })
26 </script>
27
28 <style>
29   .full-screen-background {
30     background-color: #0D1B2A;
31     min-height: 100vh;
32     color: #eee;
33     padding: 3rem 1.5rem;
34   }
35   .title {
36     font-weight: 700;
```

La carpeta **pages/** contiene las secciones interactivas del OVA. El archivo **quiz.vue** es la parte del quiz y se explica a continuación:

- **Qué hace:** Esta página, accesible en /quiz, muestra un título ("Quiz de Evaluación") y una instrucción para los estudiantes. Usa un componente llamado QuizComponent para mostrar las preguntas, que se cargan desde el servidor usando el store. Si las preguntas no están listas, muestra un mensaje de "Cargando preguntas...". El diseño usa Vuetify (v-container, v-row, v-col) para que se vea bien en diferentes dispositivos, con un fondo oscuro y texto claro.
- **Propósito en el OVA:** Permite a los estudiantes interactuar con el quiz, viendo las preguntas y respondiéndolas, lo que ayuda a reforzar su aprendizaje sobre bases de datos.

## Gestión de Estado (stores/)

La carpeta **stores/** guarda el estado del OVA usando Pinia. El archivo **questions.ts** maneja las preguntas del quiz

```
1 import { defineStore } from 'pinia'
2
3 const isClient = typeof window !== 'undefined'
4
5 export const useStoreQuestion = defineStore('questions', {
6   state: () => ({
7     questions: [],
8     currentQuestion: null,
9   }),
10  actions: {
11    async fetchQuestions() {
12      try {
13        const response = await $fetch('/api/questions', { method: 'GET' })
14        this.questions = response.map(q => {
15          const opts = q.answers[0]
16          return {
17            id: q.id,
18            pregunta: q.question,
19            opciones: [opts.a, opts.b, opts.c, opts.d],
20            respuestaCorrecta: opts[q.correct],
21          }
22        })
23      } catch (error: unknown) {
24        console.error('Error fetching questions:', error)
25      }
26    },
27    setCurrentQuestion(question) {
28      this.currentQuestion = { ...question }
29    },
30    updateQuestion(updatedQuestion) {
31      const index = this.questions.findIndex(q => q.id === updatedQuestion.id)
32      if (index !== -1) {
33        this.questions[index] = { ...updatedQuestion }
34        if (this.currentQuestion?.id === updatedQuestion.id) {
35          this.currentQuestion = { ...updatedQuestion }
36        }
37      }
38    }
39  }
40 })
```

**Qué hace:** Este archivo crea un almacén (store) llamado 'questions' que guarda las preguntas en una lista (questions) y la pregunta actual (currentQuestion). La acción fetchQuestions obtiene las preguntas desde /api/questions, las transforma para que tengan un formato claro (pregunta, opciones, respuesta correcta), y las guarda. Las acciones setCurrentQuestion y updateQuestion permiten manejar la pregunta que se está mostrando o actualizarla si cambia.

- **Propósito en el OVA:** Asegura que las preguntas estén disponibles para el quiz y que el sistema pueda seguir el progreso del estudiante, como cambiar a la siguiente pregunta o actualizar respuestas.

### Relación entre server/api/, pages/, y stores/

El OVA conecta estos componentes de la siguiente manera:

1. **server/api/questions.get.ts** envía las preguntas al sistema.
2. **stores/questions.ts** recoge estas preguntas y las guarda para usarlas.
3. **pages/quiz.vue** usa el almacén para mostrar las preguntas al estudiante.

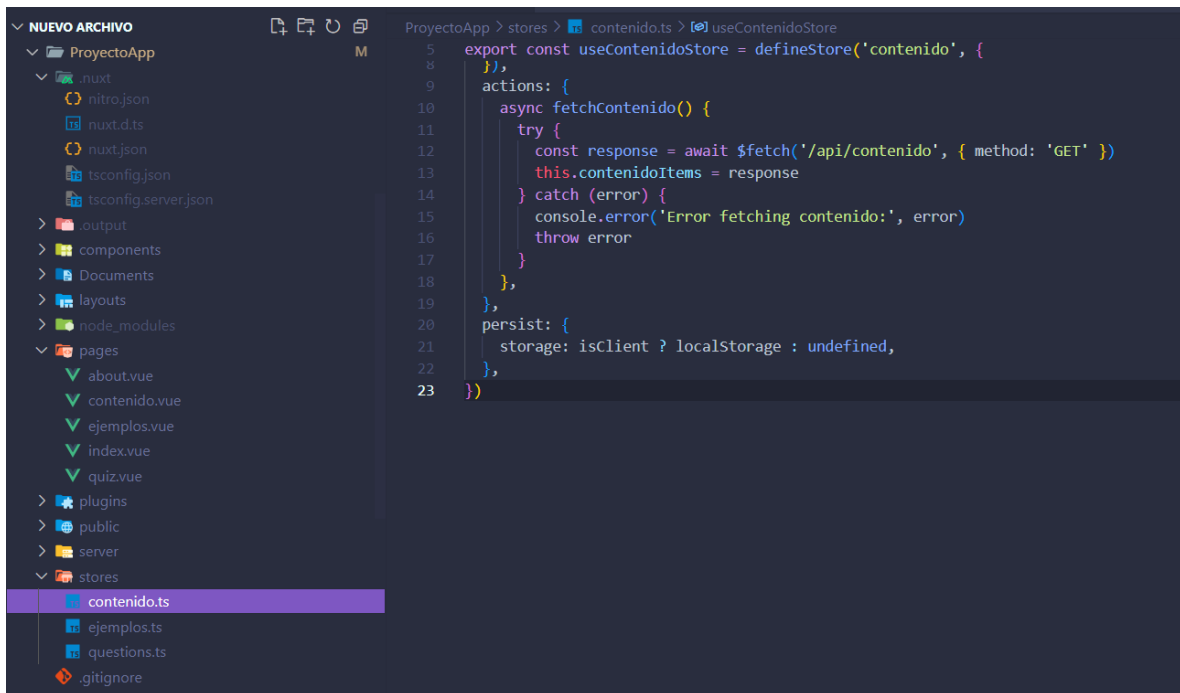
y así mismo sucede con las demás paginas como contenido, ejemplos...

### PAGES CONTENIDO Y EJEMPLOS

```
NUEVO ARCHIVO ProyectoApp > pages > contenido.vue > {} style scoped > .fade-in
1 <template>
2 <v-container fluid class="full-screen-background pa-6">
3 <v-row justify="center" style="max-width: 900px; margin: auto;">
4 <v-col cols="12" class="fade-in">
33 <div v-if="loading && !error">
53 <p v-if="item.usageContext">{{ item.usageContext }}</p>
54
55 <!-- Usos (si existen) -->
56 <v-list v-if="item.usos" dense class="dark-list">
57 <v-list-item v-for="(uso, index) in item.usos" :key="index">
58 <v-list-item-icon>
59 <v-icon color="#008809">mdi-check-circle</v-icon>
60 </v-list-item-icon>
61 <v-list-item-content>
62 <v-list-item-title class="card-text">{{ uso }}</v-list-item-title>
63 <v-list-item-content>
64 </v-list-item-content>
65 </v-list-item>
66 </v-list>
67
68 <!-- Tabla de estudiantes (si existe) -->
69 <div v-if="item.estudiantes">
70 <p class="mt-4">{{ item.exampleIntro }}</p>
71 <v-simple-table class="elevation-1 mt-2" style="color:#A9D6E5">
72 <thead>
73 <tr style="background-color: #112F3C; color: #008809;">
74 <th>ID</th>
75 <th>Nombre</th>
76 <th>Grado</th>
77 <th>Nota Matemáticas</th>
78 </tr>
79 </thead>
80 <tbody>
81 <tr v-for="est in item.estudiantes" :key="est.id">
82 <td>{{ est.id }}</td>
83 <td>{{ est.nombre }}</td>
84 <td>{{ est.grado }}</td>
85 <td>{{ est.nota }}</td>
```

```
NUEVO ARCHIVO ProyectoApp > pages > ejemplos.vue > {} template > v-container.full-screen-background.pa-6 > v-row
1 <template>
2 <v-container fluid class="full-screen-background pa-6" style="overflow-x: hidden">
3 <v-row justify="center" style="max-width: 900px; margin: auto;">
4
5 <v-col cols="12">
6 <h1 class="title mb-8">Ejemplos prácticos en educación</h1>
7 </v-col>
8
9 <!-- Aquí usamos cols="12" para móvil y md="6" para pantalla mediana -->
10 <v-col
11 v-for="(ejemplo, index) in ejemplos"
12 :key="index"
13 cols="12"
14 md="6"
15 class="d-flex"
16 >
17 <EjemploPractico
18 :titulo="ejemplo.titulo"
19 :descripcion="ejemplo.descripcion"
20 :pasos="ejemplo.pasos"
21 :imagen="ejemplo.imagen"
22 class="flex-grow-1"
23 />
24 </v-col>
25
26 <v-col cols="12" class="d-flex justify-end mt-12">
27 <v-btn
28 color="btn-primary"
29 large
30 rounded
31 to="/quiz"
32 >
33 Ir a Evaluación
34 <v-icon right>mdi-arrow-right</v-icon>
35 </v-btn>
36 </v-col>
37
```

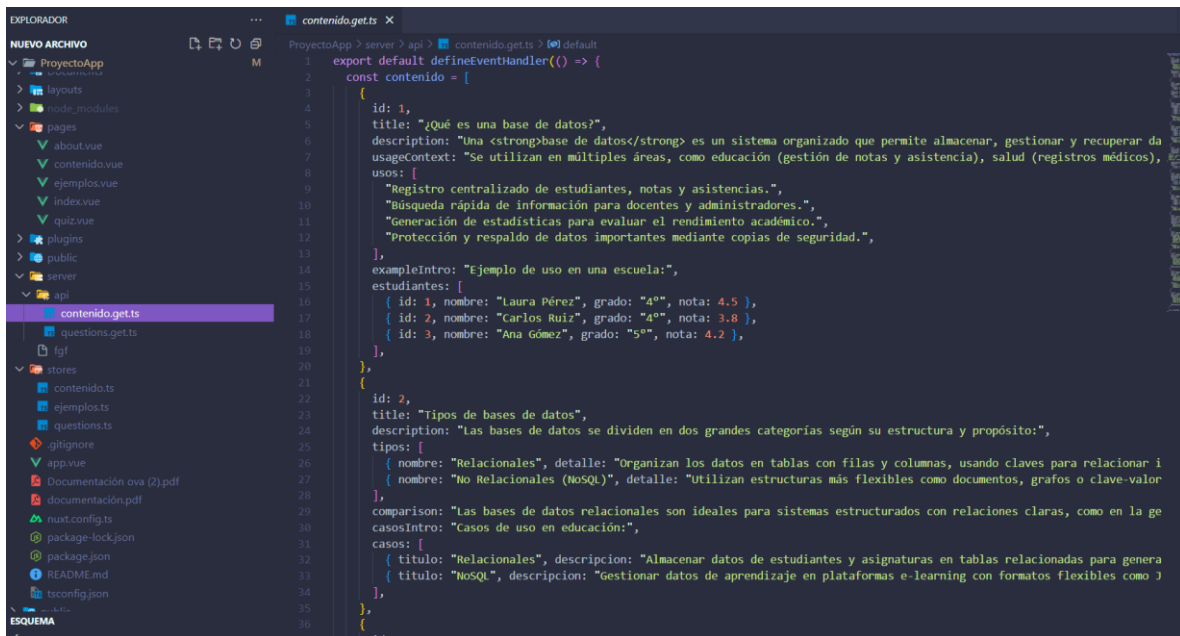
## STORES/CONTENIDO



The screenshot shows the VS Code interface. On the left, the file explorer displays the project structure. The 'stores' folder is expanded, and 'contenido.ts' is selected. The main editor shows the code for 'contenido.ts'.

```
ProjectoApp > stores > contenido.ts > useContenidoStore
5 export const useContenidoStore = defineStore('contenido', {
8   },
9   actions: {
10     async fetchContenido() {
11       try {
12         const response = await $fetch('/api/contenido', { method: 'GET' })
13         this.contenidoItems = response
14       } catch (error) {
15         console.error('Error fetching contenido:', error)
16         throw error
17       }
18     },
19   },
20   persist: {
21     storage: isClient ? localStorage : undefined,
22   },
23 })
```

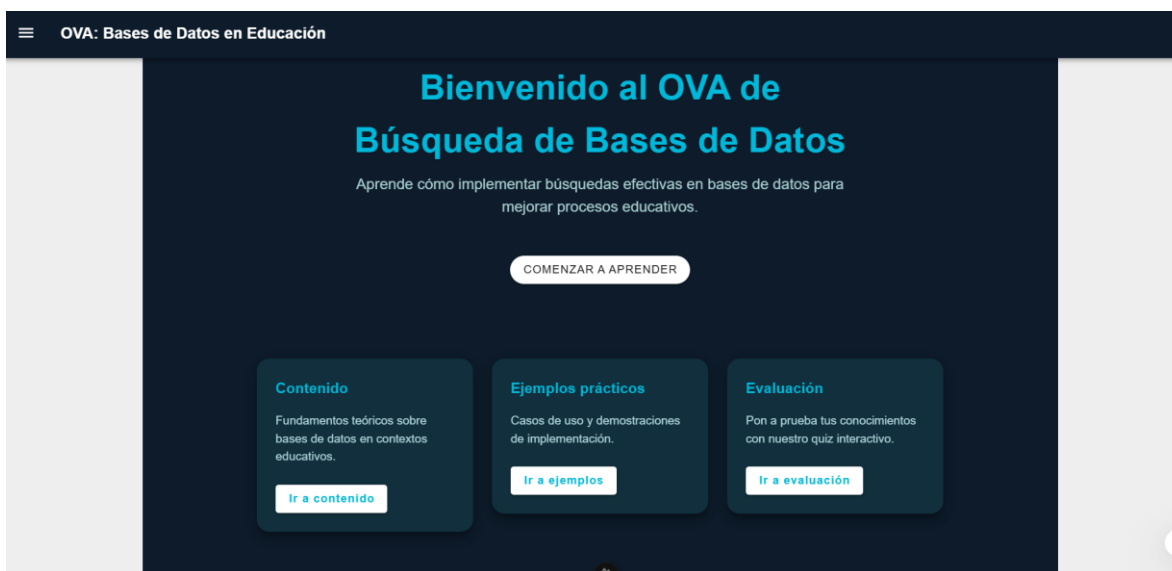
## SERVER/CONTENIDO:



The screenshot shows the VS Code interface. On the left, the file explorer displays the project structure. The 'api' folder is expanded, and 'contenido.getts' is selected. The main editor shows the code for 'contenido.getts'.

```
ProjectoApp > server > api > contenido.getts > default
1 export default defineEventHandler(() => {
2   const contenido = [
3     {
4       id: 1,
5       title: "¿Qué es una base de datos?",
6       description: "Una <strong>base de datos</strong> es un sistema organizado que permite almacenar, gestionar y recuperar da
7       usageContext: "Se utilizan en múltiples áreas, como educación (gestión de notas y asistencia), salud (registros médicos),
8       usos: [
9         "Registro centralizado de estudiantes, notas y asistencias.",
10        "Búsqueda rápida de información para docentes y administradores.",
11        "Generación de estadísticas para evaluar el rendimiento académico.",
12        "Protección y respaldo de datos importantes mediante copias de seguridad.",
13      ],
14      exampleIntro: "Ejemplo de uso en una escuela:",
15      estudiantes: [
16        { id: 1, nombre: "Laura Pérez", grado: "4º", nota: 4.5 },
17        { id: 2, nombre: "Carlos Ruiz", grado: "4º", nota: 3.8 },
18        { id: 3, nombre: "Ana Gómez", grado: "5º", nota: 4.2 },
19      ],
20    },
21    {
22      id: 2,
23      title: "Tipos de bases de datos",
24      description: "Las bases de datos se dividen en dos grandes categorías según su estructura y propósito:",
25      tipos: [
26        { nombre: "Relacionales", detalle: "Organizan los datos en tablas con filas y columnas, usando claves para relacionar i
27        { nombre: "No Relacionales (NoSQL)", detalle: "Utilizan estructuras más flexibles como documentos, grafos o clave-valor
28      ],
29      comparison: "Las bases de datos relacionales son ideales para sistemas estructurados con relaciones claras, como en la ge
30      casosIntro: "Casos de uso en educación:",
31      casos: [
32        { titulo: "Relacionales", descripcion: "Almacenar datos de estudiantes y asignaturas en tablas relacionadas para genera
33        { titulo: "NoSQL", descripcion: "Gestionar datos de aprendizaje en plataformas e-learning con formatos flexibles como J
34      ],
35    },
36  ],
37 })
```

Interfaz de nuestro OVA hasta el momento:



## Ejemplos prácticos en educación

### Búsqueda en ERIC

ERIC es una base de datos académica espe...

#### Pasos a seguir:

1. Accede a <https://eric.ed.gov/>
2. Haz clic en "Advanced Search".
3. Escribe términos como: "literacy AN...
4. Filtra por fecha, nivel educativo o tip...
5. Selecciona un resultado, revisa su r...

### Uso de operadores boole...

Aprende cómo combinar operadores como A...

#### Pasos a seguir:

1. Ve a <https://scholar.google.com/>
2. Escribe una consulta como: "matem...
3. Compara resultados con diferentes ...
4. Filtra por idioma y por año.
5. Guarda los resultados en tu perfil de...

## Quiz de Evaluación

Pon a prueba tus conocimientos sobre bases de datos.

### ¿Qué es una base de datos?

- Un lugar para guardar libros
- Un sistema para almacenar, organizar y acceder datos
- Un sitio web para buscar imágenes
- Un software para crear videojuegos

SIGUIENTE