DOCUMENTACIÓN

CARLOS ANDRÉS LARA MONTES

ÁLVARO PIO VILLALBA PAÉZ

JESI JOSÉ CORREA GALVAN

UNIVERSIDAD DE CÓRDOBA

FACULTAD DE EDUCACIÓN Y CIENCIAS HUMANAS

LICENCIATURA EN INFORMÁTICA

DOCENTE: ALEXANDER TOSCANO

MONTERÍA - CÓRDOBA

JUNIO, 2025

Documentación del OVA: Búsqueda en Bases de Datos Científicas

1. Introducción

Este Objeto Virtual de Aprendizaje (OVA) fue desarrollado como parte del curso Planeación de Proyectos de Investigación de la Licenciatura en Informática Educativa de la Universidad de Córdoba. El objetivo es enseñar a los estudiantes a realizar búsquedas estructuradas de información científica en bases de datos, utilizando operadores booleanos, descriptores, y estrategias de búsqueda eficientes.

2. Estructura Tecnológica

El OVA fue desarrollado usando las siguientes tecnologías:
- Nuxt 3 (Vue 3 y Composition API)
- Vuetify para el diseño de interfaz
- Pinia para la gestión de estado (store)
- TailwindCSS para estilos adicionales
- MongoDB (en fase de planificación)
- SCORM (para integración con LMS)

3. Páginas del OVA

3.1 Inicio

Página introductoria que presenta el objetivo del OVA y su importancia en la búsqueda de información académica. Se usa un diseño atractivo con una sección central en un card destacando el propósito general del recurso.

3.2 Contenido

La página de contenido presenta los módulos temáticos: definición de revisión sistemática, tipos de fuentes, bases de datos, operadores booleanos, estrategias de búsqueda y uso de herramientas de inteligencia artificial como PRISMA, Elicit o ChatGPT. Cada módulo se presenta con una tarjeta que adapta dinámicamente su tamaño para mostrar todo el contenido sin cortes.

3.3 Actividades

Contiene cuatro actividades diseñadas con componentes reutilizables, empleando `props`, `emits` y `v-model`. Las actividades incluyen: búsqueda guiada en Google Scholar, juego de emparejamiento de operadores, simulación de estrategia de búsqueda y un caso práctico.

3.4 Evaluación

Incluye un cuestionario funcional con 3 preguntas de opción múltiple y evaluación automática. Además, se presenta una rúbrica y un entregable que deben ser desarrollados por los estudiantes. El cuestionario fue implementado como componente separado y emite los resultados al finalizar.

3.5 Créditos

Se listan los desarrolladores, docentes supervisores y las tecnologías usadas. Se mejoró el diseño de las tarjetas, presentando la información de forma limpia y legible, una debajo de la otra.

4. Componentes Personalizados

- Quiz.vue: Componente para el cuestionario, usa props para recibir preguntas y emite los resultados.
- ActividadCard.vue: Componente para representar cada actividad.
- CreditCard.vue: Tarjeta con iconos e información organizada sobre los créditos.
- Layout default.vue: Define el diseño base, con barra lateral, app bar y pie de página.
- Uso de App.vue para heredar tema global y diseño.

5. Store y API

Se implementó un store `evaluacion.ts` en Pinia, con `state` estructurado según el servidor, y una acción asíncrona para obtener los datos de evaluación desde `/api/evaluacion`. El archivo `evaluacion.get.ts` del servidor retorna el cuestionario, entregable y rúbrica.

6. Consideraciones Finales

Este OVA es adaptable y escalable. Puede integrarse con un LMS mediante SCORM, exportarse como PWA y es fácil de personalizar gracias al uso modular de Nuxt 3 y Vuetify. A futuro se pueden agregar nuevas funcionalidades como retroalimentación automática y conexión con bases de datos reales como Scopus o PubMed.

Ejemplo de Código: Página de Evaluación

La página de evaluación combina un cuestionario interactivo con resultados, entregable y rúbrica.

```
<template>
 <v-container class="py-12">
  <v-row justify="center">
   <v-col cols="12" md="10">
    <v-sheet elevation="6" rounded color="#6A1B9A" class="pa-6">
     <h1 class="text-white">Evaluación del OVA</h1>
    </v-sheet>
   </v-col>
  </v-row>
  <v-row justify="center" class="mt-8" dense>
   <v-col cols="12" md="6">
    <v-card>
     <Quiz :preguntas="preguntasEjemplo"
@finalizado="guardarResultado" />
    </v-card>
   </v-col>
  </v-row>
 </v-container>
</template>
```

Store de Evaluación (Pinia)

```
export const useEvaluacionStore = defineStore('evaluacion', {
 state: () => ({
  cuestionario: { tipo: '', preguntas: 0, formatos: [] },
  entregable: '',
  rubrica: []
 }),
 actions: {
  async fetchEvaluacion() {
   const data = await $fetch('/api/evaluacion')
   this.cuestionario = data.cuestionario
   this.entregable = data.entregable
   this.rubrica = data.rubrica
  }
```

```
  }
})
```

API evaluacion.get.ts

```
export default defineEventHandler(() => {
 return {
  cuestionario: {
   tipo: "SCORM",
   preguntas: 10,
   formatos: ["Opción múltiple", "Falso/Verdadero"]
  },
  entregable: "Diseño de una estrategia de búsqueda para un tema
específico",
  rubrica: [
   { nivel: "Básico", descripcion: "Uso de una sola palabra clave" },
   { nivel: "Intermedio", descripcion: "Uso de operadores" },
   { nivel: "Avanzado", descripcion: "Planificación con criterios
PRISMA" }
  ]
 }
})
```
