Aquí un resumen más detallado:

markdown

# Quiz App (Next.js 15 + Prisma + PostgreSQL)

## Estructura Completa del Proyecto

src/ ├── app/ │ ├── page.tsx # Lista de quizzes y opción de crear nuevo │ ├── layout.tsx # Layout principal │ ├── globals.css # Estilos globales │ ├── quiz/[id]/page.tsx # Página dinámica para tomar quiz │ └── api/ │ ├── quiz/route.ts # POST: crear quiz desde DOCX │ └── progress/route.ts # GET/POST: manejo de progreso ├── components/ │ ├── quiz/ │ │ ├── QuizContainer.tsx # Estado principal y lógica del quiz │ │ ├── Question.tsx # Renderiza pregunta y maneja respuestas │ │ └── FileUploader.tsx # Procesa archivos DOCX │ └── ui/ │ └── Button.tsx # Botón reutilizable └── lib/ ├── prisma.ts # Cliente Prisma singleton ├── types.ts # Interfaces TypeScript └── fileProcessor.ts # Lógica de procesamiento DOCX

Copy

## Componentes Principales

### QuizContainer

```typescript

type QuizContainerProps = {

initialQuestions?: QuestionType[];

isExistingQuiz?: boolean;

quizId?: number;

savedProgress?: SavedProgress;

};

interface SavedProgress {

currentQuestion: number;

correctAnswers: number;

incorrectAnswers: number;

isFinished: boolean;

}

* Maneja todo el estado del quiz
* Guarda progreso automáticamente al responder
* Permite reiniciar o continuar quiz
* Muestra estadísticas finales

**Question**

typescript

type QuestionProps = {

question: QuestionType;

onAnswer: (isCorrect: boolean) => void;

};

* Muestra pregunta y opciones
* Feedback inmediato al responder
* Muestra explicación después de responder

**FileUploader**

typescript

Copy

type FileUploaderProps = {

onFileSelect: (file: File) => void;

};

* Acepta archivos DOCX
* Procesa y valida formato
* Envía a API para crear quiz

**Base de Datos (PostgreSQL en Neon)**

prisma

model User {

id Int @id @default(autoincrement())

email String @unique

name String?

userProgress UserProgress[]

createdAt DateTime @default(now())

updatedAt DateTime @updatedAt

}

model UserProgress {

id Int @id @default(autoincrement())

userId Int

quizId Int

currentQuestion Int @default(0)

correctAnswers Int @default(0)

incorrectAnswers Int @default(0)

startedAt DateTime @default(now())

lastAccess DateTime @updatedAt

isFinished Boolean @default(false)

user User @relation(fields: [userId], references: [id])

quiz Quiz @relation(fields: [quizId], references: [id])

@@unique([userId, quizId]) // Garantiza un solo progreso por usuario/quiz

}

model Quiz {

id Int @id @default(autoincrement())

title String

questions Question[]

userProgress UserProgress[]

createdAt DateTime @default(now())

updatedAt DateTime @updatedAt

}

model Question {

id Int @id @default(autoincrement())

quizId Int

text String

explanation String?

options Option[]

quiz Quiz @relation(fields: [quizId], references: [id])

createdAt DateTime @default(now())

updatedAt DateTime @updatedAt

}

model Option {

id Int @id @default(autoincrement())

questionId Int

text String

isCorrect Boolean

question Question @relation(fields: [questionId], references: [id])

createdAt DateTime @default(now())

updatedAt DateTime @updatedAt

}

**API Endpoints**

**POST /api/quiz**

typescript

*// Crea nuevo quiz desde archivo DOCX*

body: {

title: string;

questions: Array<{

text: string;

explanation: string;

options: Array<{

text: string;

isCorrect: boolean;

}>;

}>;

}

**GET /api/progress**

typescript

*// Obtiene progreso de un quiz*

params: {

quizId: string;

}

**POST /api/progress**

typescript

*// Guarda progreso de quiz*

body: {

quizId: number;

currentQuestion: number;

correctAnswers: number;

incorrectAnswers: number;

isFinished: boolean;

}

**Formato DOCX Esperado**

Copy

[Pregunta]

¿Cuál es la pregunta?

[Opción A]

Primera opción

[Opción B]

Segunda opción

^[Opción C]

Tercera opción (correcta)

[Opción D]

Cuarta opción

[Explicación]

Explicación de la respuesta correcta.

**Variables de Entorno**

env

# URL de conexión a Neon PostgreSQL

DATABASE\_URL="postgresql://neondb\_owner:password@ep-host-pooler.region.azure.neon.tech/neondb?sslmode=require"

**Configuración Next.js**

javascript

*// next.config.js*

const nextConfig = {

reactStrictMode: true,

swcMinify: true,

output: 'standalone'

};

module.exports = nextConfig;

**Dependencias Principales**

json

{

"dependencies": {

"@prisma/client": "^6.3.1",

"mammoth": "^1.9.0",

"next": "15.1.6",

"prisma": "^6.3.1",

"react": "^19.0.0",

"react-dom": "^19.0.0"

}

}

**Scripts**

json

{

"scripts": {

"dev": "next dev",

"build": "prisma generate && next build",

"start": "next start",

"lint": "next lint"

}

}

**Estado Actual y Funcionalidades**

1. Creación de Quiz
   * Carga de archivo DOCX
   * Procesamiento automático
   * Validación de formato
   * Guardado en base de datos
2. Toma de Quiz
   * Visualización de preguntas una a una
   * Selección de respuesta
   * Feedback inmediato
   * Explicación después de responder
   * Progreso guardado automáticamente
3. Progreso
   * Guardado al responder cada pregunta
   * Opción de continuar desde última pregunta
   * Estadísticas de rendimiento
   * Un solo registro por usuario/quiz
4. Interfaz
   * Lista de quizzes disponibles
   * Detalles de cada quiz
   * Opción de nuevo intento o continuar
   * Estadísticas al finalizar

Copy

Este resumen incluye todos los detalles técnicos y funcionales necesarios para entender y seguir desarrollando la aplicación. ¿Hay algún aspecto específico que quieras que detalle más?

 CopyRetry

NR