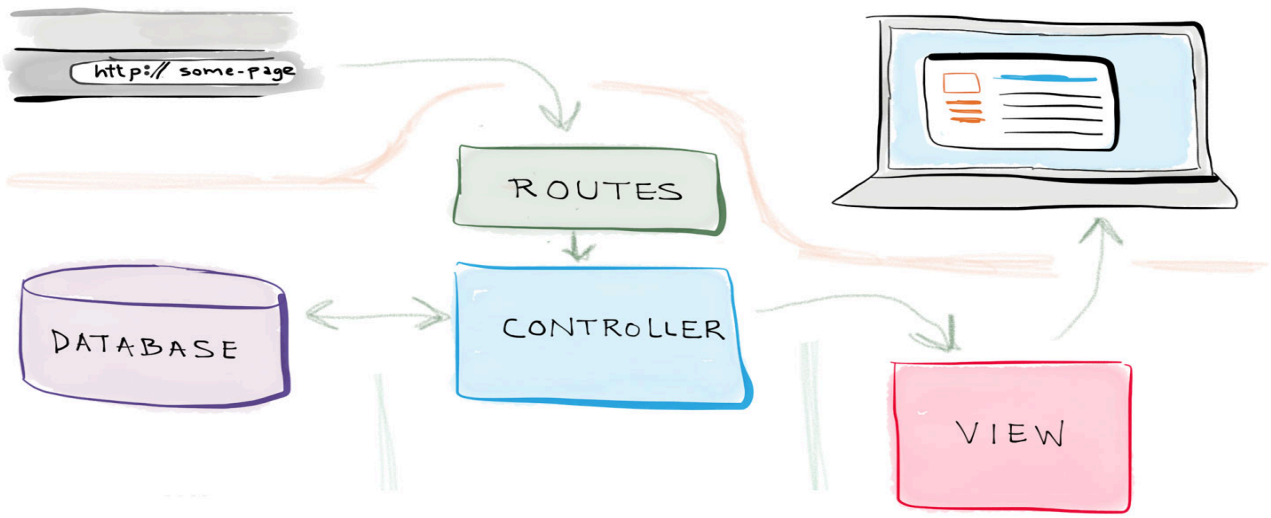

Tarea 4: Aplicación de búsqueda de piezas arqueológicas

Haremos una aplicación 'MPA' con dos páginas, la portada de la tarea anterior, con un formulario de búsqueda, y otra que muestre los resultados consultando la BD.

Nueva Base de Datos

La búsqueda que vamos a hacer es sobre los campos de texto de la BD. *SQLITE* no soporta este tipo de búsquedas, asi que nos pasamos a *PostgreSQL*. La mayor ventaja de los *ORMs* como *Prisma* es la facilidad para cambiar de BD manteniendo el código anterior.

Instalaremos la BD localmente en un contenedor docker, muy usados tanto en desarrollo como para empaquetar las aplicaciones y subirlas a la nube. Seguimos Postgres in Docker - A Practical Guide. Una vez instalada se puede comprobar su funcionamiento con alguna extensión de vscode., o con *prismastudio* una vez hecho el cliente de *prisma*.

El archivo **docker-compose.yml** queda:

```
version: '3'

volumes:
  psql:

services:
  psql:
    image: postgres:17-alpine
    restart: unless-stopped
    ports:
      - 5432:5432
    volumes:
      - psql:/var/lib/postgresql/data
    environment:
      - POSTGRES_PASSWORD=${POSTGRES_PASSWORD}
      - POSTGRES_USER=${POSTGRES_USER}
      - POSTGRES_DB=${POSTGRES_DB}
```

Donde se usa la versión `17` de *postgres*, en lugar de `latest`, que podría variar durante el desarrollo. Las variables de entorno se cogen el archivo **.env** por defecto.

Migración de la BD

Siguiedo la documentación de `Prisma` : Full text search, y Data sources, el archivo **schema.primsa** queda ahora

```
generator client {
  provider = "prisma-client-js"
  previewFeatures = ["fullTextSearchPostgres"]
}
```

```
datasource db {
  provider = "postgresql"
  url      = env("DATABASE_URL")
}

model Obra {
  id            Int @id @default(autoincrement())
  título        String @unique
  imágen        String
  descripción   String
  procedencia   String
  comentario    String
}
```

con:

```
DATABASE_URL="postgresql://${POSTGRES_USER}:${POSTGRES_PASSWORD}@local
```

en el archivo **.env**

Al cambiar de BD, hay que borrar la carpeta **migrations**, y generarla otra vez:

```
> rm -rf ./prisma/migrations
> pnpm prisma migrate dev --name a_postgress
```

pasamos otra vez el script de población, y si todo va bien podemos comprobar que funciona con *`prisma studio`*, o con la extensión de vscode.

Formulario de búsqueda

Utilizar la búsqueda de texto en *Postgres* no es inmediato, así que limitaremos la búsqueda a una palabra, usando la validación por expresón regular del *html*. Es mejor para este caso usar `GET` , que `POST` , puesto que vamos a leer, y además se generan urls de búsqueda que se pueden compartir.

Para este tipo de búsquedas de texto hubiera sido mejor usar Elastic Search.

Routers

Usaremos el patrón MVC, para tener el código ordenado, es decir en archivos y carpetas separadas, de manera que sea fácil localizar el archivo de cada parte del código. Aśi, el código para atender a los los urls que tengan que ver con la tabla 'Obras' ('/obras'), estará en la carpeta **./routes/obras.mjs**

```
// ./routes/obras.mjs
import express from "express"
const router = express.Router();

import { PrismaClient } from '@prisma/client'
const prisma = new PrismaClient()

router.get('/buscar', async (req, res)=>{ /obras/buscar
  const búsqueda = req.query.búsqueda
  console.log(búsqueda)
  try {
    ...
    res.render('resultados.njk', {...})    // ../views/resultados.njk
  } catch (err) {
        console.error(err)
    res.status(500).send({err}) // o usar una página de error personal
  }
```

```
})
export default router
```

que enlaza con el servidor **index.mjs**

```
// index.mjs
...
import obrasRouter from "./routes/obras.mjs"
app.use("/obras", obrasRouter);           // para los urls que empiece
...
```

La respuesta puede ser más o menos como esta, pero incluyendo solo los tres primeros resultados, ya que se pueden ordenar por relevancia, es decir saldrán primero los que más se repita la palabra, o esta salga antes en el texto.

Referencias:

- La etiqueta <form>
- Form Validation Part 1: Constraint Validation in HTML
- Understanding Routing In Express.js As Simple As Possible
- How To Retrieve URL and POST Parameters with Express
- JavaScript Array slice() Method
- Nunjucks Filters
- Prisma Full Text Search