**Next.js y Strapi**

**CSS**

Como principal estará “globals.css” ahí es donde se pone todos los datos generales para la página, ya sea color o fuente o tamaños.

:root {

  --primary: #e99401;

  --negro: #262626;

  --blanco: #FFF;

  --fuente-principal: 'Outfit', sans-serif;

}

html {

  font-size: 62.5%;

  box-sizing: border-box;

}

\*, \*:before, \*:after {

  box-sizing: inherit;

}

body {

  font-size: 2rem;

  font-family: var(--fuente-principal);

  line-height: 1.5;

}

img {

  max-width: 100%;

}

[class\*="contenedor"] {

  max-width: 120rem;

  margin: 0 auto;

  width: 95%;

}

[class\*="heading"] {

  font-size: 4rem;

  text-align: center;

  font-weight: 900;

  color: var(--primary);

  margin: 5rem 0;

}

Root : en este se pone los colores y fuentes principales de el proyecto.

Para usar estos estilos en nuestro proyecto cuando estemos creando los CSS para cada pagina o sección de la misma, lo llamamos de esta manera:

.navegacion a {

  color: var(--blanco);

}

En donde poner “color” escribimos “var” y entre paréntesis el nombre.

En casos de ajustar según si es pantalla de móvil o de ordenaro, existe la opción “@media” que debemos añadir en la CSS que usemos en la página.

.navegacion {

  margin-top: 5rem;

  display: flex;

  gap: 2rem;

}

@media (min-width: 768px) {

  .navegacion {

    margin-top: 0;

  }

}

En ella se escribirá lo que queremos editar según el tamaño, en caso de no cambiarse no escribimos nada.

(FALLO, si ocurre un error en consola posiblemente sea por poner un link con una imagen sin texto, en ese caso debes poner <a> </a> antes y después de la imagen dejandolo como se ve abajo

<Link href="/">

            <a>

              <Image

                width={400}

                height={100}

                src="/img/logo.svg"

                alt="Imagen logo"

              ></Image>

            </a>

          </Link>

**STRAPI**

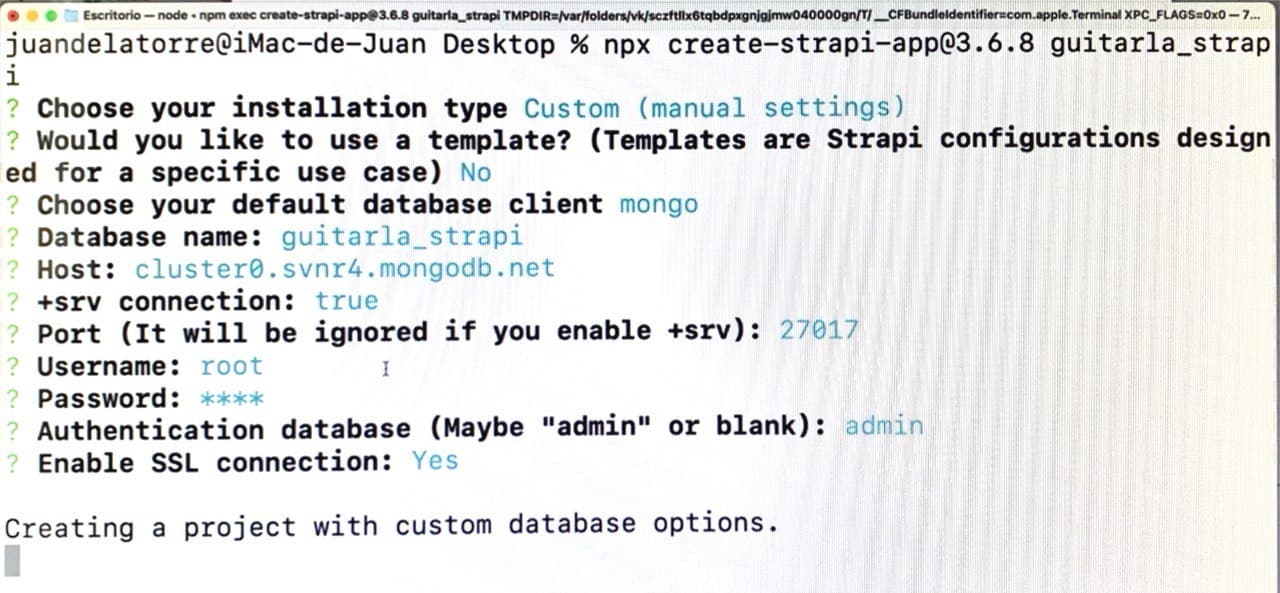
Hay que instalar strapi en nuestro proyecto debemos seguir varios pasos.

En primer lugar, hay que tener en cuenta que la versión de Strapi 4.x no funciona correctamente con Node.js versión 16.14.0. Los errores que da son varios, aunque funcione tiene muchos que son bastante molestos como no poder obtener meta datos mas haya de título y descripción.

Usamos la versión de Strapi 3.6.8 (la ultima antes de la versión 4.x) funciona correctamente con la versión de Node.js 16.14.0.

* Debemos abrir terminal en nuestro proyecto y escribir esto “npx create-strapi-app@3.6.8”.
* Después nos pedirá el nombre del proyecto.
* Después elegimos seleccionamos “Custom”, nos preguntará si queremos “template2 y decimos que no, y nos dará una lista para elegir qué base de datos usaremos, en nuestro caso será Mongo.
* Nos pedirá confirmar el nombre de la base de datos (en este caso la misma).
* Host de Mongo, deberemos crear un usuario y contraseña para autenticarnos (no es necesario, pero es más seguro) se encuentra en “Quickstart”. Luego para obtendremos el “Host de Mongo” en la página en “Database” clicamos en “Connect” y luego damos a “Connect your application” veremos un string de conexión donde solo copiaremos desde después del primer -@- hasta -.net-.
* Escribimos “true” en +srv.
* Port ponemos el que salga por defecto, clicando en Enter.
* Escribimos nuestro usuario y contraseña.
* Autenticación ponemos “admin”.
* SSL ponemos “Yes”.

Debería ser algo como esto aproximadamente.



Obtener datos en Next.js (consulta API)

Existen 3 formas:

* getStaticProps – Obtener los datos cuando se crea el build. Cuando son datos staticos y no se necesite un constante refresco de ellos.
* getStatiscPatchs – Obtener el listado de paginas que se van a crear cuando se realiza el build.
* getServerSideProps – Los datos se obtienen en cada request. Cuando son datos que van cambiando cada poco, consume mucho de el servidor usar para datos cambiantes.

Ejemplo blog:

import Layout from "../components/Layout";

import Entrada from "../components/Entrada";

import styles from "../styles/Blog.module.css";

const Blog = ({ entradas }) => {

  return (

    <div>

      <Layout pagina="Blog">

        <main className="contenedor">

          <h2 className="heading">Blog</h2>

          <div className={styles.blog}>

            {entradas.map((entrada) => (

              <Entrada key={entrada.id} entrada={entrada} />

            ))}

          </div>

        </main>

      </Layout>

    </div>

  );

};

export async function getStaticProps() {

  const url = `http://localhost:1337/blogs`;

  const respuesta = await fetch(url);

  const entradas = await respuesta.json();

  return {

    props: {

      entradas,

    },

  };

}

export default Blog;

Cuando queramos entrar en un blog en este caso sale erro 404, para solucionarlo creamos una carpeta en pages, que se llame blog y lo nombramos [id].js eso hará que sea una id dinámica y no poner ids de manera manual.

OJO! Para crear una llamada a nuestro API de Strapi, solo podemos hacerlo en la carpeta “pages”.

FILTRO CON STRAPI DE NUESTROS PROPIOS PRODUCTOS

export async function getServerSideProps() {

  const url = `${process.env.API\_URL}/guitarras?\_sort=precio:desc`;

  const respuesta = fetch(url);

  const guitarras = await (await respuesta).json();

  return {

    props: {

      guitarras,

    },

  };

}

En donde vemos la url, podemos añadir al final “?\_sort=precio:desc”