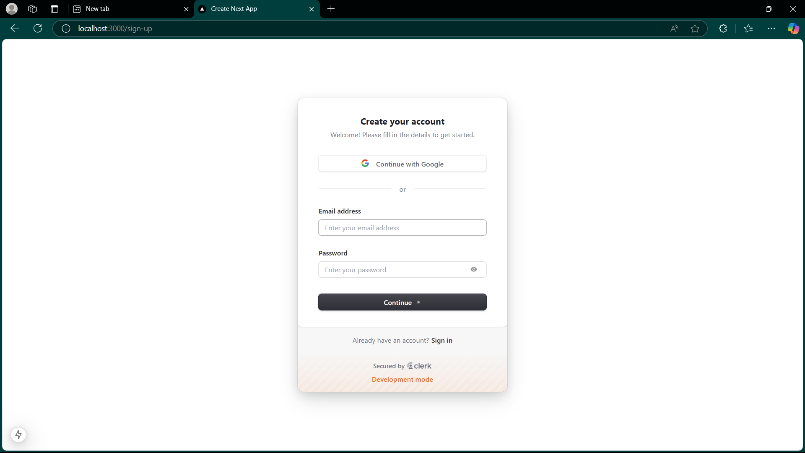
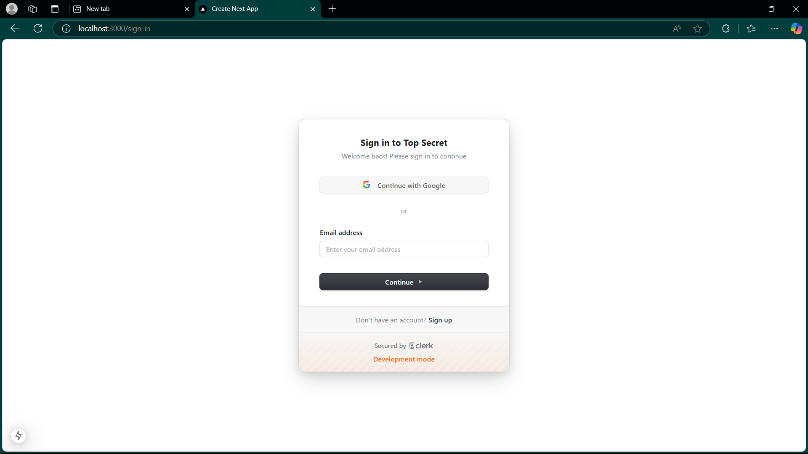
Nama : Maulidan Akmal Gandi

NPM : 22312137

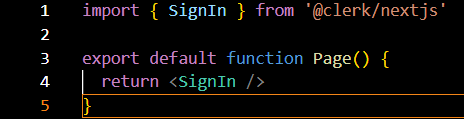
Dokumentasi Project Pemograman Web 2

1. Pembuatan Tampilan Sign-In dan Sign-Up menggunakan Clerk

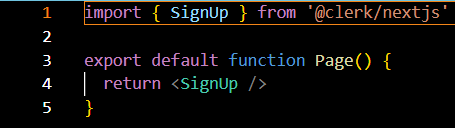




Sign-In

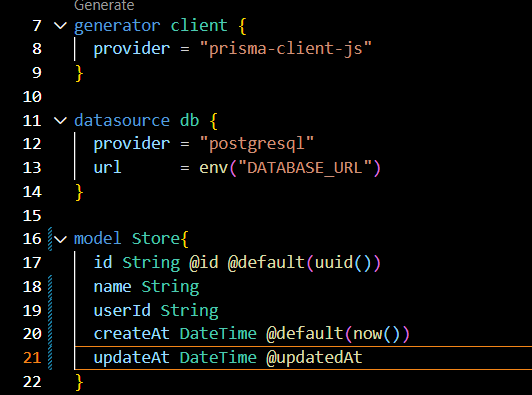


Sign-Up



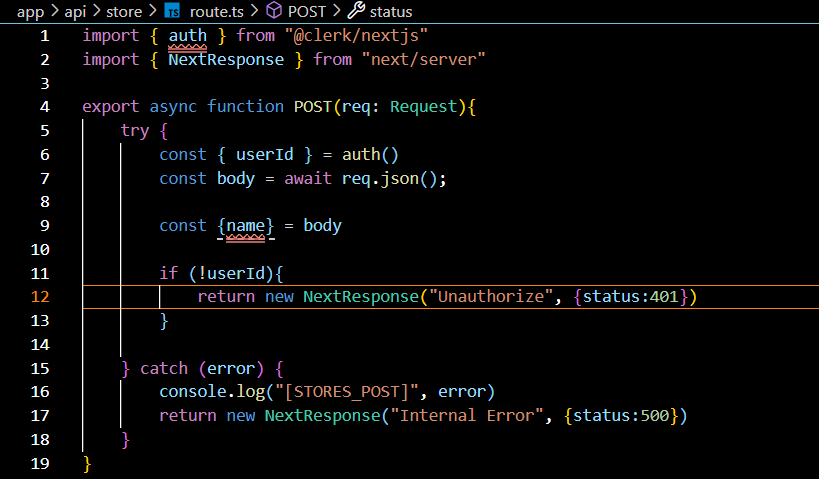
Digunakan untuk membuat akun dan masuk ketika mempunyai akun dan masuk kedalam website yang sudah dibuat

1. Membuat Database

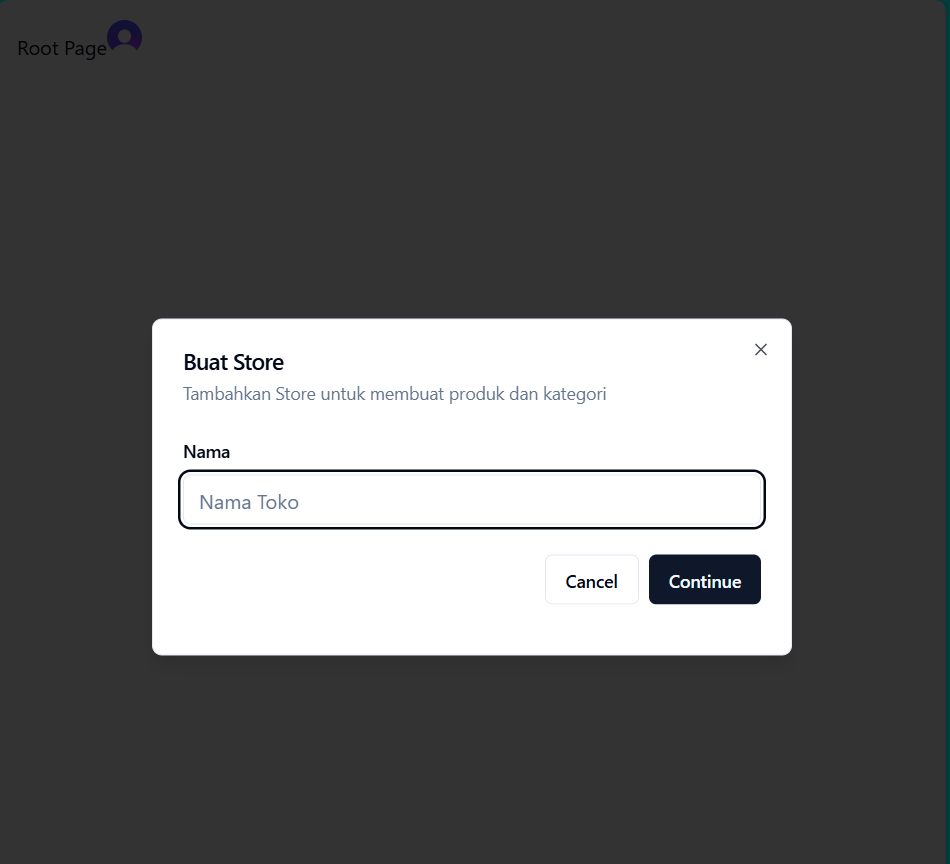


Membuat database dengan menggunakan provider “postgresql” dan table dengan nama “Store”

1. Membuat API dan Memperbaiki page.tsx di dalam folder app/(root)



Pembuatan API(Belum Selesai)

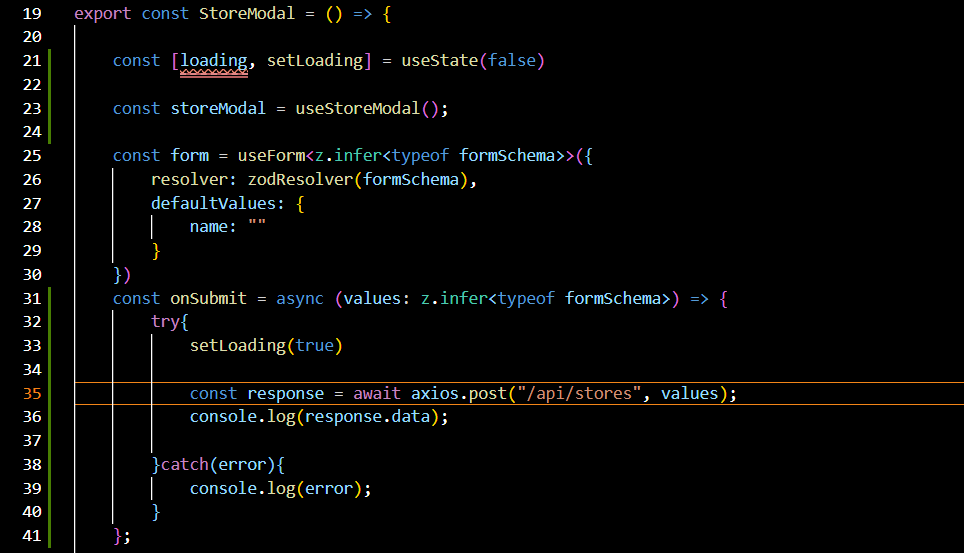


Diatas adalah tampilan page.tsx dalam folder app/(root) yang sudah diperbaiki dan akan muncul tampilan Buat Store

1. Memperbaiki API agar dapat menambahkan data ke dalam database, menambahkan fungsi agar bisa check data apakah ada data yang belum terisi, dan juga membuat fungsi untuk dapat memasukkan data kedalam prisma



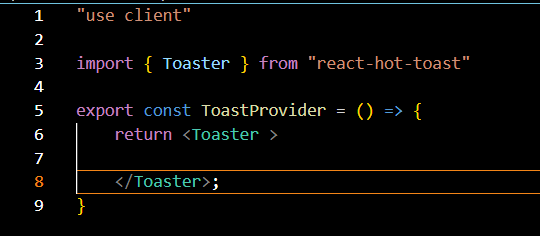
Ini adalah kode untuk API



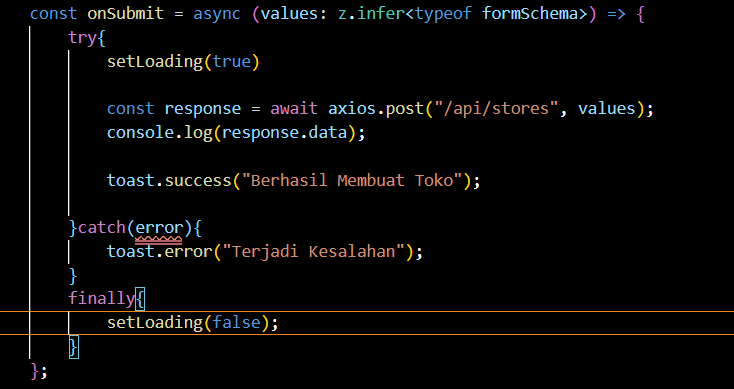
Ini adalah fungsi untuk memasukan data kedalam prisma menggunakan axios

1. Membuat provider toast dan memberikan alert jika terjadi kesalahan saat membuat toko dan membuat alert ketika berhasil membuat toko

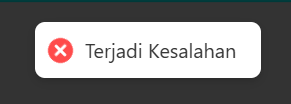
Membuat Provider Toast



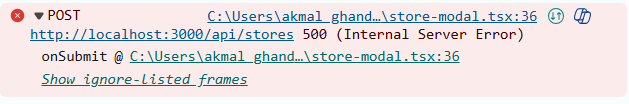
Fungsi untuk membuat alert menggunakan toast



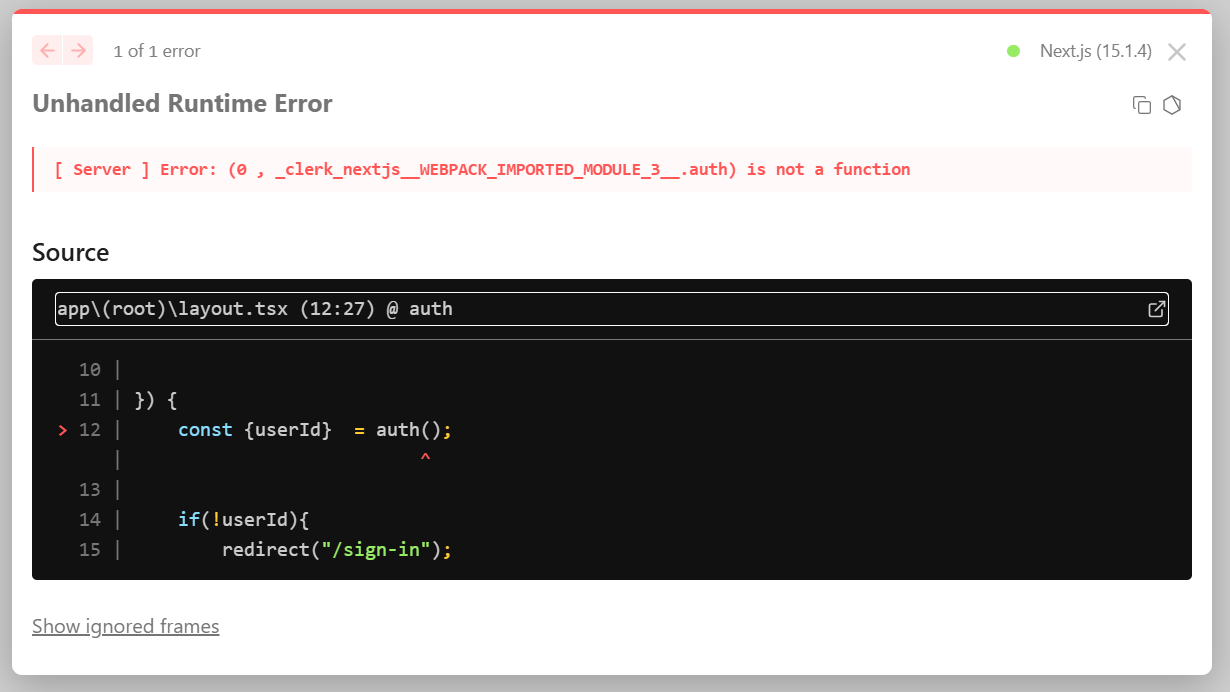
Hasil alert Ketika terjadi kesalahan saat membuat toko



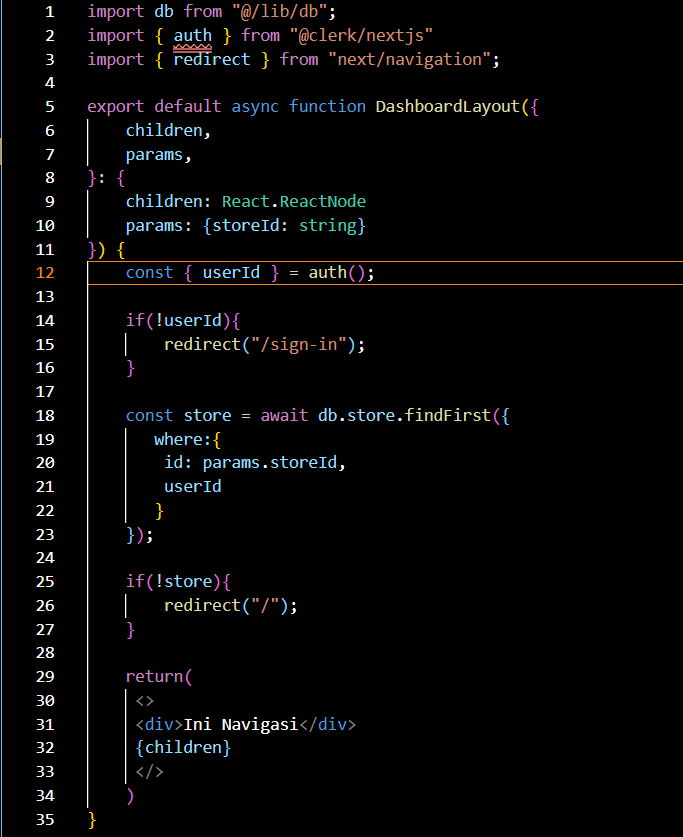
Dan hasil hasil alert Ketika berhasil membuat toko terjadi error karena post tidak terbaca di on submit



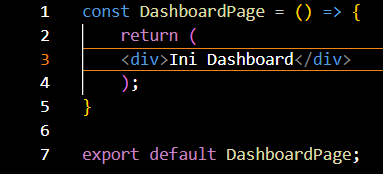
1. Membuat dashboard layout dan membuat root layout agar setelah membuat toko langsung dapat menuju dashboard layout



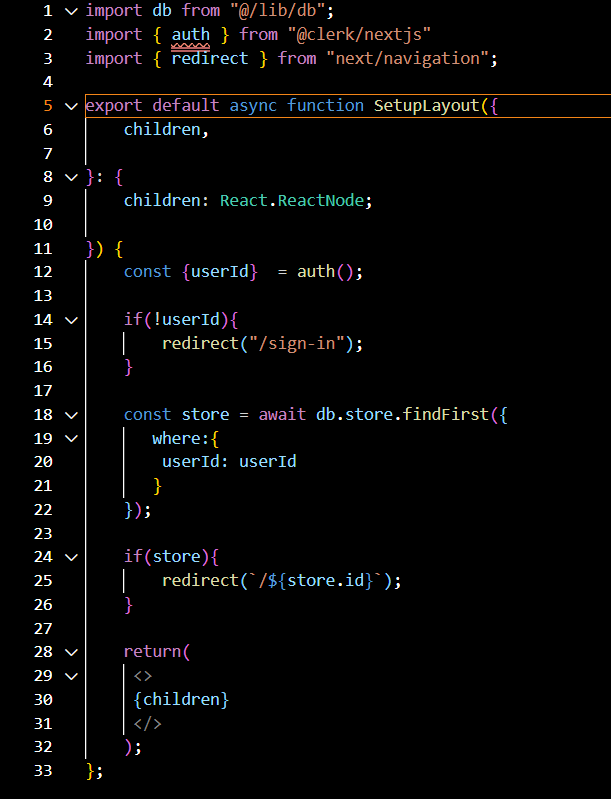
Belum dapat terbaca dikarenakan error pada auth di dalam file layout tsx pada folder (root).



Berikut adalah kode di dalam folder Dashboard/storeid yang didalamnya diberi nama DashboardLayout

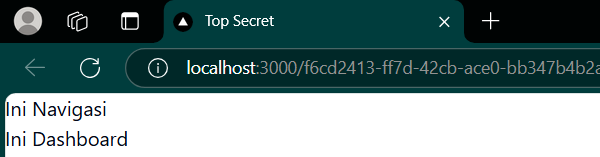


Berikut adalah kode di dalam folder Dashboard/storied/routes yang didalamnya diberi nama DashboardPage

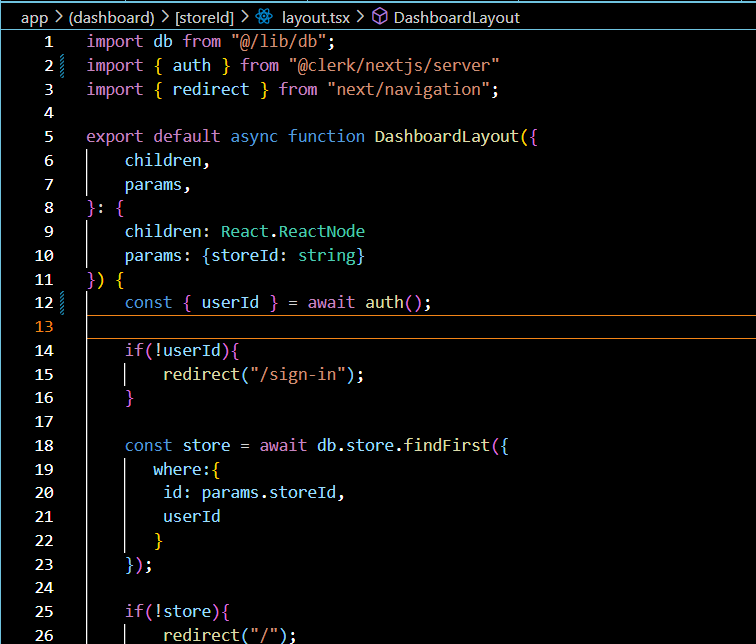


Berikut adalah kode di dalam folder root yang didalamnya diberi nama SetupLayout

1. Memperbaiki layout pada dashboard layout dan setup layout agar memunculkan tampilan halamannya



berikut adalah tampilan dashboard layout

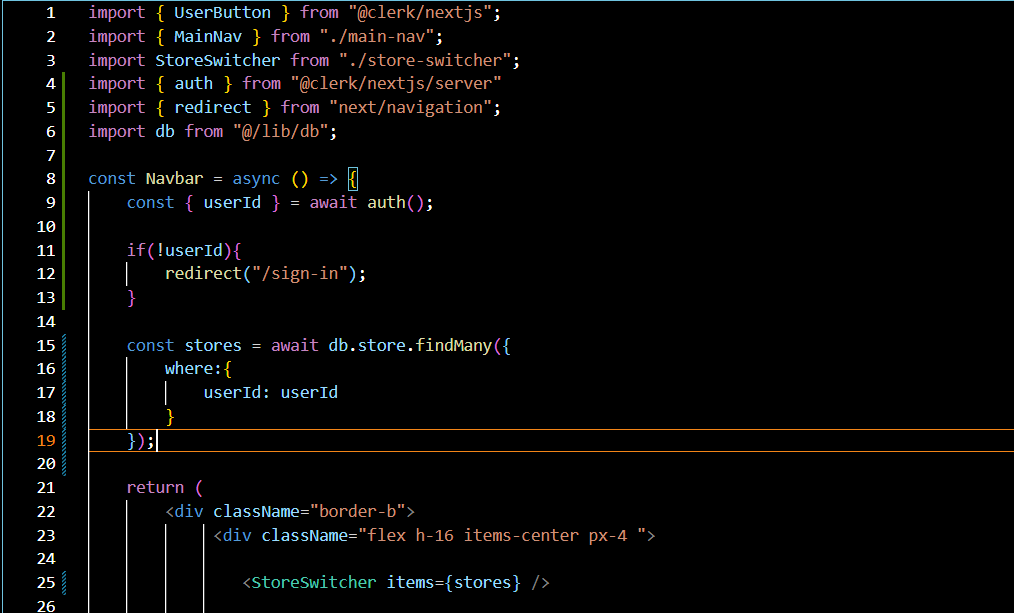


Berikut kode yang sudah diperbaiki pada baris 12 dan baris 2 di dalam dashboard layout

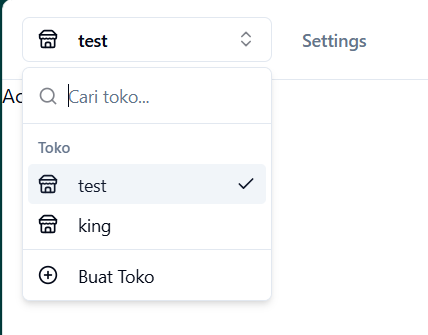


Berikut kode yang sudah diperbaiki pada baris 12 dan baris 2 di dalam setup layout

1. Memperbaiki bar navigasi agar bisa menampilkan toko yang sudah di buat dan bisa membuat toko

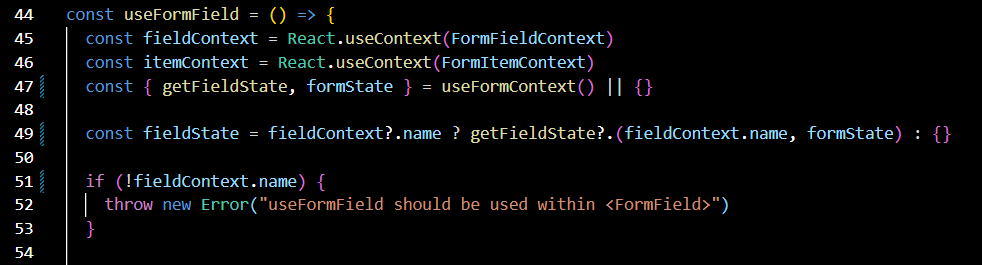


Berikut fungsi kode yang diperbaiki dan digunakan untuk menampilkan toko yang sudah dibuat dan bisa membuat toko di bar navigasi



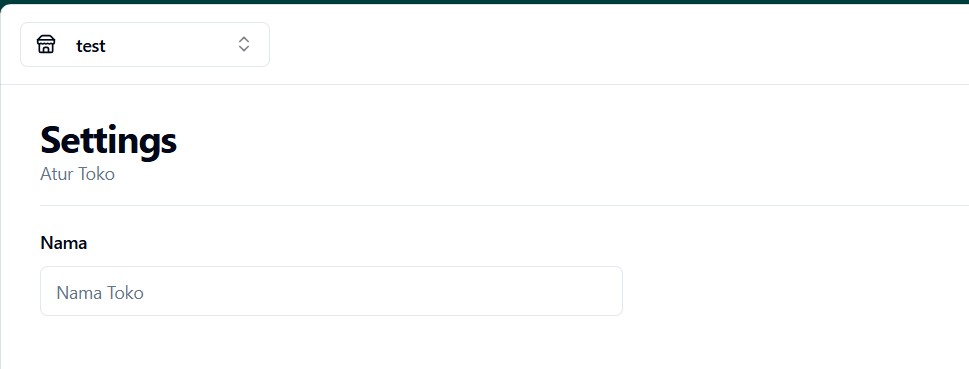
Berikut bar navigasi yang dibuat yang menampilkan nama toko yang sudah dibuat dan bisa membuat toko

1. Memperbaiki fungsi untuk menu settings agar bisa menampilkan apa yang ada di dalam menu settings



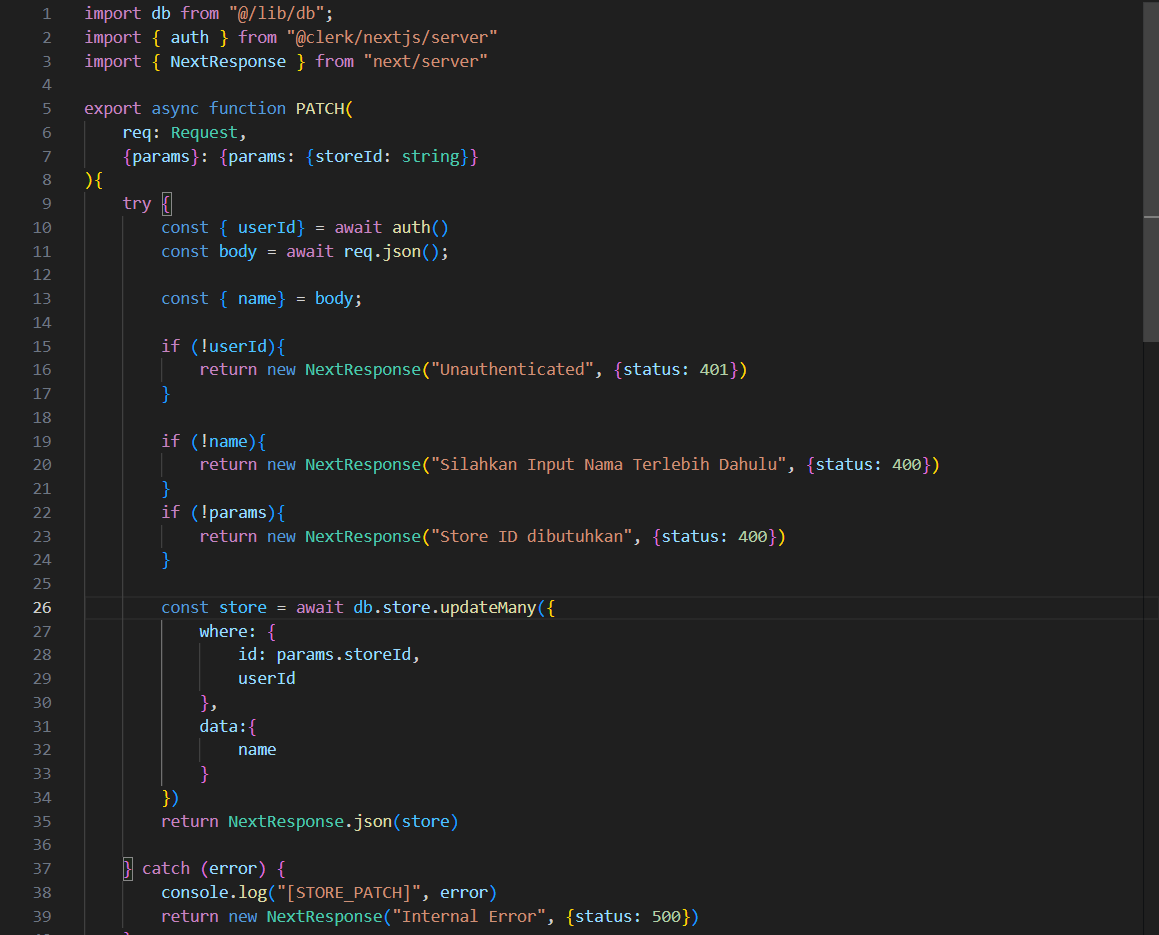


Berikut adalah kode yang diperbaiki agar menampilkan tampilan menu settings

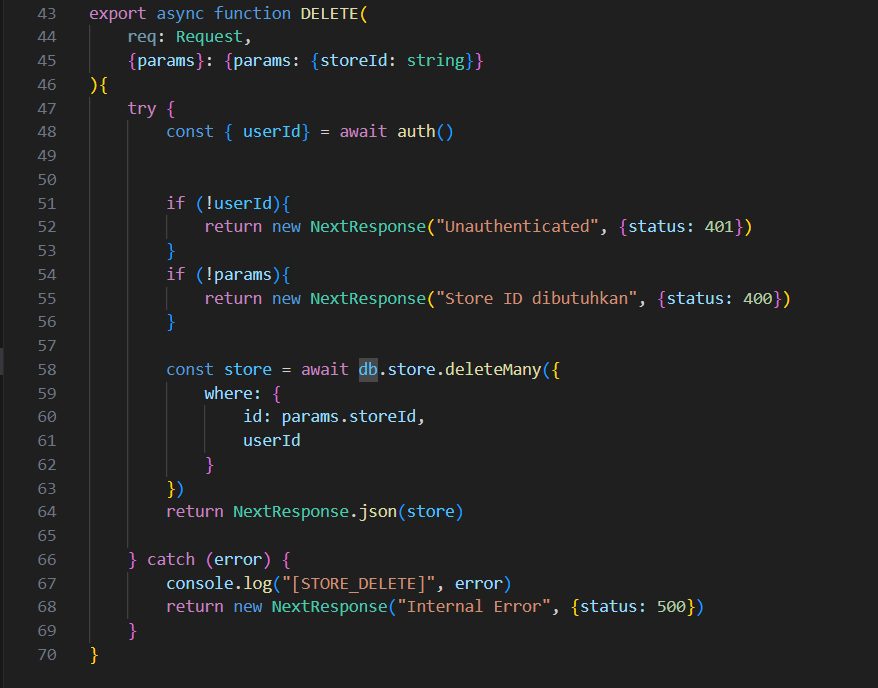


Berikut adalah tampilan menu settings yang sudah ditampilkan tetapi masih ada yang perlu ditambahkan

1. Membuat fungsi untuk update data dan hapus data

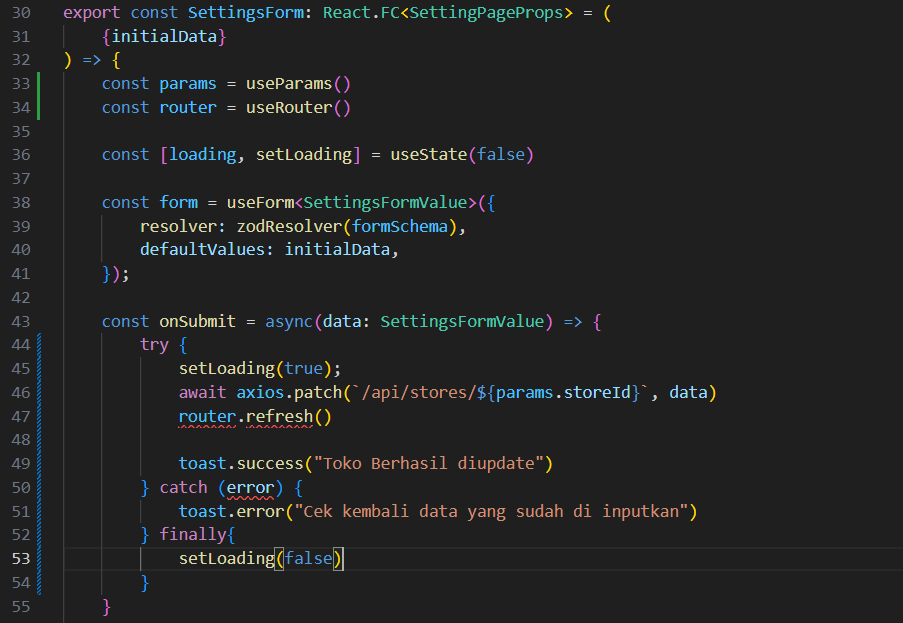


Berikut adalah fungsi untuk mengupdate data

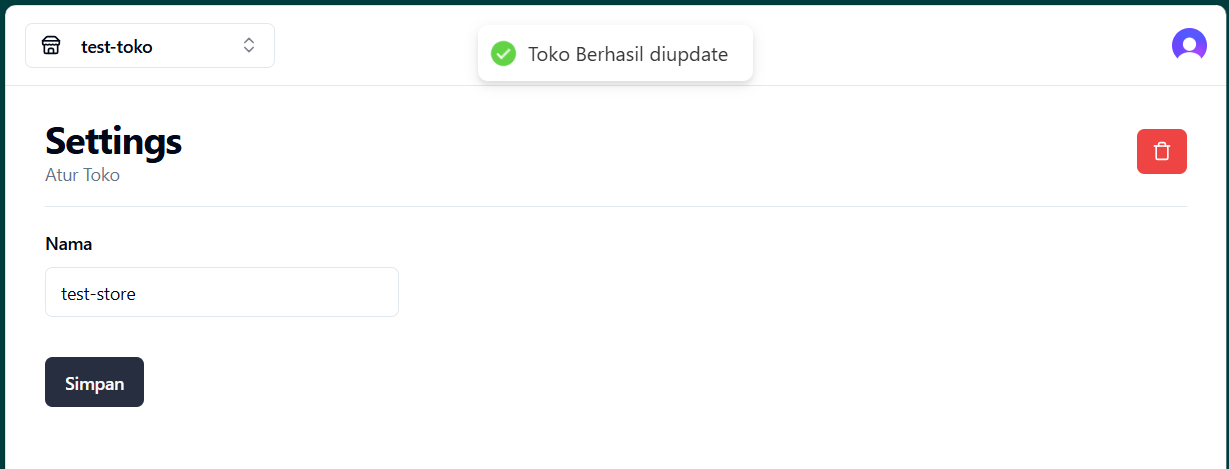


Berikut adalah fungsi untuk menghapus data

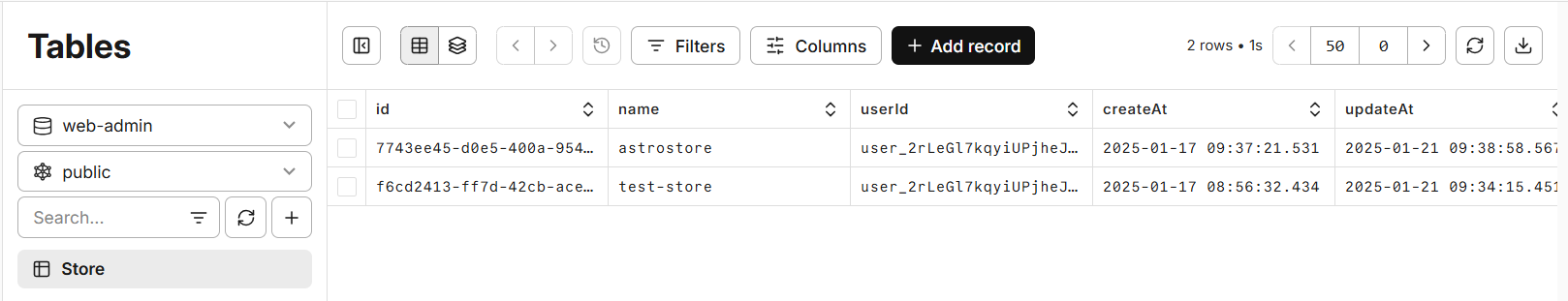
1. Menarik fungsi untuk mengupdate dan menghapus toko dari folder api/stores/[storeId]



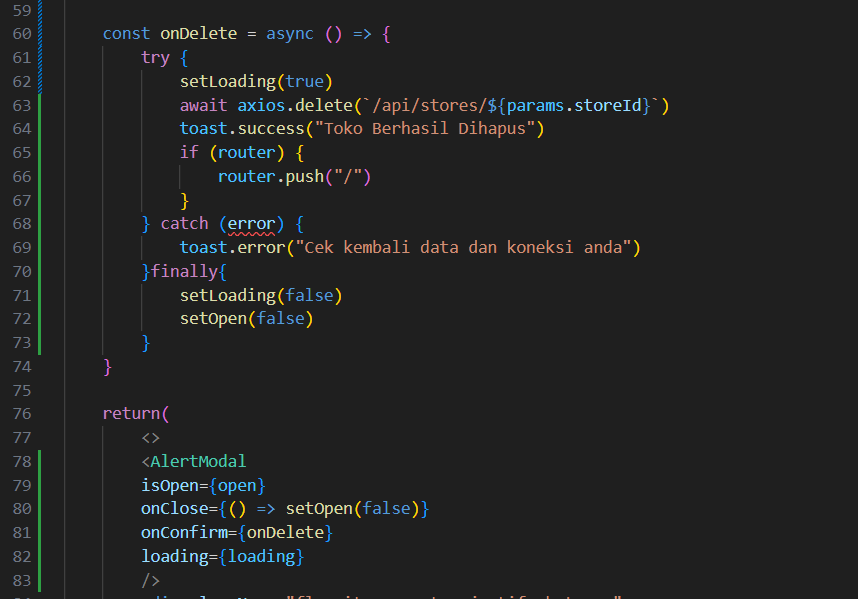
Berikut kode yang dibuat untuk menarik fungsi mengupdate toko yang sudah diinputkan



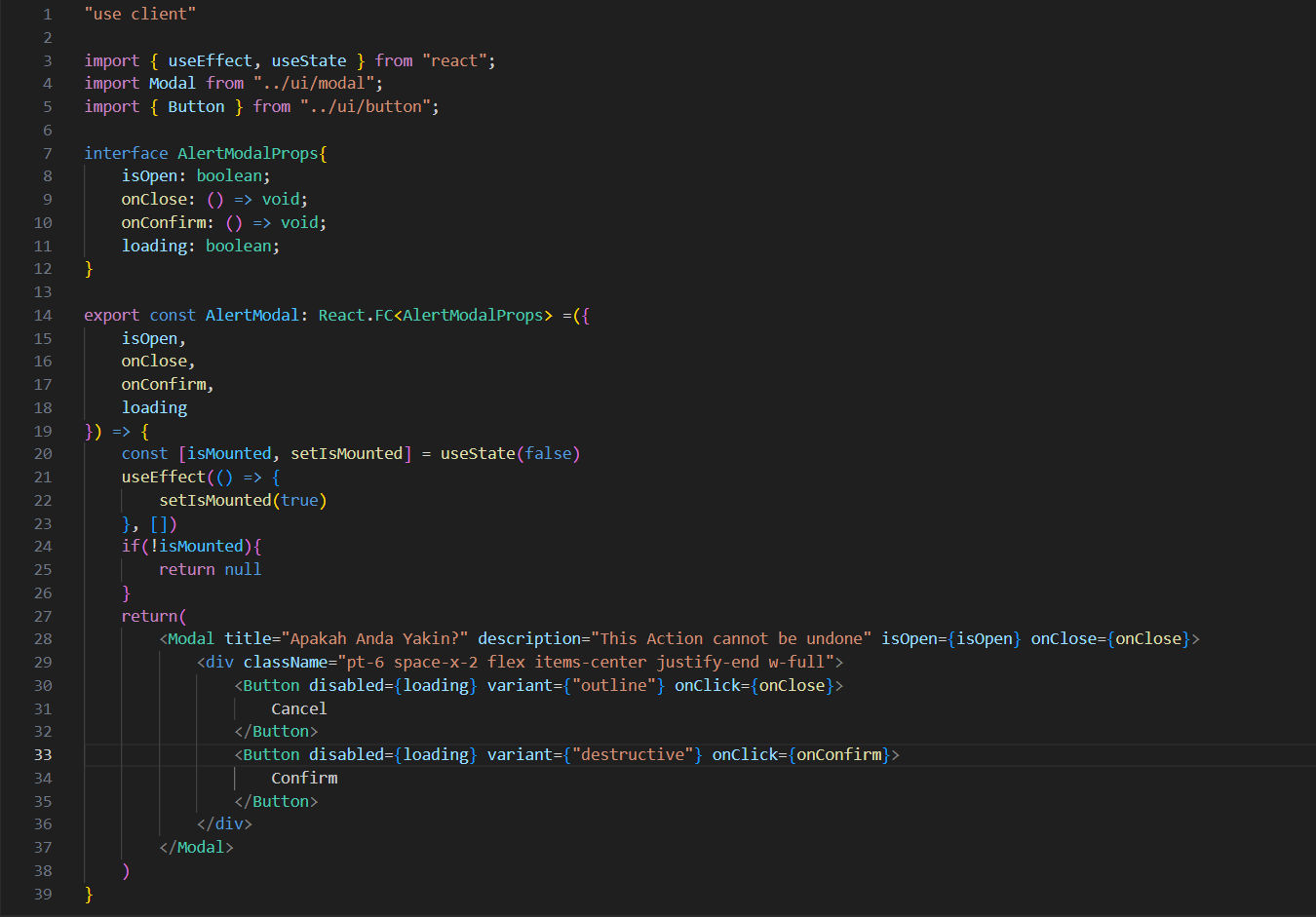
Berikut adalah tampilan Ketika berhasil mengupdate toko



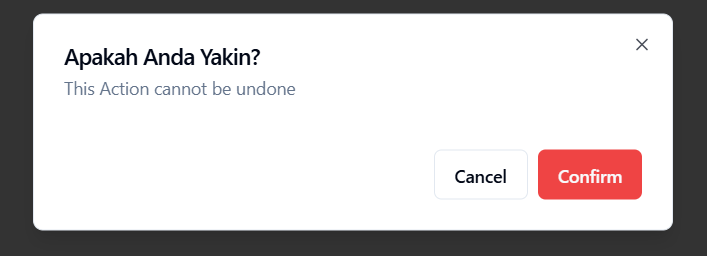
Berikut tampilan data yang sudah di ubah di dalam database



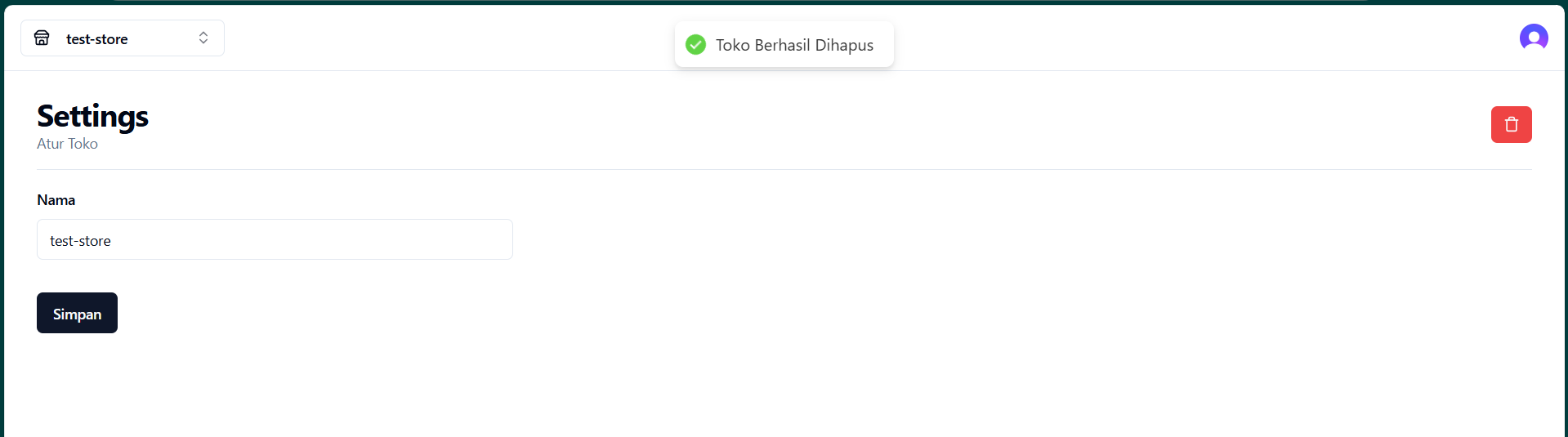
Berikut kode yang dibuat untuk menarik fungsi menghapus toko yang sudah diinputkan



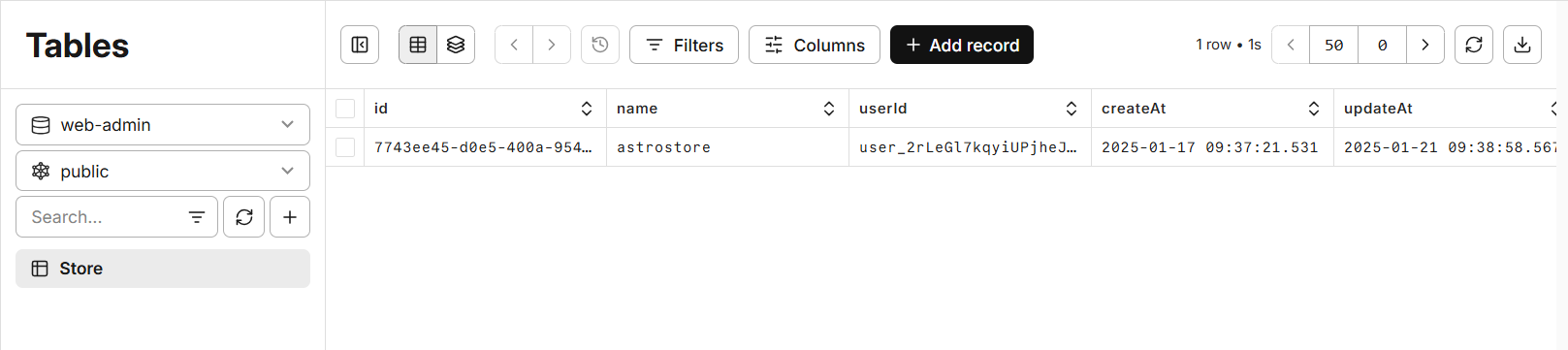
Berikut adalah kode alert yang dibuat untuk menampilkan alert mengkonfirmasi atau membatalkan menghapus toko



Berikut adalah alert untuk mengkonfirmasi dan membatalkan untuk menghapus toko



Berikut adalah tampilan data yang sudah berhasil dihapus

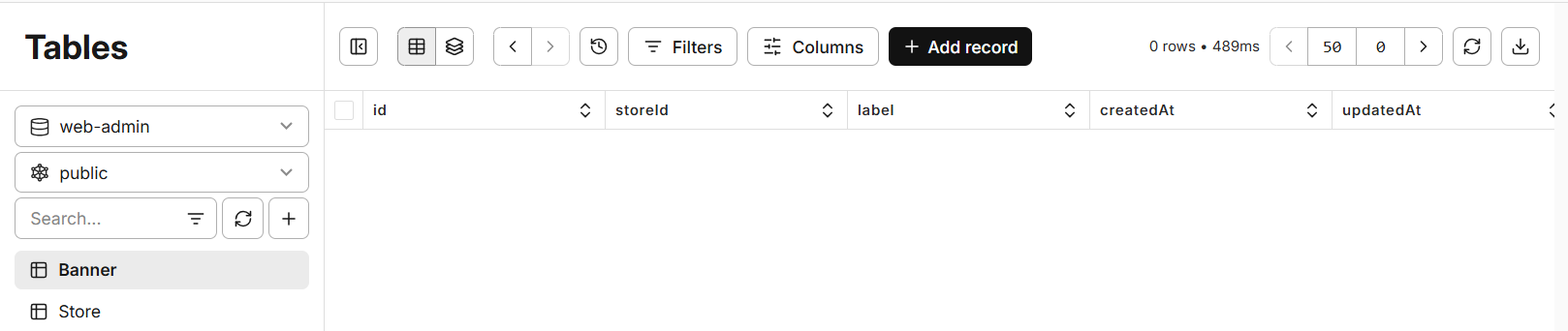


Berikut adalah tampilan data yang sudah hilang di dalam database Ketika berhasil dihapus di website

1. Menambahkan table Banner di dalam database



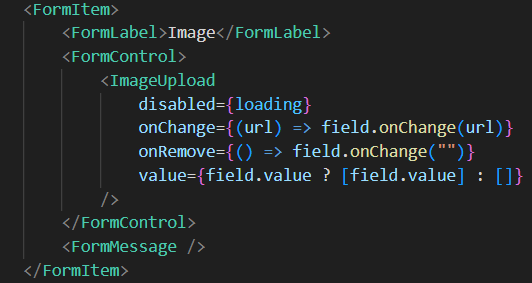
Berikut adalah kode untuk Membuat table Banner di dalam database



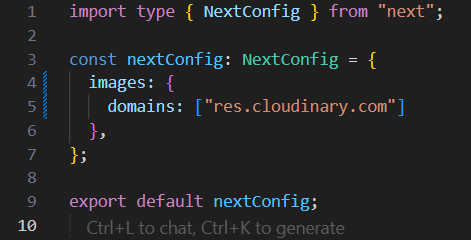
Berikut adalah tampilan table Banner di dalam database web-admin

1. Membuat tombol dan fungsi untuk upload gambar Banner di navigasi banner lalu klik add new

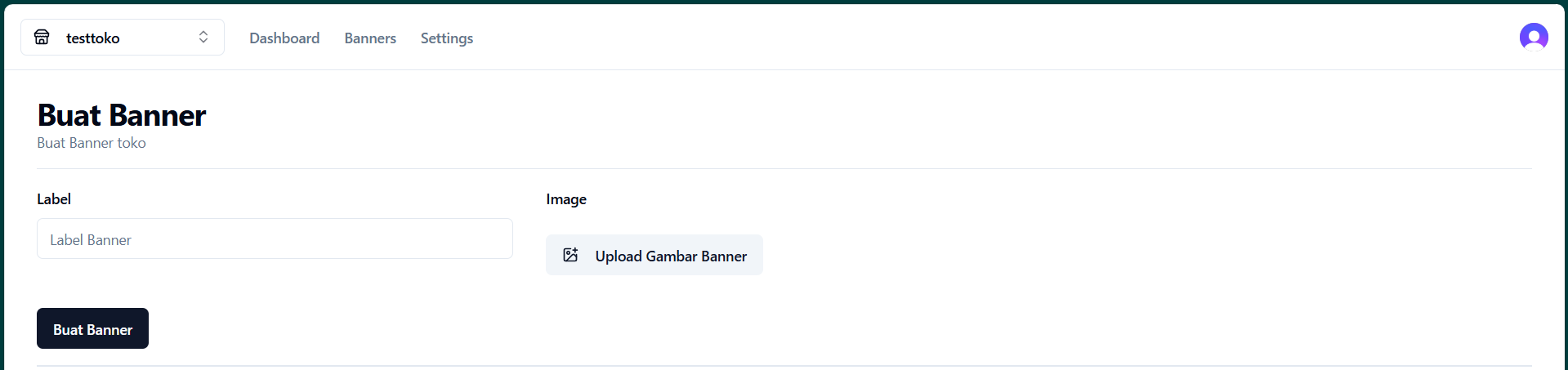
|  |
| --- |
| "use client";  import { useEffect, useState } from "react";  import { Button } from "./button";  import { ImagePlus, Trash } from "lucide-react";  import Image from "next/image";  import { CldUploadWidget } from "next-cloudinary";  interface ImageUploadProps {    disabled?: boolean;    onChange: (value: string) => void;    onRemove: (value: string) => void;    value: string[];  }  const ImageUpload: React.FC<ImageUploadProps> = ({    disabled,    onChange,    onRemove,    value,  }) => {    const [isMounted, setIsMounted] = useState(false);    useEffect(() => {      setIsMounted(true);    }, []);    const onUpload = (result: any) => {      onChange(result.info.secure\_url);    };    if (!isMounted) {      return null;    }    return (      <div>        <div className="mb-4 flex items-center gap-4">          {value &&            value.map((url) => (              <div                key={url}                className="relative w-[200px] h-[200px] rounded-md overflow-hidden"              >                <div className="z-10 absolute top-2 right-2">                  <Button                    type="button"                    onClick={() => onRemove(url)}                    variant={"destructive"}                    size={"icon"}                  >                    <Trash className="h-4 w-4"></Trash>                  </Button>                </div>                <Image                  fill                  className="object-cover"                  alt="Image"                  src={url}                ></Image>              </div>            ))}        </div>        <CldUploadWidget onSuccess={onUpload} uploadPreset="tscret20">          {({open}) => {            const onClick = () => {              open();            };            return (              <Button                type="button"                disabled={disabled}                variant={"secondary"}                onClick={onClick}              >                <ImagePlus className="h-4 w-4 mr-2"></ImagePlus>                Upload Gambar Banner              </Button>            );          }}        </CldUploadWidget>      </div>    );  };  export default ImageUpload; |



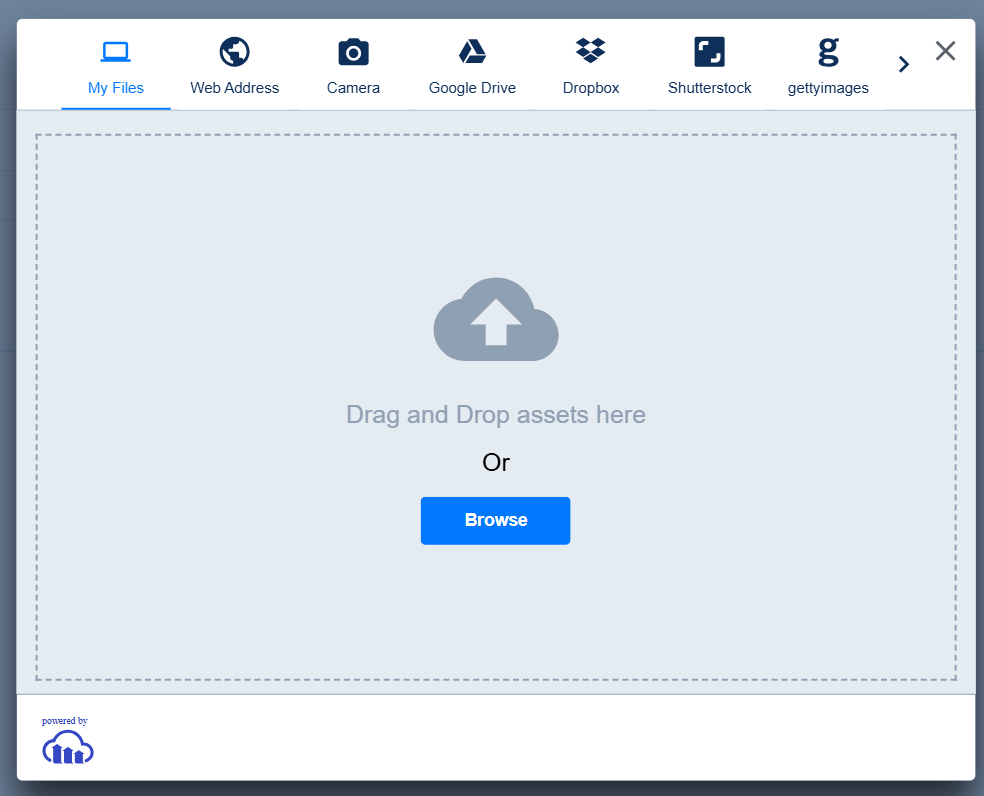
Berikut adalah code untuk Membuat fungsi dan tombol Upload Gambar Banner



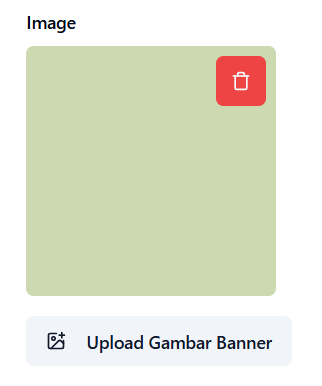
Berikut adalah kode untuk merubah configurasi agar cloudinary terbaca oleh next config



Berikut hasil tombol untuk upload gambar banner di navigasi add new banner



Berikut adalah tempat upload banner menggunakan cloudinary



Berikut adalah gambar banner yang dipilih

1. Membuat fungsi untuk membuat banner dan menghapus banner



Berikut adalah kode fungsi untuk membuat dan menghapus banner

1. Membuat fungsi untuk check data banner apakah sudah diisi atau belum

import db from "@/lib/db";

import { auth } from "@clerk/nextjs/server";

import { NextResponse } from "next/server";

export async function POST(req: Request,

    {params} : {params: {storeId: string}}) {

  try {

    const { userId } = await auth();

    const body = await req.json();

    const { label, imageUrl } = body;

    if (!userId) {

      return new NextResponse("Unauthorize", { status: 401 });

    }

    if (!label) {

      return new NextResponse("Nama Banner Perlu di Tambahkan", { status: 400 });

    }

    if (!imageUrl) {

        return new NextResponse("Image Banner Perlu di Tambahkan", { status: 400 });

    }

    if(!params.storeId){

        return new NextResponse("Store id URL dibutuhkan")

    }

    const storeByUserId = await db.store.findFirst({

        where: {

            id: params.storeId,

            userId: userId

        }

    })

    if(!storeByUserId){

        return new NextResponse("Unautorized", {status: 403});

    }

    const banner = await db.banner.create({

      data: {

        label,

        imageUrl,

        storeId: params.storeId

      },

    });

    return NextResponse.json(banner);

  } catch (error) {

    console.error("[BANNERS\_POST]", error);

    return new NextResponse(JSON.stringify({ message: "Internal Error" }), { status: 500 });

  }

}

export async function GET(req: Request,

    {params} : {params: {storeId: string}}) {

  try {

    if(!params.storeId){

        return new NextResponse("Store id URL dibutuhkan")

    }

    const banner = await db.banner.findMany({

      where: {

        storeId: params.storeId

      }

    });

    return NextResponse.json(banner);

  } catch (error) {

    console.error("[BANNERS\_GET]", error);

    return new NextResponse(JSON.stringify({ message: "Internal Error" }), { status: 500 });

  }

}

Berikut adalah kode untuk membuat fungsi check data banner

1. Membuat fungsi untuk menyimpan data banner ke dalam database dan terbaca table di halaman banner yang sudah dibuat

'use client'

import \* as z from 'zod'

import { use, useState } from 'react'

import { Button } from "@/components/ui/button"

import { Heading } from "@/components/ui/heading"

import { Separator } from "@/components/ui/separator"

import { Store } from "@prisma/client"

import { Trash } from "lucide-react"

import { useForm } from 'react-hook-form'

import { zodResolver } from '@hookform/resolvers/zod'

import { Form, FormControl, FormField, FormItem, FormLabel, FormMessage } from '@/components/ui/form'

import { Input } from '@/components/ui/input'

import toast from 'react-hot-toast'

import axios from 'axios'

import { useParams } from 'next/navigation'

import { useRouter } from 'next/compat/router'

import { AlertModal } from '@/components/modals/alert-modal'

import { ApiAlert } from '@/components/ui/api-alert'

import { UseOrigin } from '@/hooks/use-origin'

import ImageUpload from '@/components/ui/image-upload'

interface BannerFormProps {

    initialData: Banner | null;

}

const formSchema = z.object({

    label: z.string().min(1),

    imageUrl: z.string().min(1)

})

type BannerFormValue = z.infer<typeof formSchema>

export const BannerForm: React.FC<BannerFormProps> = (

    {initialData}

) => {

    const params = useParams();

    const router = useRouter();

    const origin = UseOrigin();

    const [open, setOpen] = useState(false);

    const [loading, setLoading] = useState(false);

    const title = initialData ? "Edit Banner" : "Buat Banner"

    const description = initialData ? "Edit Banner toko" : "Buat Banner toko"

    const toastMassage = initialData ? "Banner Berhasil di edit" : "Banner Berhasil dibuat"

    const action = initialData ? "Simpan Banner" : "Buat Banner"

    const form = useForm<BannerFormValue>({

        resolver: zodResolver(formSchema),

        defaultValues: initialData || {

            label: '',

            imageUrl: ''

        }

    });

    const onSubmit = async(data: BannerFormValue) => {

        try {

            setLoading(true);

            if (initialData) {

                await axios.patch(`/api/${params.storeId}/banners/${params.bannerId}`, data);

            }else{

                await axios.post(`/api/${params.storeId}/banners`, data);

            }

            toast.success("Toko Berhasil diupdate");

            if (router) {

                router.refresh("/")

            }

        } catch (error) {

            toast.error("Cek kembali data yang sudah di inputkan");

        } finally {

            setLoading(false);

        }

    };

    const onDelete = async () => {

        try {

            setLoading(true)

            await axios.delete(`/api/${params.storeId}/banners/${params.bannerId}`);

            toast.success("Banner Berhasil Dihapus")

            if (router) {

                router.push("/")

            }

        } catch (error) {

            toast.error("Cek kembali data dan koneksi anda")

        }finally{

            setLoading(false)

            setOpen(false)

        }

    }

    return(

        <>

        <AlertModal

        isOpen={open}

        onClose={() => setOpen(false)}

        onConfirm={onDelete}

        loading={loading}

        />

        <div className="flex items-center justify-between">

            <Heading

                title={title}

                description={description}

            />

            {initialData &&(

            <Button

            disabled={loading}

            variant="destructive"

            size="sm"

            onClick={() => setOpen(true)}

            >

                <Trash className="h-4 w-4"/>

            </Button>

            )}

        </div>

        <Separator />

        <Form {...form}>

        <form onSubmit={form.handleSubmit(onSubmit)} className="space-y-8 w-full">

            <div className="grid grid-cols-3 gap-8">

                <FormField

                    control={form.control}

                    name="label"

                    render={({field}) => (

                        <FormItem>

                            <FormLabel>Label</FormLabel>

                            <FormControl>

                                <Input placeholder="Label Banner" disabled={loading} value={field.value} onChange={field.onChange} />

                            </FormControl>

                            <FormMessage />

                        </FormItem>

                    )}

                />

                <FormField

                    control={form.control}

                    name="imageUrl"

                    render={({field}) => (

                        <FormItem>

                            <FormLabel>Image</FormLabel>

                            <FormControl>

                                <ImageUpload

                                    disabled={loading}

                                    onChange={(url: string) => field.onChange(url)}

                                    onRemove={() => field.onChange("")}

                                    value={field.value ? [field.value] : []}

                                />

                            </FormControl>

                            <FormMessage />

                        </FormItem>

                    )}

                />

            </div>

            <Button

            disabled={loading}

type='submit'

            >

                {action}

            </Button>

        </form>

        </Form>

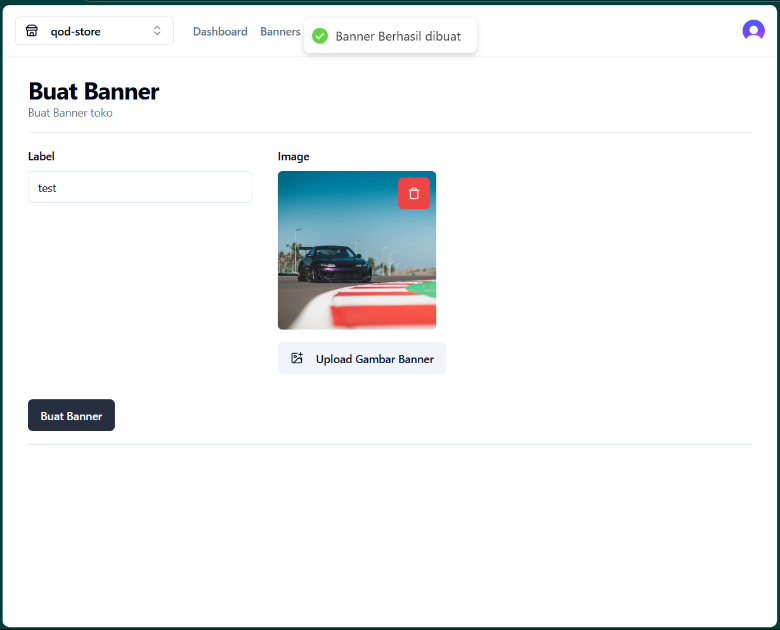
        <Separator />

        </>

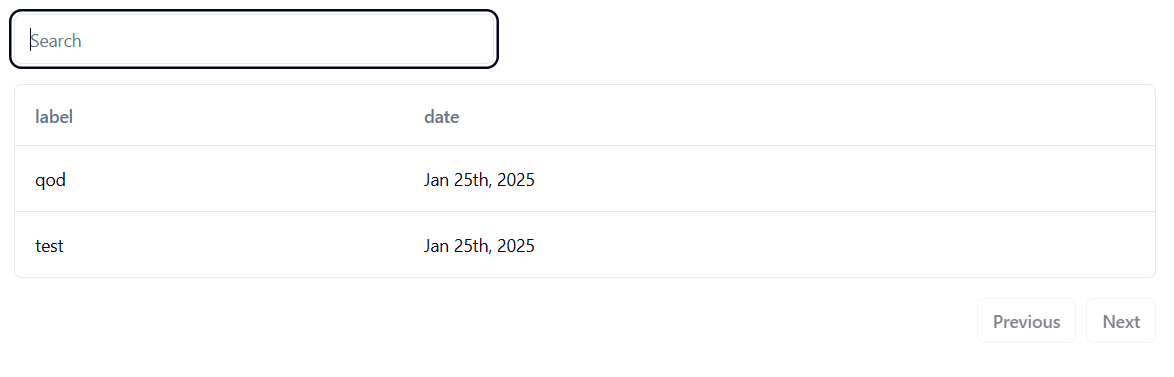
    );

};

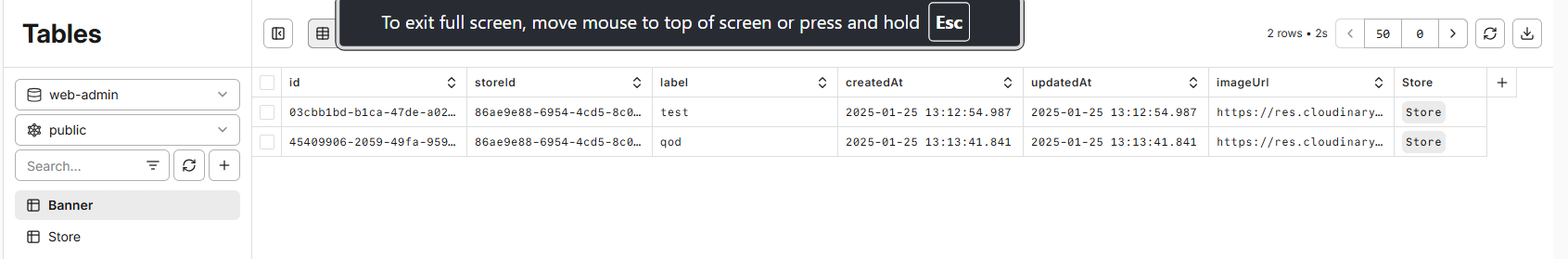
Berikut adalah fungsi untuk tombol agar di klik akan menyimpan data banner



Berikut adalah tampilan jika sudah dapat mengupload banner

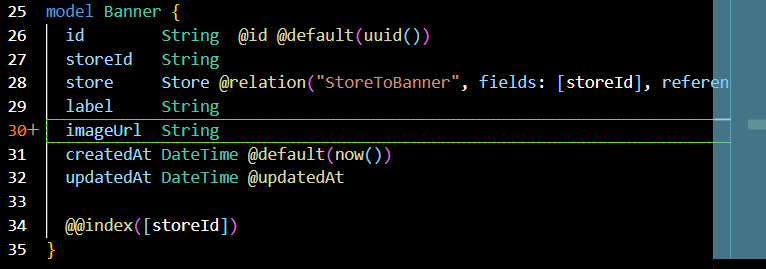


Berikut adalah tampilan table data banner yang sudah di buat



Berikut adalah data banner yang masuk Ketika di klik Buat Banner

1. Update database menambahkan imageUrl di dalam database



Berikut adalah kode imageUrl yang ditambahkan di schema.prisma dan sudah di push ke dalam database