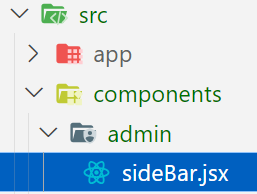
Atelier 2 NextJS

Dashboard Admin

# SideBar menu

Créer sous le répertoire /src/components un dossier intitulé admin.

Dans ce dossier créer un composant appelé sideBar.jsx



On va se référer à ce site pour le code correspondant à ce composant :

<https://www.npmjs.com/package/react-pro-sidebar>

Installer le package :

npm i react-pro-sidebar

Les icônes à partir de ce site :

<https://mui.com/material-ui/material-icons/>

'use client';

import { Sidebar, Menu, MenuItem } from 'react-pro-sidebar';

import Link from 'next/link';

import AdminPanelSettingsIcon from '@mui/icons-material/AdminPanelSettings';

import DashboardIcon from '@mui/icons-material/Dashboard';

import ArticleIcon from '@mui/icons-material/Article';

import CategoryIcon from '@mui/icons-material/Category';

import ContentCopyIcon from '@mui/icons-material/ContentCopy';

import GroupIcon from '@mui/icons-material/Group';

import ReceiptIcon from '@mui/icons-material/Receipt';

import CalendarMonthIcon from '@mui/icons-material/CalendarMonth';

import BarChartIcon from '@mui/icons-material/BarChart';

import LightModeIcon from '@mui/icons-material/LightMode';

const SideBar = () => {

  return (

    <Sidebar>

        <Menu

            menuItemStyles={{

            button: ({ level, active, disabled }) => {

                // only apply styles on first level elements of the tree

                if (level === 0)

                return {

                    color: disabled ? '#b4c2c0' : '#718280',

                    backgroundColor: active ? '#4f5d73' : '#071121',

                };

            },

            }}

        >

      <MenuItem>

<div className="card" style={{"backgroundColor": "#44ebe2"}}>

        <div className="card-body">

          <AdminPanelSettingsIcon /> Administrateur

        </div>

      </div>

      </MenuItem>

      <MenuItem component={<Link **href**="/admin/dashboard" />}> <DashboardIcon/> Dashboard </MenuItem>

      <MenuItem component={<Link **href**="/admin/products" />}> <ArticleIcon/> Produits </MenuItem>

      <MenuItem component={<Link **href**="/admin/categories" />}> <CategoryIcon/> Catégories </MenuItem>

      <MenuItem component={<Link **href**="/admin/scategories" />}> <ContentCopyIcon/> Sous/Catégories</MenuItem>

      <MenuItem component={<Link **href**="/admin/users" />}> <GroupIcon/> Utilisateurs </MenuItem>

      <MenuItem component={<Link **href**="/admin/orders" />}> <ReceiptIcon/> Commandes </MenuItem>

      <MenuItem component={<Link **href**="/admin/calendars" />}> <CalendarMonthIcon/> Calendrier </MenuItem>

      <MenuItem component={<Link **href**="/admin/chart" />}> <BarChartIcon/> Chart </MenuItem>

      <MenuItem component={<Link **href**="/admin/theme" />}> <LightModeIcon/> Thème </MenuItem>

    </Menu>

  </Sidebar>

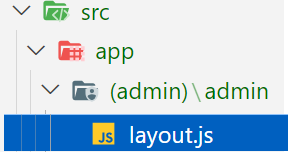
  )

}

export default SideBar

Ensuite, créer sous /src/app le dossier (admin) contenant le dossier admin puis le fichier layout.js

Layout est une interface utilisateur partagée entre les routes.



import SideBar from "@/components/admin/sideBar";

export default function AdminLayout({ children }) {

  return (

    <div className="row mt-4">

    <div className="col-md-3">

    <SideBar />

    </div>

    <div className="col-md-9">

    {children}

    </div>

    </div>

  )

}

Ainsi, le sideBar sera visible dans tous les enfants de admin. On lui réservé 3 colonnes et les 9 autres pour l’enfant qui sera appelé via les de sidebar.

Avec mt-4 on a défini une marge en haut égale à 4.

# Comment créer la page d'analyse

On va créer des cards pour représenter les totaux puis un graphique et un tableau de la liste des utilisateurs.

Pour le graphique on va utiliser la bibliothèque recharts.js

L'objectif principal de cette bibliothèque est de vous aider à rédiger des graphiques dans des applications React sans aucune difficulté.

Pour l’installer :

npm install recharts

Site : <https://recharts.org/en-US/>

# Le composant cardsAnalytics

Créer sous /src/components/admin le dossier dashboard puis le fichier cardsAnalytics



'use client'

const CardsAnalytics = () => {

  return (

    <>

      <section>

        <div className="flex m-4 gap-2">

          <div className="flex-1 px-2 justify-center w-16 bg-gray-700 shadow rounded h-300px">

            <div className="">

              <p className="text-white font-bold">Total returns</p>

              <p className="py-4 font-bold text-yellow-500">50,000 TND</p>

              <p className="text-green-300">+24.5%</p>

            </div>

          </div>

          <div className="flex-1 px-2 justify-center w-16 bg-gray-700 shadow rounded max-h-300px">

            <div className="">

              <p className="text-white font-bold">Total sales</p>

              <p className="py-4 font-bold text-yellow-500">30,000 TND</p>

              <p className="text-green-300">+34.5%</p>

            </div>

          </div>

          <div className="flex-1 px-2 justify-center w-16  bg-gray-700 shadow rounded max-h-300px">

            <div className="">

              <p className="text-white font-bold">Total subscriptions</p>

              <p className="py-4 font-bold text-yellow-500">20,000 TND</p>

              <p className="text-green-300">+24.5%</p>

            </div>

          </div>

          <div className="flex-1 px-2 justify-center w-16  bg-gray-700 shadow rounded h-300px">

            <div className="">

              <p className="text-white font-bold">Total orders</p>

              <p className="py-4 font-bold text-yellow-500">60,000 TND</p>

              <p className="text-red-300">-14.5%</p>

            </div>

          </div>

        </div>

      </section>

      <section className="flex m-4 gap-2">

        <div className="w-full h-[300px] bg-gray-700 rounded"></div>

      </section>

      <section className="flex m-4 gap-2">

        <div className="w-full h-[250px] bg-gray-700 rounded"></div>

      </section>

    </>

  );

};

export default CardsAnalytics;

Flexbox : Tailwind facilite la mise en place des modèles de disposition flexbox avec des classes comme flex, justify-center, et gap-2. Ces classes vous aident à organiser les éléments en ligne et à les aligner au centre.

Marges et Espacements : Les classes comme m-4 et gap-2 contrôlent les marges et les espacements entre les éléments. Par exemple, m-4 ajoute une marge de 1.5 rem (24 pixels) autour de tous les côtés de l'élément, et gap-2 ajoute un espacement de 0.5 rem (8 pixels) entre les éléments enfants d'un conteneur flex.

Couleurs et Ombres : Les classes comme bg-gray-700, text-white, text-yellow-500, text-green-300, et text-red-300 définissent les couleurs de fond et de texte. Tailwind utilise un système de noms de couleurs cohérent et des nuances pour vous permettre de choisir facilement les couleurs souhaitées. De plus, les classes comme shadow ajoutent des ombres à un élément.

Taille et Dimensions : Les classes comme w-16, max-h-300px, h-[300px], et h-[250px] définissent les tailles et les dimensions des éléments. Tailwind utilise un système de noms de classe basé sur une échelle pour définir les tailles de manière cohérente et prévisible.

Arrondi des coins : Les classes comme rounded et shadow sont utilisées pour ajouter des coins arrondis à certains éléments, ce qui peut contribuer à adoucir l'apparence globale de votre interface utilisateur.

# Page Dashboard

Par la suite, sous /src/(admin)/admin, créer un dossier dashbord. Ce dernier contenant un fichier page.js



import React from 'react'

import CardsAnalytics from '@/components/admin/dashboard/cardsAnalytics'

const page = () => {

  return (

  <div className="container">

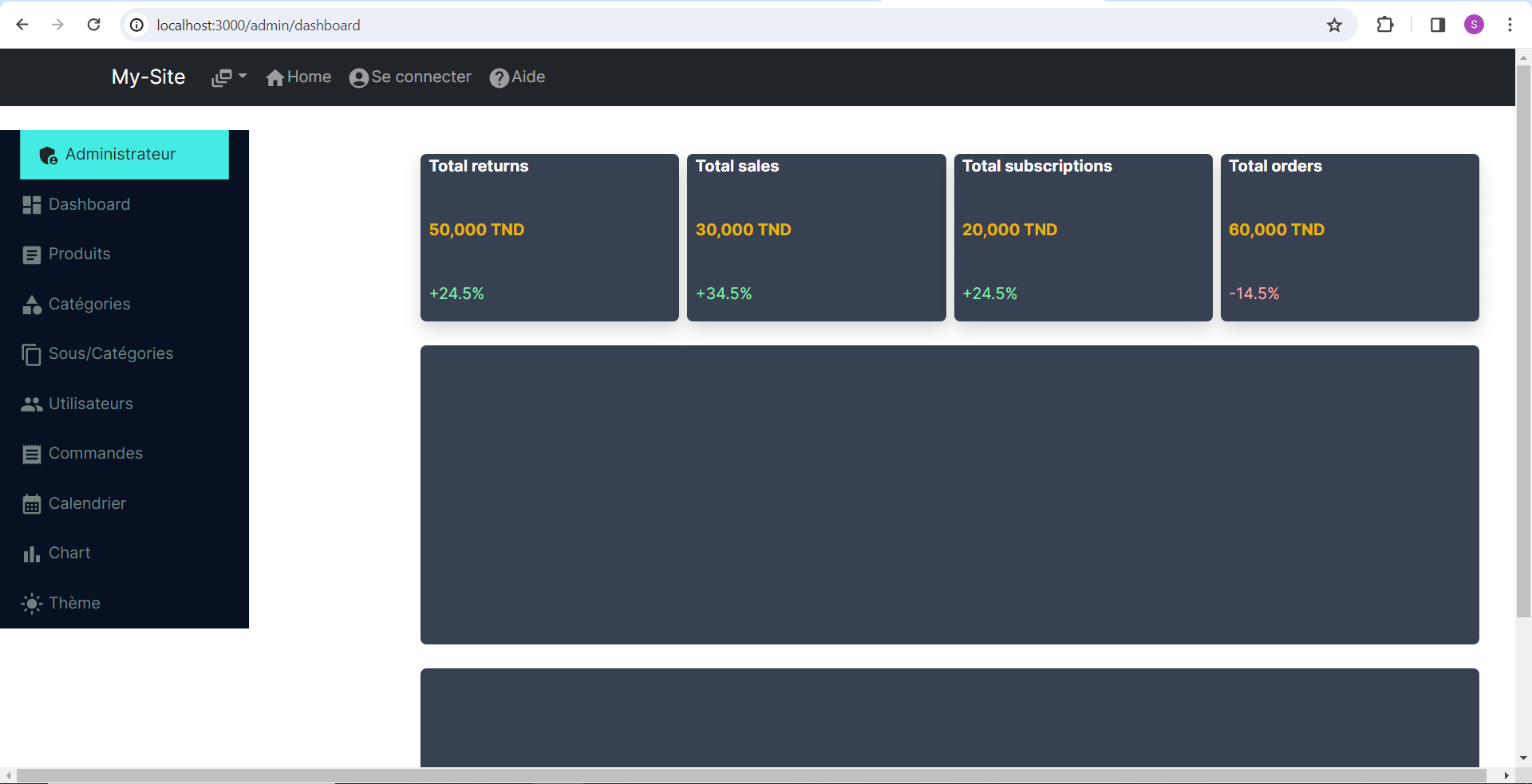
    <CardsAnalytics/>

</div>

  )

}

export default page



# Le composant areaChartPlot

Références :

<https://recharts.org/en-US/guide/getting-started>

<https://recharts.org/en-US/examples/StackedAreaChart>

Les composants Recharts ajoutés représentent différents éléments d'un graphique et sont les suivants :

AreaChart : composant principal représentant le graphique en aires.

Zone : les deux composants Zone définissent les zones à tracer sur le graphique. Dans les deux composants Area, nous transmettons la valeur des deux éléments que nous voulons représenter sur le graphique, Iphone et Samsung.

XAxis : représente l'axe des X du graphique. Nous transmettons la prop dataKey comme "year" pour indiquer que la propriété year de l'objet de données doit être utilisée comme valeur de l'axe des x.

Axe Y : représente l'axe Y du graphique.

Info-bulle : un composant général (ce qui signifie que vous pouvez l'utiliser avec d'autres types de graphiques) et personnalisable qui nous permet d'afficher des info-bulles lorsque vous survolez des points de données sur le graphique.

ResponsiveContainer : composant général qui permet de rendre le graphique réactif à différentes tailles d'écran et de l'adapter à son conteneur parent.

Créez ensuite un composant AreaChartPlot sous /src/components/admin/dashboard et définissez-y un tableau de données. Ce seront les données que nous tracerons sur le graphique en aires.



import React from 'react'

import { AreaChart, Area, XAxis, YAxis, Tooltip, ResponsiveContainer } from 'recharts';

const AreaChartPlot = () => {

    const data = [

        {

          "year": "2016",

          "Iphone": 4000,

          "Samsung": 2400

        },

        {

          "year": "2017",

          "Iphone": 3000,

          "Samsung": 1398

        },

        {

          "year": "2018",

          "Iphone": 2000,

          "Samsung": 9800

        },

        {

          "year": "2019",

          "Iphone": 2780,

          "Samsung": 3908

        },

        {

          "year": "2020",

          "Iphone": 1890,

          "Samsung": 4800

        },

        {

          "year": "2021",

          "Iphone": 2390,

          "Samsung": 3800

        },

        {

          "year": "2022",

          "Iphone": 3490,

          "Samsung": 4300

        }

      ]

  return (

    <ResponsiveContainer width="100%" height="100%" >

    <AreaChart width={730} height={250} data={data}

      margin={{ top: 10, right: 30, left: 0, bottom: 0 }}>

      <defs>

        <linearGradient id="colorUv" x1="0" y1="0" x2="0" y2="1">

          <stop offset="5%" stopColor="#8884d8" stopOpacity={0.8} />

          <stop offset="95%" stopColor="#8884d8" stopOpacity={0} />

        </linearGradient>

        <linearGradient id="colorPv" x1="0" y1="0" x2="0" y2="1">

          <stop offset="5%" stopColor="#82ca9d" stopOpacity={0.8} />

          <stop offset="95%" stopColor="#82ca9d" stopOpacity={0} />

        </linearGradient>

      </defs>

      <XAxis dataKey="year" />

      <YAxis />

      <Tooltip />

      <Area type="monotone" dataKey="Iphone" stroke="#8884d8" fillOpacity={1} fill="url(#colorUv)" />

      <Area type="monotone" dataKey="Samsung" stroke="#82ca9d" fillOpacity={1} fill="url(#colorPv)" />

    </AreaChart>

  </ResponsiveContainer>

  )

}

export default AreaChartPlot

Données du graphique :

const data = [...]: Définit un tableau d'objets JavaScript contenant les données du graphique. Chaque objet représente une année avec le nombre de ventes pour les appareils iPhone et Samsung.

Rendu du composant :

<ResponsiveContainer> : Ce composant fournit un conteneur réactif pour le graphique, ce qui signifie que le graphique s'ajustera automatiquement à la taille de son conteneur parent.

<AreaChart> : Ce composant représente le graphique à aires. Il prend en charge plusieurs propriétés, notamment width et height pour définir les dimensions du graphique et data pour spécifier les données à afficher.

<defs> : Cette balise contient les définitions des dégradés utilisés pour colorer les aires du graphique.

<XAxis> et <YAxis> : Ces composants représentent respectivement les axes X et Y du graphique, définissant les échelles et les étiquettes des axes.

<Tooltip> : Ce composant active l'affichage d'une info-bulle lorsque l'utilisateur survole les points du graphique, affichant des informations supplémentaires sur les données.

<Area> : Ces composants représentent les aires du graphique. Chaque Area est définie par un ensemble de données dataKey (correspondant à la clé dans les données) et des propriétés spécifiant les couleurs de remplissage (fill), les couleurs de contour (stroke) et d'autres styles.

<stop> est un élément SVG (Scalable Vector Graphics) utilisé pour définir les points d'arrêt dans un dégradé linéaire ou radial. Voici une explication de chaque attribut utilisé dans <stop> :

offset="5%": Cet attribut spécifie la position du point d'arrêt le long du dégradé. Dans ce cas, le point d'arrêt est positionné à 5% de la longueur totale du dégradé. Cela signifie que la couleur spécifiée commencera à être appliquée à partir de 5% de la longueur du dégradé.

stopColor="#8884d8": Cet attribut définit la couleur du point d'arrêt. Dans cet exemple, la couleur utilisée est un bleu défini par le code hexadécimal #8884d8.

stopOpacity={0.8}: Cet attribut définit l'opacité du point d'arrêt. Une valeur de 0 signifie complètement transparent, tandis qu'une valeur de 1 signifie complètement opaque. Dans ce cas, l'opacité est définie à 0.8, ce qui signifie que la couleur du point d'arrêt sera légèrement transparente, permettant à une partie de l'arrière-plan de transparaître.

Puis modifier cardsAnalytics pour faire appel à areaChartPlot.



'use client'

import AreaChartPlot from "./areaChartPlot";

const CardsAnalytics = () => {

  return (

    <>

      <section>

        <div className="flex m-4 gap-2">

          <div className="flex-1 px-2 justify-center w-16 bg-gray-700 shadow rounded h-300px">

            <div className="">

              <p className="text-white font-bold">Total returns</p>

              <p className="py-4 font-bold text-yellow-500">50,000 TND</p>

              <p className="text-green-300">+24.5%</p>

            </div>

          </div>

          <div className="flex-1 px-2 justify-center w-16 bg-gray-700 shadow rounded max-h-300px">

            <div className="">

              <p className="text-white font-bold">Total sales</p>

              <p className="py-4 font-bold text-yellow-500">30,000 TND</p>

              <p className="text-green-300">+34.5%</p>

            </div>

          </div>

          <div className="flex-1 px-2 justify-center w-16  bg-gray-700 shadow rounded max-h-300px">

            <div className="">

              <p className="text-white font-bold">Total subscriptions</p>

              <p className="py-4 font-bold text-yellow-500">20,000 TND</p>

              <p className="text-green-300">+24.5%</p>

            </div>

          </div>

          <div className="flex-1 px-2 justify-center w-16  bg-gray-700 shadow rounded h-300px">

            <div className="">

              <p className="text-white font-bold">Total returns</p>

              <p className="py-4 font-bold text-yellow-500">60,000 TND</p>

              <p className="text-red-300">-14.5%</p>

            </div>

          </div>

        </div>

      </section>

      <section className="flex m-4 gap-2">

        <div className="w-full h-[300px] bg-gray-700 rounded">

            <AreaChartPlot />

        </div>

      </section>

      <section className="flex m-4 gap-2">

        <div className="w-full h-[250px] bg-gray-700 rounded"></div>

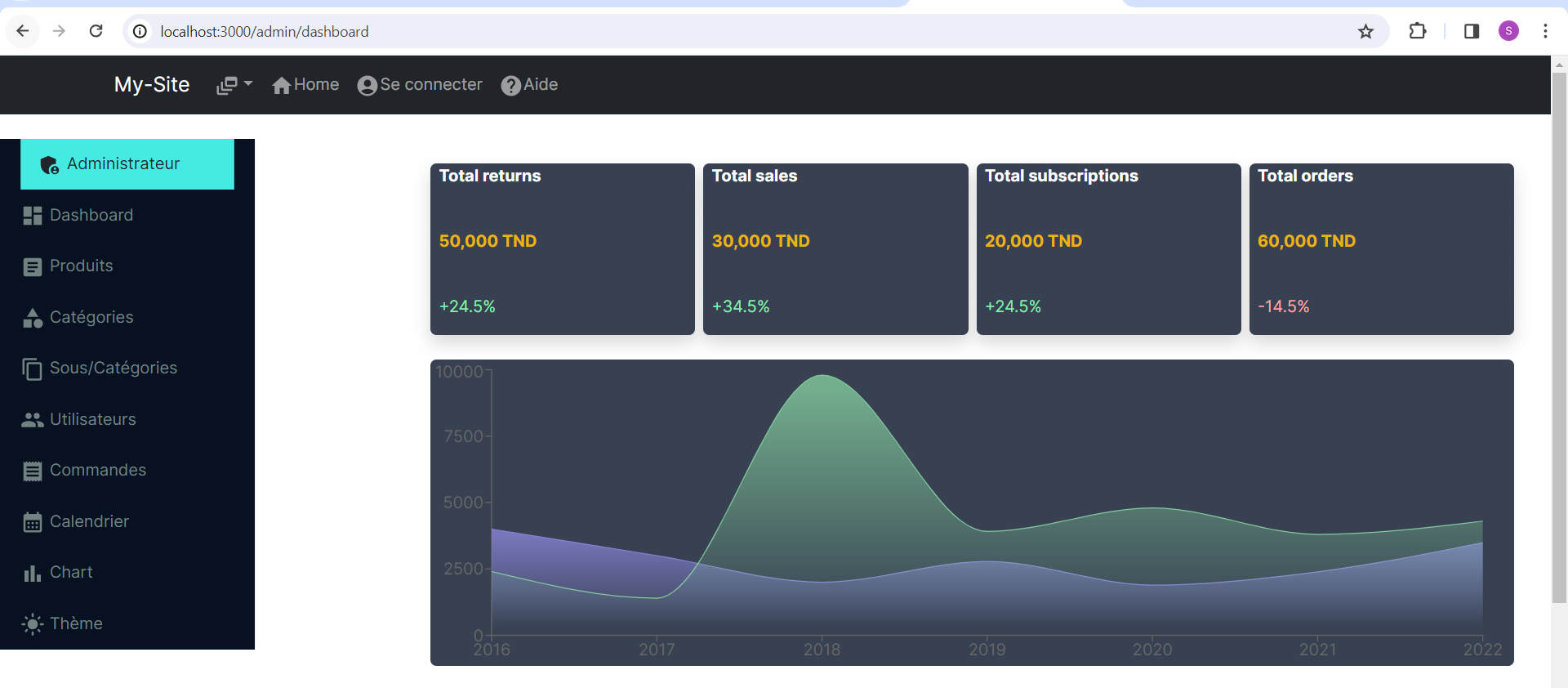
      </section>

    </>

  );

};

export default CardsAnalytics;



# Le composant areaUsersList

Abordons maintenant le tableau des utilisateurs. Ce tableau affichera les utilisateurs à partir de l’url : https://apigenerator.dronahq.com/api/9PdhJb9d/users

Il s’agit d’une fake API online

Dans ce composant on va afficher la liste des utilisateurs avec un affichage spécial pour rôle avec un badge coloré.



async function getUsers() {

    const res = await fetch('https://apigenerator.dronahq.com/api/9PdhJb9d/users')

    return res.json()

  }

export default async function AreaUsersList() {

    const users= await getUsers();

  return (

    <div className="card">

    <table className="table table-striped">

      <thead>

        <tr>

          <th>Avatar</th>

          <th>Name</th>

          <th>Email</th>

          <th>Role</th>

        </tr>

      </thead>

      <tbody>

        {users.map((user) => (

          <tr key={user.id}>

            <td>

              <img src={user.avatar} width="40" height="40" style={{ borderRadius: "50%" }} alt={user.name + "'s Avatar"} />

            </td>

            <td>{user.name}</td>

            <td>{user.email}</td>

            <td>

                {user.role === "admin" ? (

                  <span className="badge bg-pink-500 text-white">{user.role}</span>

                ) : (

                  <span className="badge bg-green-500 text-white">{user.role}</span>

                )}

            </td>

          </tr>

        ))}

      </tbody>

    </table>

  </div>

  );

}

Puis modifier cardsAnalytics en faisant appel à AreaUsersList



'use client'

import AreaChartPlot from "./areaChartPlot";

import AreaUsersList from "./areaUsersList";

const CardsAnalytics = () => {

  return (

    <>

      <section>

        <div className="flex m-4 gap-2">

          <div className="flex-1 px-2 justify-center w-16 bg-gray-700 shadow rounded h-300px">

            <div className="">

              <p className="text-white font-bold">Total returns</p>

              <p className="py-4 font-bold text-yellow-500">50,000 TND</p>

              <p className="text-green-300">+24.5%</p>

            </div>

          </div>

          <div className="flex-1 px-2 justify-center w-16 bg-gray-700 shadow rounded max-h-300px">

            <div className="">

              <p className="text-white font-bold">Total sales</p>

              <p className="py-4 font-bold text-yellow-500">30,000 TND</p>

              <p className="text-green-300">+34.5%</p>

            </div>

          </div>

          <div className="flex-1 px-2 justify-center w-16  bg-gray-700 shadow rounded max-h-300px">

            <div className="">

              <p className="text-white font-bold">Total subscriptions</p>

              <p className="py-4 font-bold text-yellow-500">20,000 TND</p>

              <p className="text-green-300">+24.5%</p>

            </div>

          </div>

          <div className="flex-1 px-2 justify-center w-16  bg-gray-700 shadow rounded h-300px">

            <div className="">

              <p className="text-white font-bold">Total orders</p>

              <p className="py-4 font-bold text-yellow-500">60,000 TND</p>

              <p className="text-red-300">-14.5%</p>

            </div>

          </div>

        </div>

      </section>

      <section className="flex m-4 gap-2">

        <div className="w-full h-[300px] bg-gray-700 rounded">

            <AreaChartPlot />

        </div>

      </section>

      <section className="flex m-4 gap-2">

        <div className="w-full h-[250px] bg-gray-700 rounded">

        <AreaUsersList />

        </div>

      </section>

    </>

  );

};

export default CardsAnalytics;

