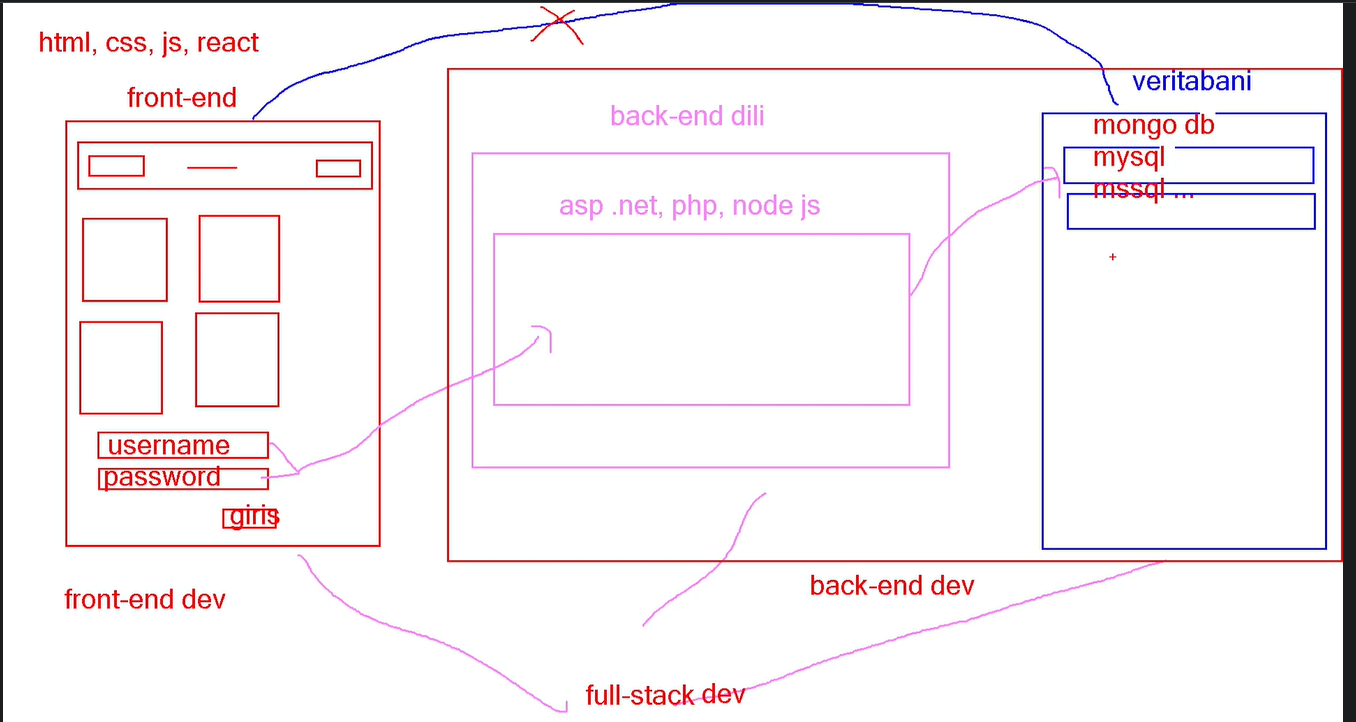
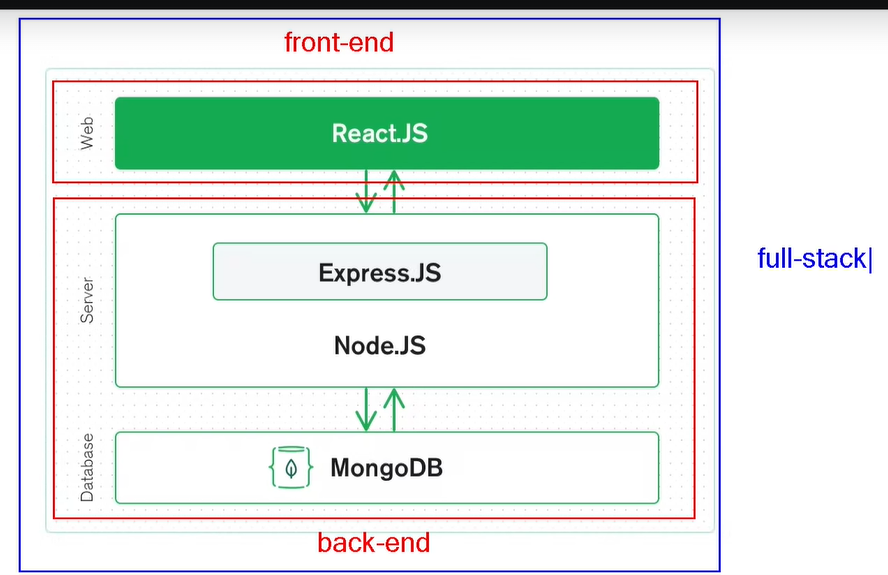
**MERN STACK**

* Full Stack Yazılımın Ana Hatları



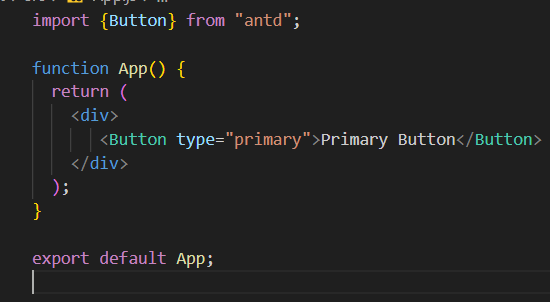
* MERN Stack Ana Hatları



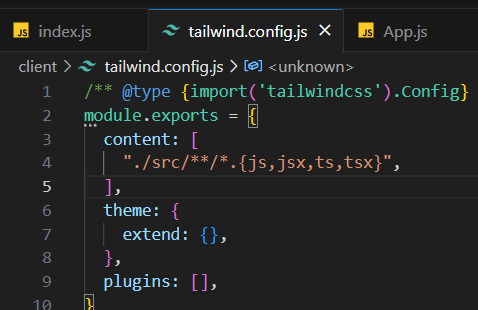
* ExpressJs NodeJs’İN http modülü üzerinde çalışan bir frameworktur.
* React frontend için kullanılabilecek bir javascript kütüphanesidir.
* NodeJs ile javascript dilini sunucu tarafında kullanmayı sağlıyoruz.
* Mongodb NOSQL bir veri tabanıdır verileri burada tutarız.
* Veri tabanında bulunan datalara erişmek veya düzenleme yapmak için bu olayı bir API a dökmemiz gerekiyor. Bu API da bulunan işlemler tamamen json data formatındadır.
* Örneğin, bir kullanıcının kaydolma formundaki bilgileri API'ye göndermek istediğinizi düşünelim. Bu veriler, JavaScript nesneleri veya bir form veri nesnesi olarak alınır ve JSON.stringify() yöntemi kullanılarak JSON formatına dönüştürülür. Sunucu tarafında, API bu JSON verilerini alır, gerektiğinde doğrular ve işler.
* Benzer şekilde, API'den alınan veriler genellikle JSON formatında döner. Sunucu, istemciye (frontend) JSON formatında veri gönderir ve istemci bu verileri JavaScript nesnelerine dönüştürerek kullanır.
* API, istemciden (frontend) gelen verileri alırken bu verileri JSON formatına dönüştürebilir. JSON formatında veri, API tarafından işlenir, doğrulanır ve veritabanına kaydedilir. Veritabanı, JSON formatındaki verileri uygun bir şekilde saklayabilir ve gerektiğinde bu verileri geri döndürebilir.

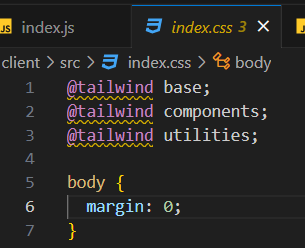
**Proje Kurulumu**

* Proje klasörüne bir Client birde Api adında iki klasör oluşturuyoruz.
* Daha sonra Frontend kısmımız olan Client klasörüne terminalden cd client diyerek bu klasörün içine geçeceğiz. Buraya terminalden react kurulumu yapmak için npx create-react-app . diyerek yeni bir isimli klasör oluşturmadan direkt bu Client klasörünün içerisine react dosyalarını yükleyecek. Tabi bunu yapmak için bilgisayarda nodejs kurulu olmalıdır. Terminale node -v diyerek bilgisayarda kurulu olan node versiyonunu görebiliriz.
* Client klasörünün içerisine geçerek terminalde npm start diyerek react uygulamasını ayağa kaldırabiliriz. Uygulama vanilla olarak 3000 portunda ayağa kalkacaktır.
* React projesi kurulduktan sonra proje içerisinde değişiklik yaparız. Src klasörü içerisinde setupTests, reportWebVitals, logo, App.css, App.test.js dosyalarını silebiliriz. Bunu yapınca proje hata verecektir src klasöründeki ilgili dosyalarda ilgili import kodlarını silerek hatadan kurtuluruz.
* React tamam şimdi Ant Design kuracağız projeye. React projesi ayakta olduğu için önce ctrl+c ile onu durdularım sonra Client klasörü içinde terminalde olmaya dikkat ederek terminale npm install antd yazıp paketi yükleriz. AntDesign için Bootstrapin react içerisinde kullanılan versiyonu olarak düşünebiliriz. Ancak bootstarp kullanmak istersek düz bootsrapt yerine react-boostrrapt kullanmalıyız çünkü örnek olarak butonları düz bir buton olarak değilde içerisinde verilen react için component bazlı butonları kullanıyoruz bu durum ant-designda da aynı. Elemanları component bazlı kullanmak için yani.
* Ant disgn elemanlarımı react projemde kullanmak için kullanacağım component içerisinde elemanı implement edip component olarak çağırmam gerekiyor;

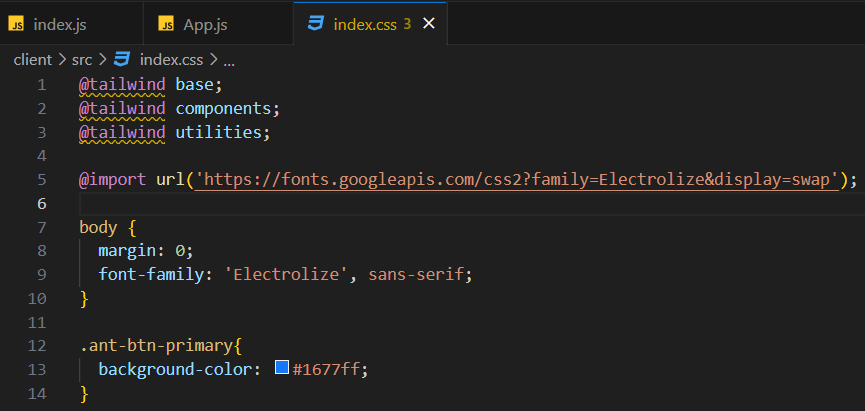
****

* Tailwind Css bootsraptin daha gelişmiş daha özgürce css kullandığımız versiyonu diyebiliriz. kurulumu react projesine yapmak için Client klasörü içerisindeki terminalde npm install -D tailwindcss postcss autoprefixer yazıyoruz. Daha sonra init dosyalarını da yüklemek için npx tailwindcss init -p yazıyoruz bu sayede postcss.config.js dosyası projeye eklenir. Daha sonra tailwind.config.js dosyası içerisinde conten propuna content: [ "./src/\*\*/\*.{js,jsx,ts,tsx}", ] yazıyoruz. Daha sonra index.ss dosyası içerisine @tailwind base; @tailwind components; @tailwind utilities; bunları body css i üstüne yapıştırıyorum. Eğer tailwind css kodları otomatik olarak gelsin istiyorsak vs code a Tailwind Labsin geliştirdiği Tailwind CSS IntelliSense paketini yükleyin. Ve artık text yazıp ctrl+space bastığınızda otomatik doldurmalar gelir.



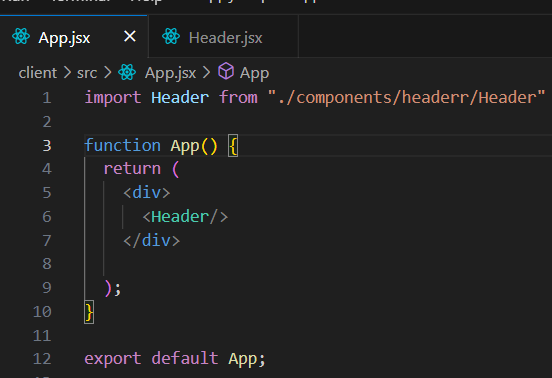


* AntDesign kurduktan sonra TailwindCss kurduğumuz için ufak bir çakışma oldu. Bunu çözmek için index.css dosyasına gireriz ve body altında yeni bir css yazarmış gibi .ant-btn-primary{background-color: #1677ff;} yazarak bu hatadan kurtuluruz. Ayrıca body üstüne tailwind attributeları altına @import url('https://fonts.googleapis.com/css2?family=Electrolize&display=swap'); fontunuda eklemem gerekiyor sonra body cssine ise font-family: “Elctrolize”, sans-serif; ekliyorum.

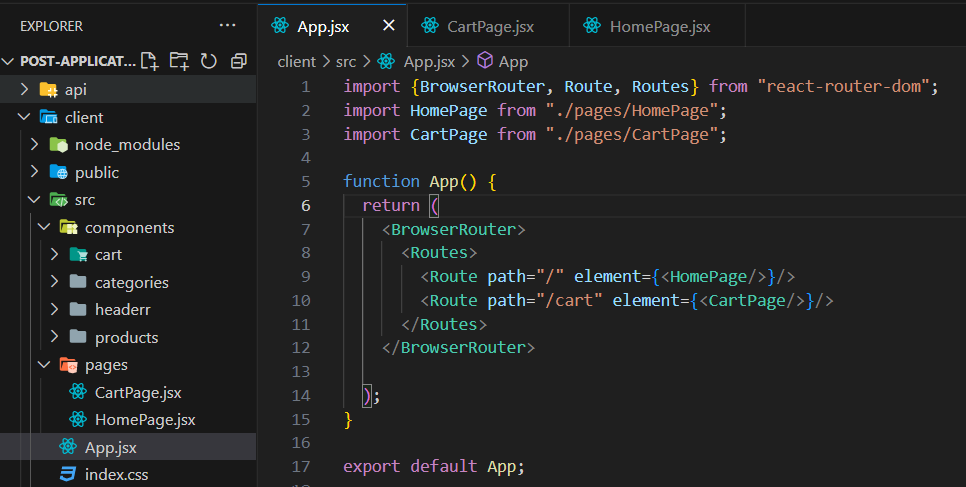


React Forntend Başlangıç

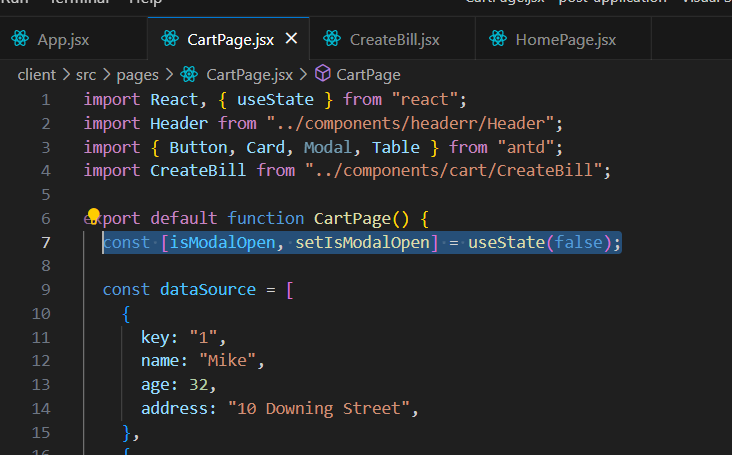
* React component bazlı bir sistem kullanır. Src klasörü içerisine components adında bir klasör oluştururuz componentlerimiz burasının içerisinde tanımlanır. Componentler bir kere tanımlanır ve her sayfada kullanılır defalarca aynı kodları yazmaktan bizi kurtarır. Components kalsörümün içerisinde header diye bir klasör tanımlıyorum ve onun içerisinde Header.jsx dosyasını ekliyorum. Bu benim componentim oluyor. Bu componentleri ana componentim olan App.jsx içerisinde import edip çağırarak kullanabiliyorum. React jsx uzantılı dosyalar kullanır. Jsx dosayalarda hem html kodları hemde javascript kodları yazabiliyoruz. Dinamik olarak html yazmamızı sağlıyor aynı cshtml veya ejs uzantıları gibi.



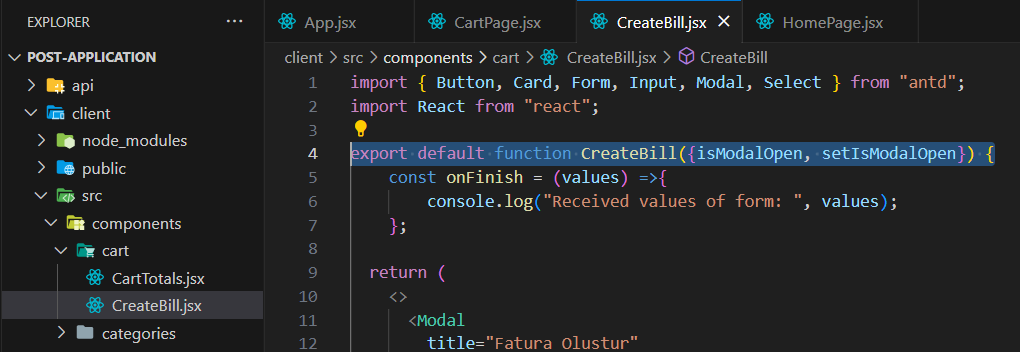
* Tailwind Css için tailwind.config.js dosyasında extend propunda özel şeyler yazabiliriz. Mesela grid sistemini burada kendimize özgü olacak şekilde genişletebiliriz yazarak.
* Sayfadaki yazıların kopyalanmaması için select-none kullanabiliriz. Üzerine gelindiğinde bir gölge oluşsun istersek hover:shadow-lg kullanabiliriz.
* React Router Dom paketi ile sayfalama yapabiliriz. React bir single page application olduğu için bir tane index.html vardı. Yani yazdığım tüm sayfaları bir dosya içerisinde saklıyor bunun içerisinden çağrılıyor sadece. Yani aslında tek bir sayfa var bu sayfa üzerinde sadece componentler değişiyor yoksa sayfa aynı kalıyor. Normal htmldeki gibi farklı farklı sayfalara istek atmıyoruz sunucu tarafını yormuyoruz. Tüm sayfaları componentleri değiştirerek tek bir sayfa üzerinde görmüş oluyoruz. İşte bu olayıda routerlar sayesinde yapıyoruz bu yüzden React Router dom a ihtiyacımız var. React router dom kurmak için Client klasörünün terminalinde npm install react-router-dom yazarız. Daha sonra src klasörü içerisine pages adında bir klasör oluşturacağım tüm ayrı sayfalarımı toplayacağım alan burası olacak. Home sayfamı App.jsx sayfamdan ayırdım ve artık router kullnarak home componentini App.jsx componentine çekip route yapacağım.
* BrowserRouter ana kapsayıcıdır tüm projeyi kapsar. Bunun içerisinde birden fazla sayfam olacağı için bir kapsayıcım olması gerekiyor bu ise Routestir. Birden fazla sayfam olacağı için bunun kapsayıcısı oluyor bu farklı farklı sayfalara yönlendirme yapıyor. Routesin içine ise Routeları yazarız tekli olacak şekilde her bir sayfa bir Routetur bu routea bir path yani url olarak nereye gidecğini söyleyeceğiz ve bu path geldiğinde bu componenti çağır diyeceğiz.



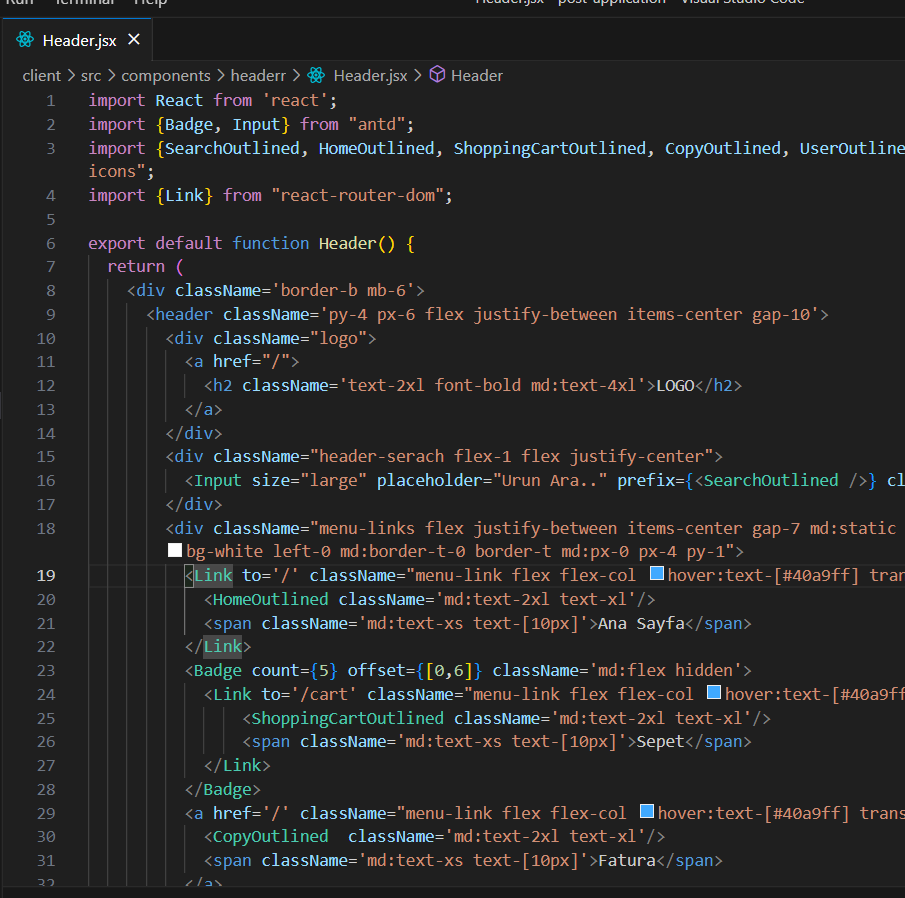
* const [isModalOpen, setIsModalOpen] = useState(false); State tanımlarken öncelikle statein ismini veriyoruz virgülden sonra değişim fonksiyonunu veriyoruz yani bu statete herhangi bir değişim olduğunda bunu setState ile yapacağız ardından eşittir atıp bunun bir useState olduğunu söylerek parantez içnide bu statein ilk değerini atıyoruz.
* Proplar ile bir componentten diğerine fonksiyon, veri veya state taşıması yapabiliriz. Yani bir componentten diğerine veri göndermeyi sağlar.
* Bir state tanımlayalım ve bu statein verisini ve değişme fonksiyonunu diğer componente prop yardımıyla gönderelim.



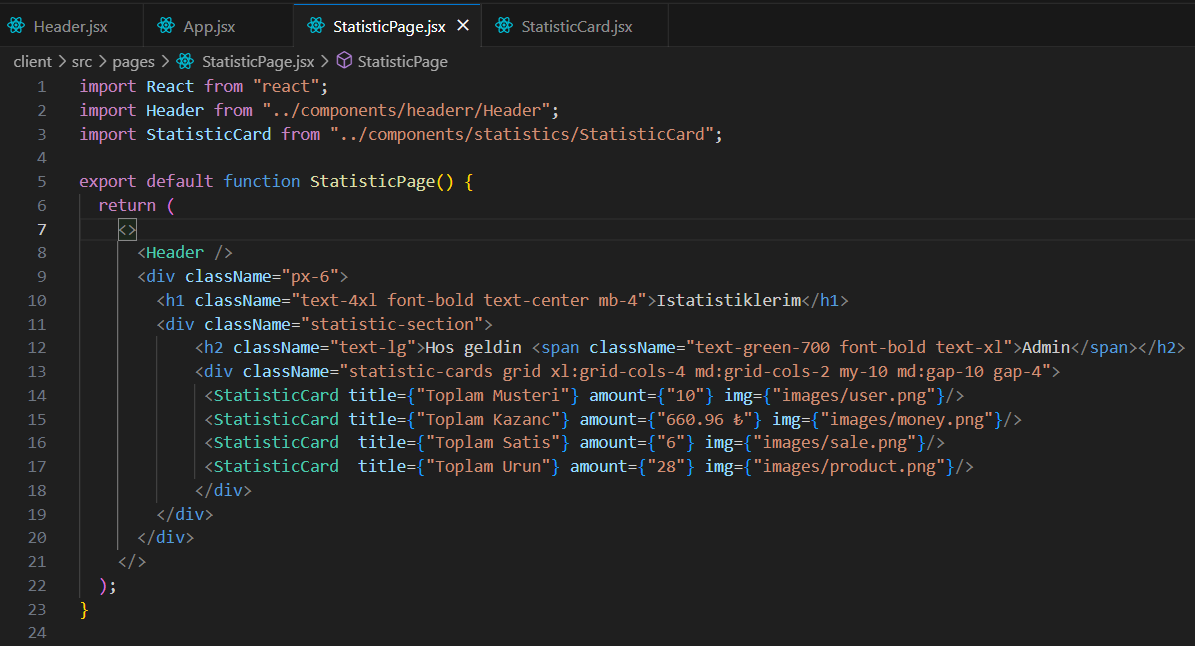


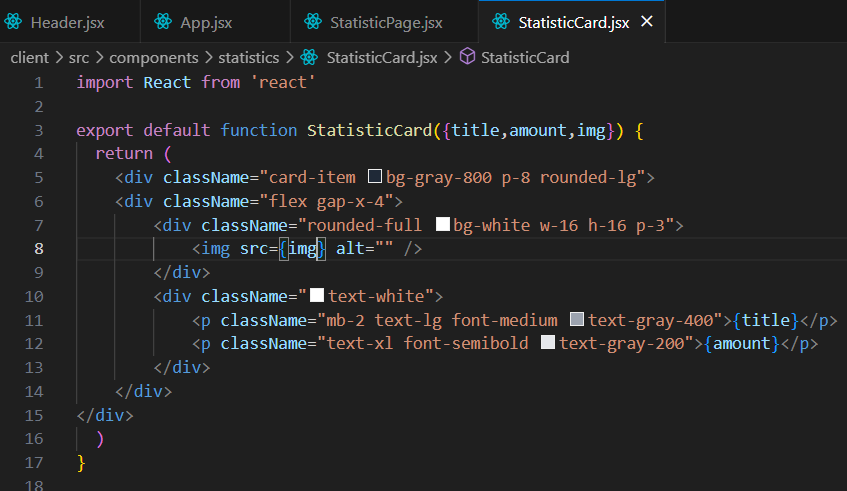


* React Router Domda bulunan Link componentini kullanarak route sayfalarına yönlendirme işlemi yapabiliriz. Ayrıca link etiketi sayesinde parametrede kullanıcının id si gibi bilgileri de göndererek getbyid yani detail sayfaları oluştururuz. Yani App.jsxte oluşturduğumuz routelara link etiketi sayesinde sayfa içinden yönlendirme yapıyoruz. Burda can alıcı nokta herhangi bir sayfa yenilenmesi olmuyor sayfalar arası geçiş yaparken. Eğer burda link yerina a etiketi kullanıp to lar yerine href yazsaydık bu sefer sayfalar arası geçiş yaptığımda sayfa yenilenmesi olurdu. Bu link etiketini asp.net tarafındaki html helperslar olarak düşünebiliriz.



* REACt ile bir componentten props ile dinamik veri çekmeyi gösterelim bu sayede her bir item componentte kendi özel verisiyle oluşur;
* AntDesign ile çok kolay bir şekilde Carousel kullanımı yaptık kodlarda mevcut login ve register sayfasında.

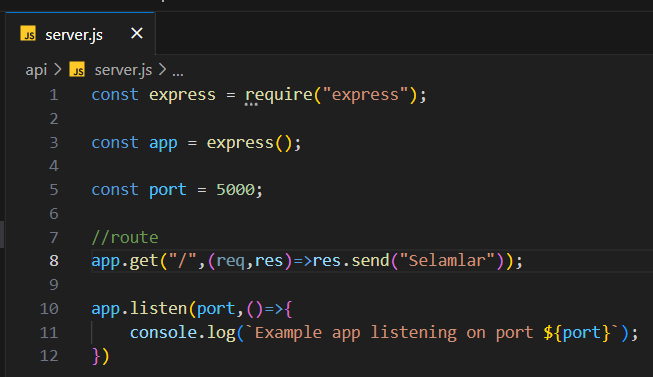




* İstatistiksel grafikler yapmak için AntDesign in çok güzel bir kütüphanesi var. Bu sayede kolayca chartslar yapabiliyoruz. Bunu kullanmazsak react charts kullanabiliriz. Biz AntDesign kullanalım. Bunu kurmak için Client klasörünün terminalinde npm install @ant-design/charts --save yazıyoruz. Sürüm farklılığı hatası çıkabilir kursun videosundan çözümüne bakabilirsin.

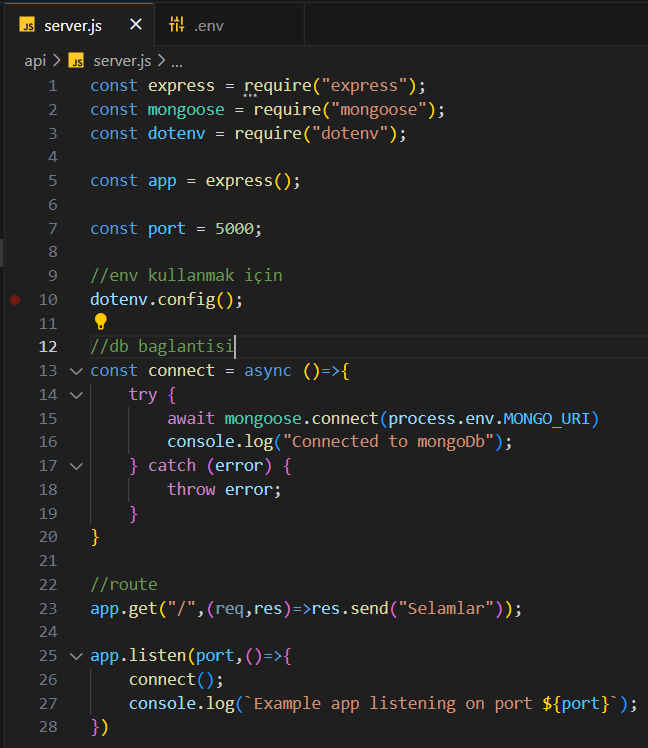
**BACKEND KURULUM**

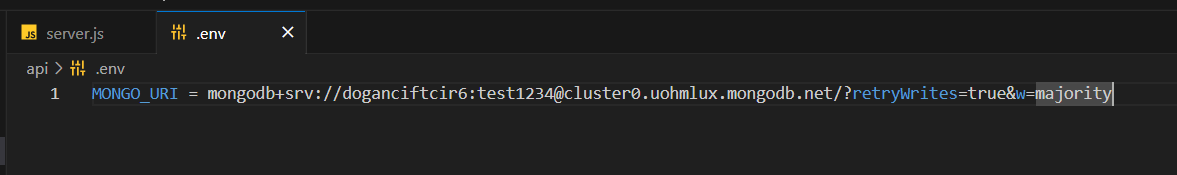
* Api klasörünün içine server bilgilerimizi tutacak olan server.js dosyamı oluştururum.
* Daha sonra cd api diyerek api klasörünün içerisine terminalde giriyoruz.
* Terminal ekranında node service.js diyerek bu js dosyamı sunucu tarafında çalıştırmış oluyorum. Ancak bir değişiklik yaptığımda sunucuyu durdurup tekrar ayağa kaldırmam gerekiyor. Bu durumdan kurtulmak için nodemon paketini kullanabiliriz, ama artık güncel sürümlerle birlikte nodemon paketi yüklemeye bile gerek yok. Terminale node --watch server.js yazdığımda otomatik olarak benim sunucumu anlık olarak dinliyor. Bu özellik node un v18 inde eklendi.
* Api klasörümüzün içindeki terminale önce npm init --y diyerek package.json dosyamızı projeye ekleyelim.
* React projesinin içinde gelen gitignore dosyasını bu api klasörünün içine atalım ki gereksiz olan dosyaların comiti atılmasın. Gereksiz dosyalardaki değişikliği görmesin yani.
* Serverimizi kurmak için bir nodejs frameworku olan Express kullanacağız. Node ve Express arka planda aslında http modüllerini kullanır. Expressi kurmak için api klasörünün terminali içerisinde npm install Express yazıyoruz. Daha sonra bunu server dosyamda require ederek çağırmam lazım çağırarak basit bir server iskeleti oluşturalım. Ayrıca basit bir route oluşturarak bu portlu serverda response olarak veri gönderelim. Res.send() yaparak response döndürmüş oluyoruz.



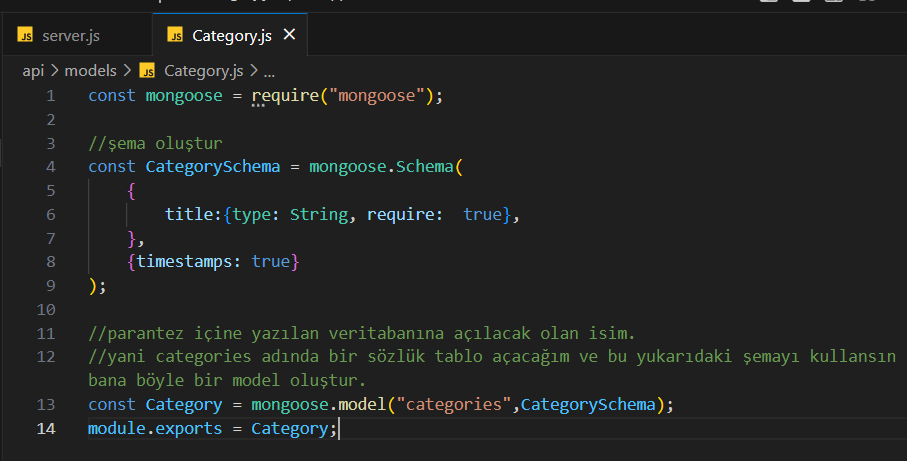
**VERİTABANI KURULUMU VE PROJE VERİTABANI MODELLERİNİ OLUŞTURMA**

* MongoDb Atlas a hesabımızla giriş yapalım.
* Sol üstten Data Servicesin yanından Proje ismine tıklayarak new Project diyoruz. Projeye isim veriyoruz. Direkt create Project diyoruz daha sonra. Projem kurulmuş oldu şimdi proje içine database kurmak için direkt karşımıza gelen yerde Build a Database diyoruz. En sağdaki kart olan free kartı seçiyoruz. Daha sonra direkt Create diyoruz. Bir username ve password belirliyorum bu bilgiler ile db ye bağlanıcaz. Daha sonra en altta IP adres kısmına Add My Current IP Adress diyebiliriz ama böyle dersek modemi yenilersek ıp adresi değişeceği için tekrar girip bunu ayarlamamız lazım yoksa db ye bağlanamayız. Bunun yerine Ip adress inputuna 0.0.0.0 yazıyorum ve add entry diyorum. Daha sonra Finish and Close diyorum ve Go to database diyorum.
* Artık mongodb yi projemde Express ile kullanabilmem için api klasörünün terminalinde npm install mongoose diyerek paketi yüklüyorum. Server dosyamda mongoose paketini require ediyorum.
* Server dosyasında gerekli bağlantıyı kurabilmek için mongodb panelinden cluster içinden Connect butotuna taklayıp Drivers a tıklayıp connect URi sini çekiyorum. Bunu direkt server dosyasına girmektense bir envoriment oluşturarak ordan bu bilgiyi çekerek daha güvenli hale getiriyorum yoksa server dosyasında açık bir şekilde dururdu güvenli olmazdı. Api klasörünün içerisine .env dosyasını oluşturup connection stringi buraya yazıp burdan çekeceğim. Bunu yapabilmek için npm install dotenv paketini kuruyorum. Server dosayama require ediyorum ve kullanıyorum.



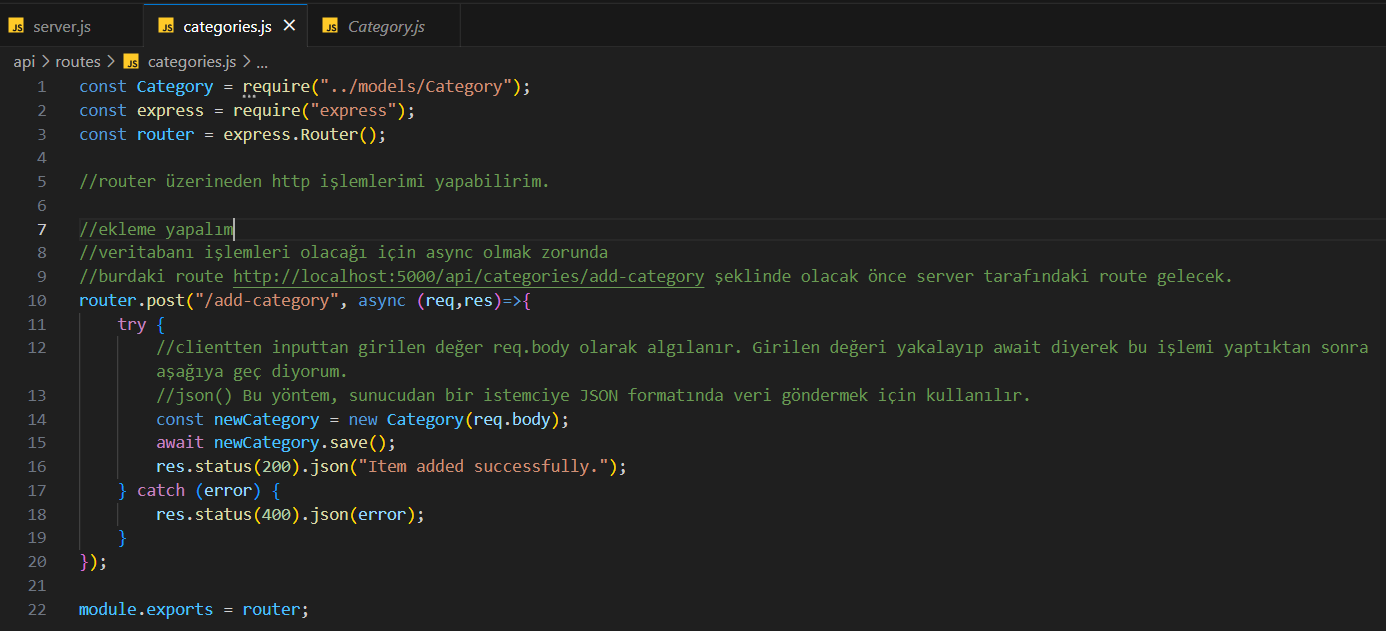


* Projedeki tüm verilerimizi için veritabanında tutulan yerleri oluşturalım yani şemalarımızı ve modellerimizi. Mongoose paketi ile kolay bir şekilde şema oluşturabiliyoruz. Api klasörüne models diye bir klasör oluşturuyorum. Bu klasör içerisine her bir sözlüğe karşılık gelen yani sqllerde bulunan tabloya benzetebiliriz her bir tablonun ayrı ayrı şemalarını, modellerini oluşturmuş oluyorum entitylerim olarak düşünebiliriz. Şema içerisinde timestamps true verirsek kayıt oluştuğu sırada otomatik olarak oluşturulma tarihi eklenir. Örnek bir şema ve bu şemadan model oluşturalım;

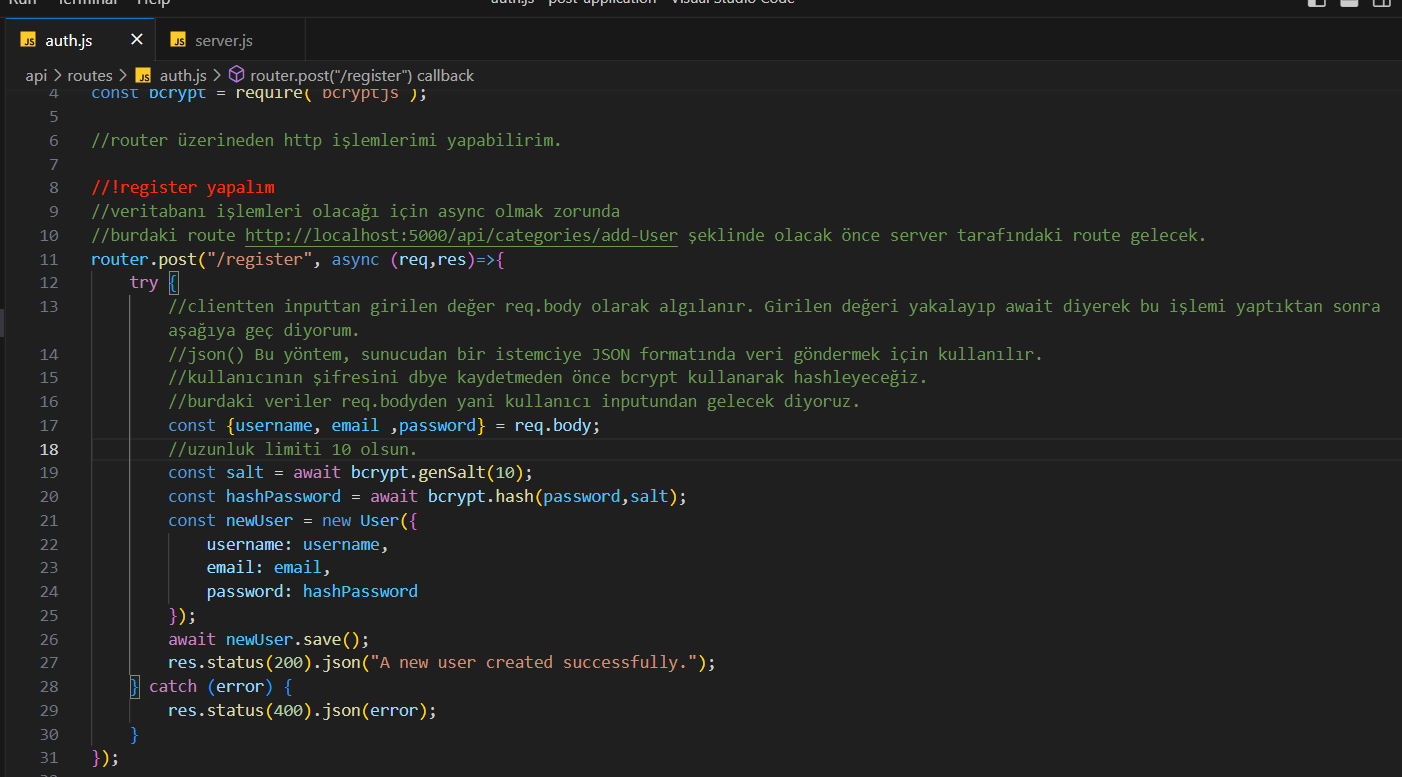


* Routelari direkt App.js içerisinde oluşturursam çok karışıklık olur bu yüzden routerları ayrı bir yerde oluşturucağız.api klasörü içerisine routes adında bir klasör oluşturuyoruz. Express içerisindeki routes özelliğini kullanacağız. Routelarım içerisinde oluşturduğum modelleri kullanarak routelar oluşturucam ve bu routea istek geldiği zaman şunu şunu yap şu responseu döndür diyeceğiz. http isteklerinde hata almamak için npm i cors paketini yüklemeliyiz. Bu paketide server içerisinde middlewear olarak kullanmalıyız. Bu aslında req.body in çalışması için bodyde gelen değeri yakalayabilmek için kullanılmalı. Basit bir post ekleme işlemi yapalım;



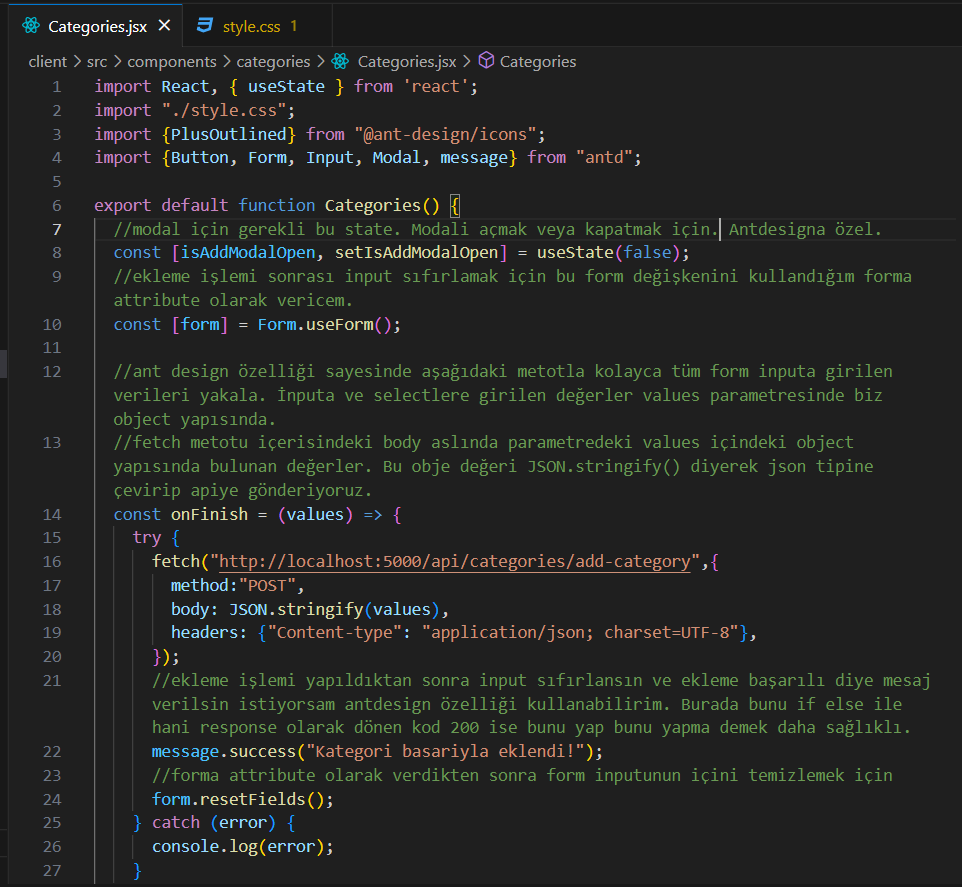


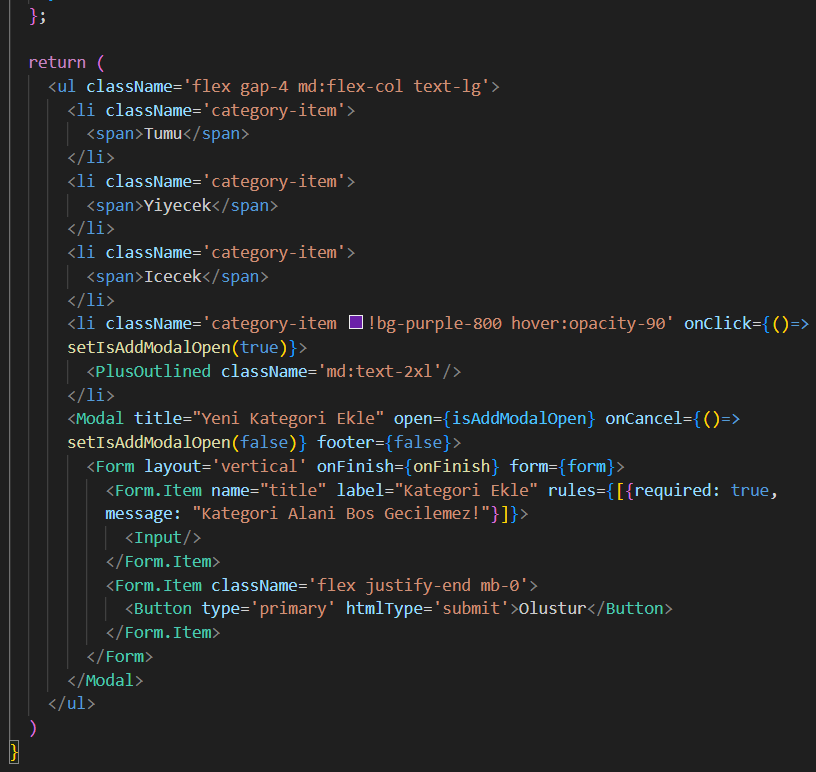
* Register işlemi sırasında kullanıcının şifresini db ye açık bir şekilde kaydetmek tavsiye edilmez. Bcrypt paketini kullanarak şiferyi db ye kayıt etmeden önce hashleme işlemi yapabiliriz. Paketi indirmek için api klasörünün içerisindeki terminalden npm install bcryptjs yazıyoruz.



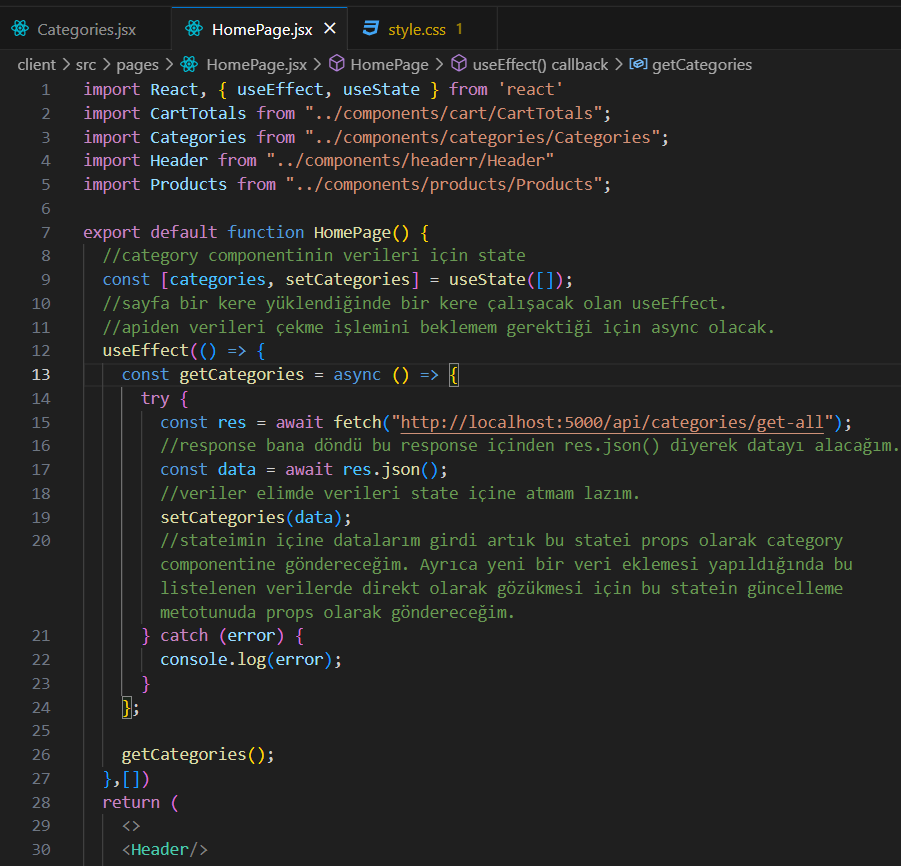
**API CLİENT BİRLEŞTİRMEK**

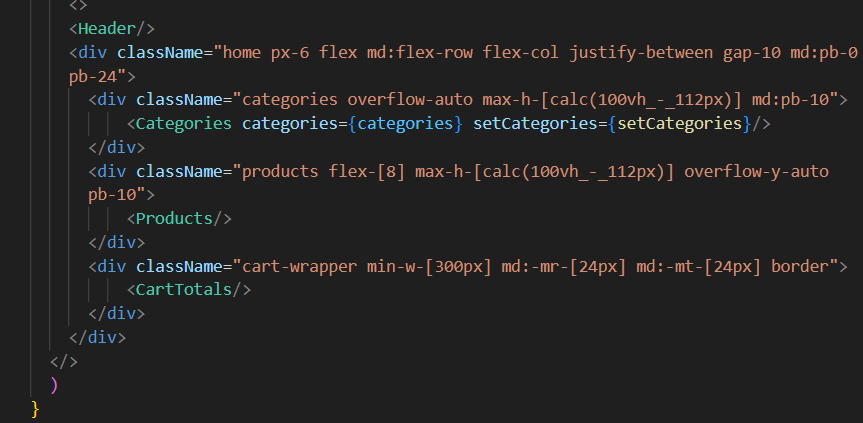
* İnputlardaki name özelliklerini apideki modelimizdeki isimlendirme alanlarına göre yapmalıyız. AntDegisn form etiketine layout=”vertical” diyerek form elemanlarının alt alta gelmesini sağlayabiliriz.
* Custom form kullanmak yerine antdesign formlarını kullanırsak eğer, hem validation oluşturma ve mesajlarını düzenleme gibi kolaylıkları var hem hiç css yazmadan güzel görünümlü elemanlar elde edebiliyoruz hemde normalde uzun uzun kullanıcının inputa girdiği değeri yakalamak için bir state oluşturup inputlara onchange eventi ile tek tek içerisine eleman.value diyerek değerleri yakalayıp statein change metotuyla state değerlerini değiştirmek yerine tek tek uzun uzun vs demek yerine yani burda 20 inputumuz olsa 20 tane state tanımlamam gerekirdi bunun yerine küçük bir metotla tüm inputlardaki selectlerdeki verileri obje olarak kolayca yakalyabiliyoruz. Bu tarz güzel kolaylılları var. Bunu yapmak için onFinish metotunu yazıp bu metotu formun onfinish attributeuna veriyoruz. Değerler bu onFinish metotunun values parametresine düşecektir. Geriye dönen sözlük yapısındaki key değeri inputun name değerine verdiğimiz alandan geliyor valuesi ise input içerisine yazılan değerdir.
* Form inputlarındaki değerleri alıp apiye post istediği atmak için ya fetch yönetimini kullanırız ya da axios paketlerini kullanabiliriz. axios paketi kullanmak daha kolay ama kursta fetch yapıyoruz. İşte burdaki inputlardan fetch ile apiye gönderdiğimiz değerler apide req.body denilerek yakalanıyor önemli nokta burası.
* //ant design özelliği sayesinde aşağıdaki metotla kolayca tüm form inputa girilen verileri yakala. İnputa ve selectlere girilen değerler values parametresinde biz object yapısında.
* //fetch metotu içerisindeki body aslında parametredeki values içindeki object yapısında bulunan değerler. Bu obje değeri JSON.stringify() diyerek json tipine çevirip apiye gönderiyoruz.
* Hadi frontendden inputlarımızdan apiye bir post işlemi atalım; (ancak bu kodlardaki ekleme işleminde eğer categoriler get isteğiyle çekilirse sayfaya yeni eklenen veri sayfayı yenilemeden ekranda gözükmeyecektir bunu aşağıdaki diğer kodlarda kurtardı k düzelttik yani)

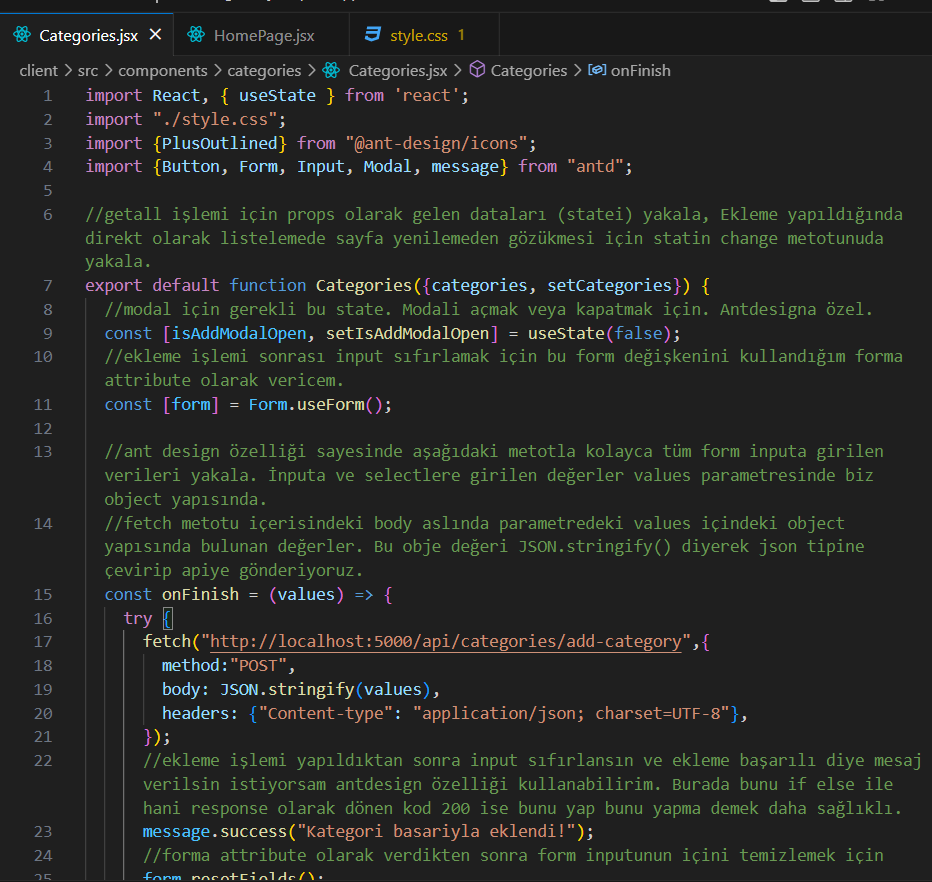
****

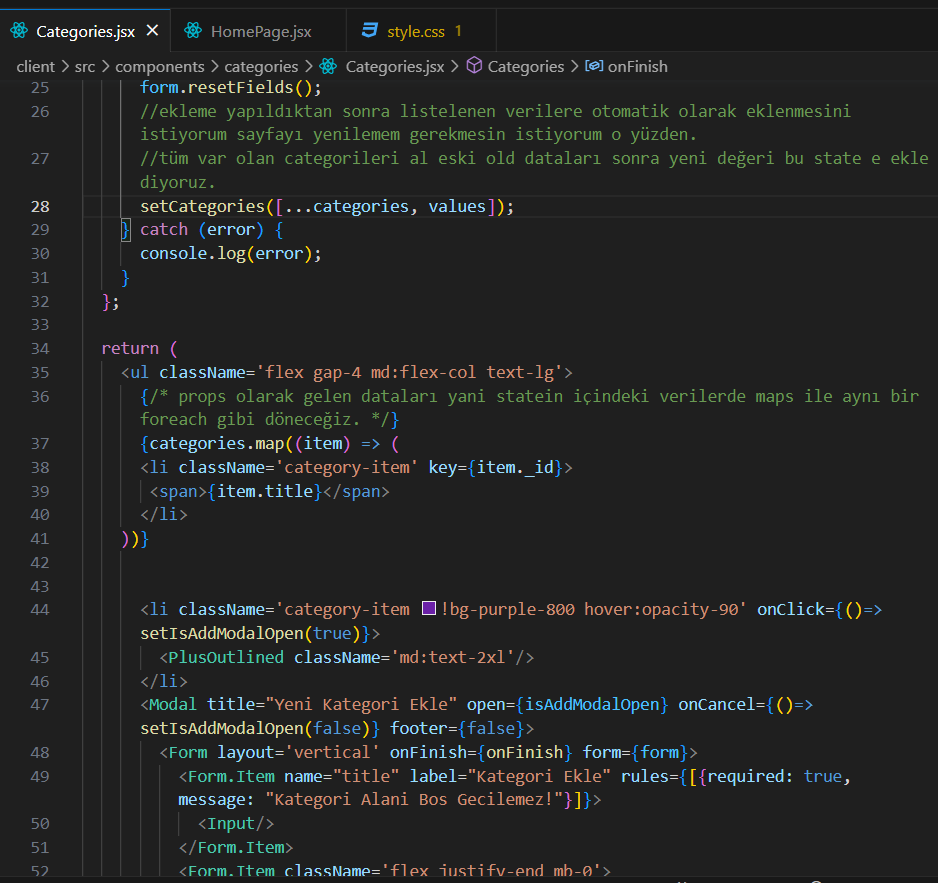
****

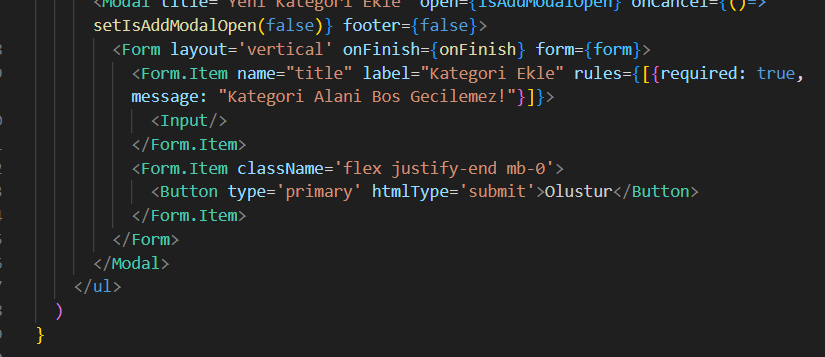
* Şimdi bir get işlemi yapacağız. Category sayfasında kategorileri apiden çekeceğiz. Bunun için Category componentinde state oluşturabilirim ama ben bunu tüm sayfada kullanmak istiyorum. Yani anasayfama girdiğinde buradaki category kısmını eğer apiden çektiyse bu sayfayı yükle demek istiyorum. Bunun için bu statei categoryde değil home componentinde yapmam gerekiyor. Yani sonuçta bu category componenti home componentinde gözükecek bu yüzden böyle yapıyoruz. Category componentine veriler gelmediyse home componentini çalıştırmanın bir anlamı yok. Bu yüzden home componentine statei tanımlayıp bu statei category componentine props olarak gönderip öyle category sayfasında bu verileri kullancağız mantık böyle işliyor. Eğer bu statei category componentine oluştursaydım bunu home componentine props olarak göndermem sıkıntılıydı alt componentten üst componente props olarak veri göndermek sıkıntılı bir iş zaten bu yüzden context API kullanıyoruz react içerisinde. React tek taraflı binding özelliğine sahip bunu unutmayalım.
* Get işlemi yapmak için bir state oluşturuyoruz. Verileri apiden yine fetch metotu ile çekeceğiz. Bunun için useEffect hooksunu kullanacağız. Nedeni ise istediğimiz verileri şu zamanda render et şu zamanda çalıştır demek için. Bu useEffectin sayfa yüklendiğinde bir kere çalışmasını isteyeceğiz.



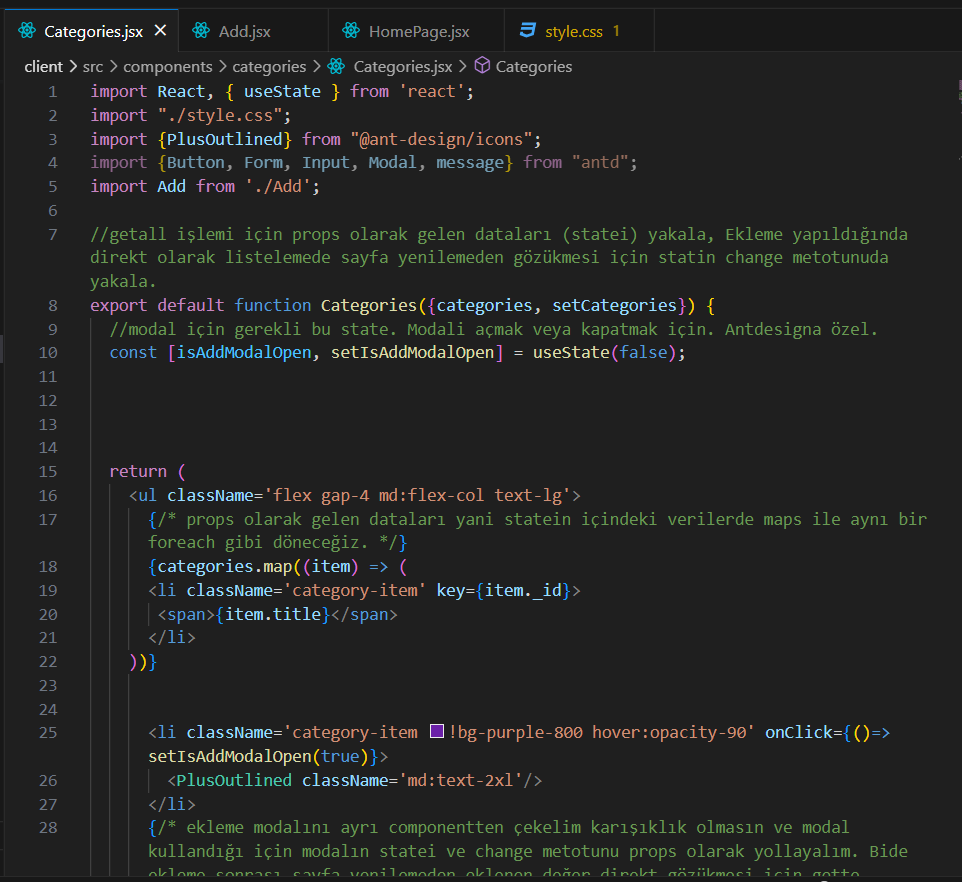


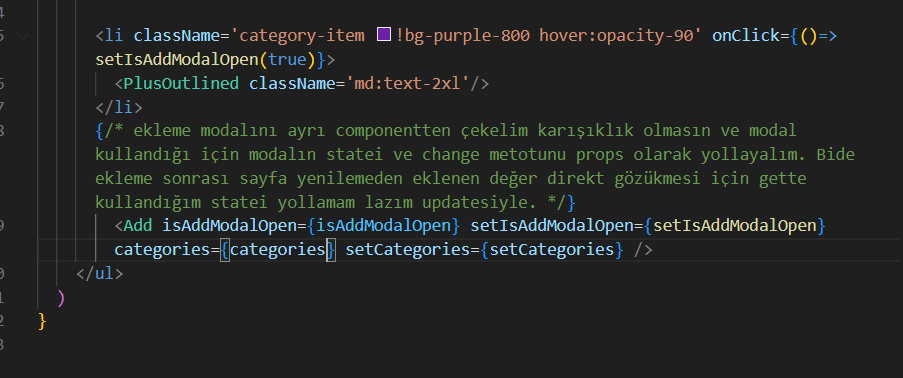


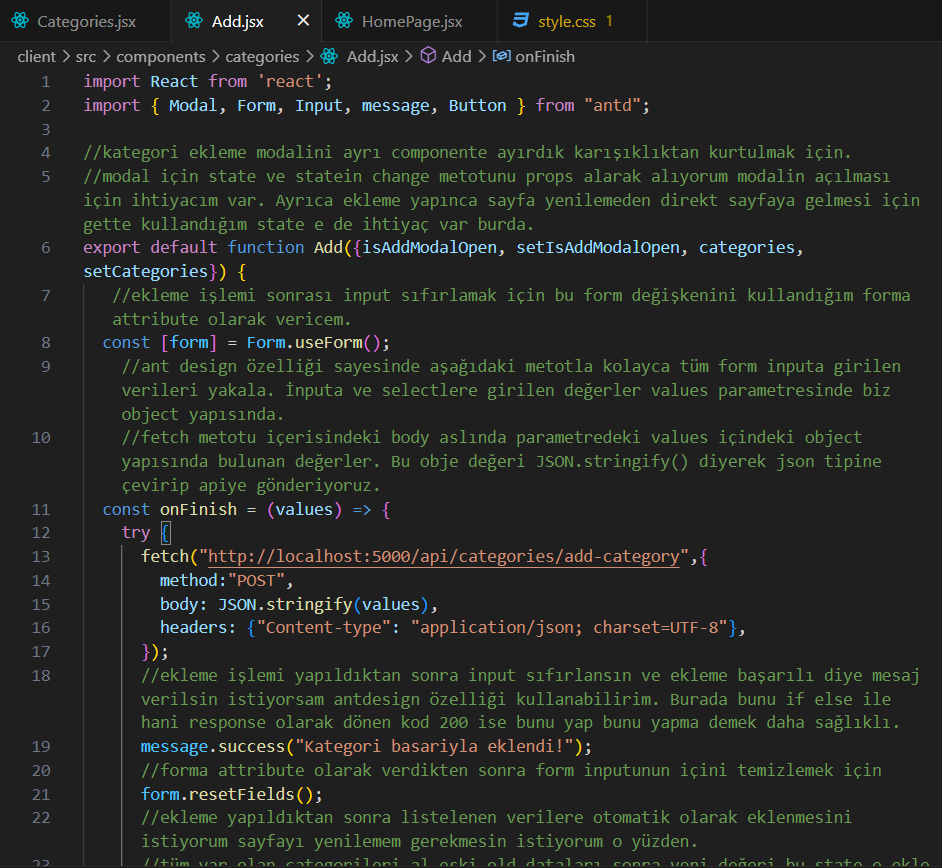


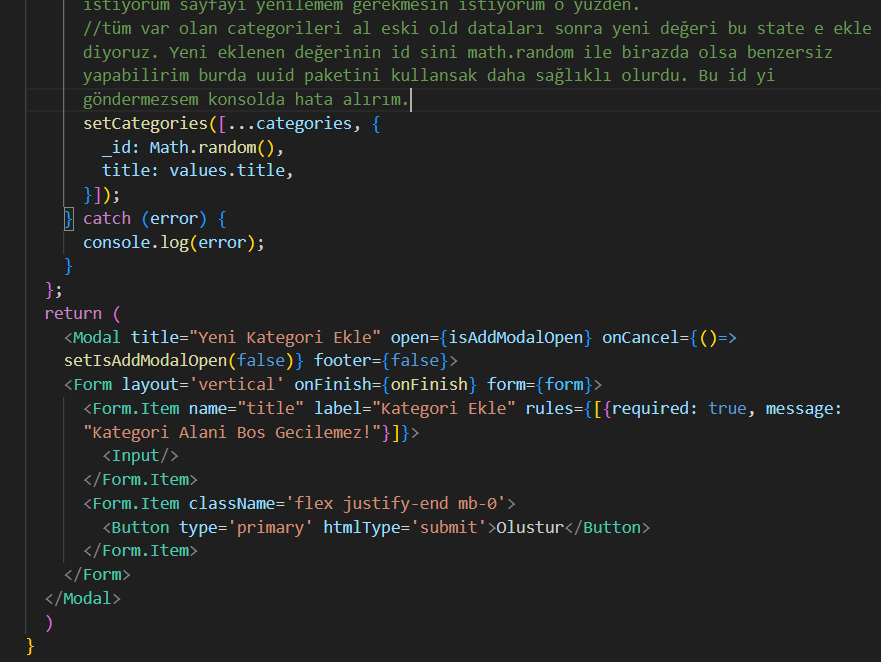


* Olaya düzenleme ve silme işlemleri gireceği için, düzenlmeyide bir modalda yapacağımız için bu kodlar çok karışacak o yüzden ekleme ve düzenleme modallarını ayrı componentlere ayırıp daha temiz bir görünüm elde edeceğiz bu tarafta. Bu düzenleme sonrası getall işlemi olan ve bir modal üzerinden ekleme yapan kodlarım;





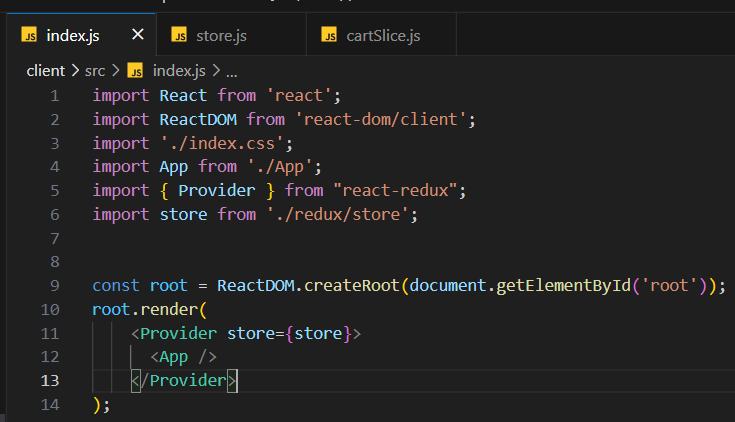


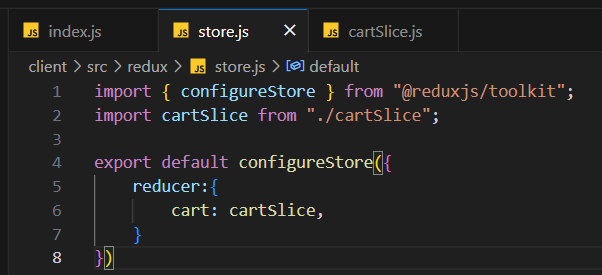


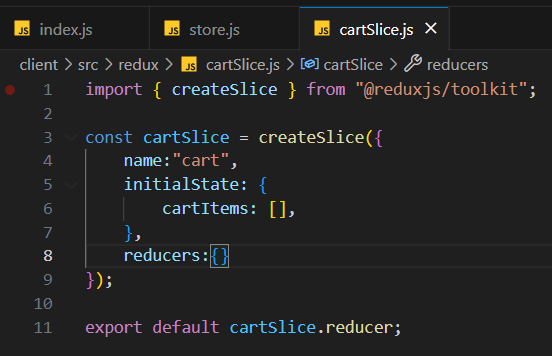
* Edit sayfası için yine add işleminde olan tamamen aynu işlemler var. Farklılıklar var projede mevcut yapımı.

REDUX TOOLKİT

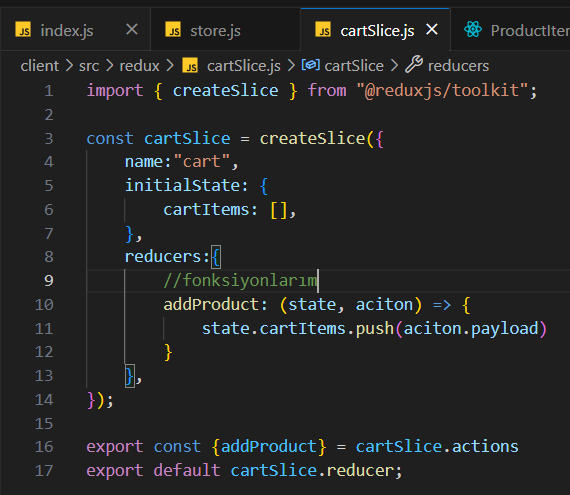
* Bir component içinde oluşturduğumuz örneğin bir state i başka bir componente atmak için propslar ile çalışıyorduk. Bu ise işi hem karmaşık gösteriyor hem fazla uğraştırıyor. Yani günün sonunda ortaya bir karmaşa çıkıyor. Bunun yerine daha kolay kullanım yapabiliyoruz. İşi basitleştiren ve sadeleştiren state yönetim araçları var. 6 araç var ama en çok Redux ve Context API kullanılıyor. Sürekli iç içe props kullanımı aktarımı yapmaktan kaçınıp state yönetim araçlarını kullanarak olaya bir sadelik, ferahlık katmalıyız. Redux Toolkit, reduxın daha basit yazılmış paketi daha da sadeleştirilmiş hali diyebiliriz. Önceden redux öğrenmek çok zordu ama artık redux toolkit ile bu daha da basitleştirildi. Redux toolkit kurmak için client tarafının terminalinde npm install @reduxjs/toolkit yazıyoruz. Redux toolkiti sadece reactta değil tüm js uygulamalarında kullanabiliyoruz. Ama react için kullanacaksak birde npm install react-redux paketini de kurmam gerekiyor. Eğer projede ant design charts paketi varsa bu kısımda hata alacağız. Önce charts paketini remove et ant design versiyonunu düşür, react paketlerini indir, sonra tekrar charts indir ve ant design paketini yükselt.
* Toolkit paketlerini yükledikten sonra index.js içerisine gidiyoruz. İmport {Provider} from “react-redux” yazıyoruz. Daha sonra StrictMode içinde tüm projeyi sarmalamış oluyoruz. Daha sonra client src içine redux isminde bir klasör ve bu klasör içerisine store.js adında bir dosya. Bu dosya içerisinde import {configureStore} from “@reduxjs/toolkit” çağırıyorum ve burda reducerlerimi export etmiş olacağım. Reducerlerimi oluşturup burada tanımlayacam. Reducerleride redux klasörü içerisinde tanımalayacağım. Reduclarimi store içerisinde çağırıcam storeu ise index.js içeirsinde çağırıcam.
* İndexjste tüm projeyi kapsayacak olan bir Provider yazdık bu Provider react-reduxtan geliyor çünkü reduxla reacti birleştiren paket. Bu Provider benden bir store istiyor yani benim tüm statelerim nerde nerden geliyor benim bir tane mağazam var burda redux storeum var. Bu store içerisinde ben cart açtıysam cartla ilgili cartSlice içerisinde yarın öbür gün gelir başka bir şey açmak istersem yine bu store içerisine yazabilirim. Yani açmak istediğim stateleri burada tanımlıyorum register ediyorum. Mesela auth ile ilgili bir şey yapıcam ozaman auth : authSlice isminde yazabilirim burda topluyorum hepsini yani. Buradaki tanımladığım stateleri tüm proje içerisinde yani tüm componentlerde çekip çekip kolayca kullanabiliyorum.

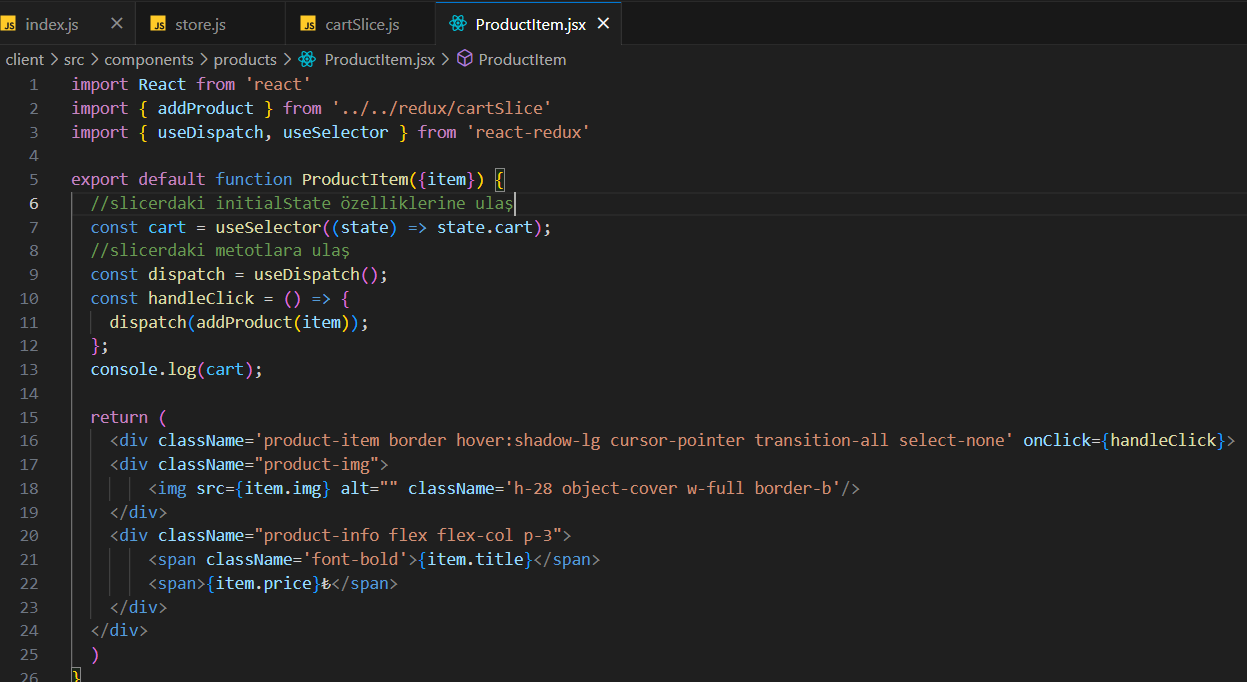






* Google chromea Google react developer tools extensionu indirirsek sayfanın consol tarafından tüm componentlerimizi oradan görebiliyoruz.
* Bu reducerlarım veya Slicelerlarım içindeki reducers kısmı metotlarımızı ifade ediyor. Cart ile ilgili olan metotlarımı burada tanımlayabilirim. Metotlar parametre olarak bir state alıyor birde aciton alıyor. Action dediğimiz parametre bu metotu kullandığımızda çağırdığımız veriler yani bu metotla birlikte gönderdiğimiz veriler, metotu kullandığımız yerde metot parametresine verdiğimiz veriler anlamına geliyor yani. State parametresi ise slice içindeki yukarıdaki initialState içerisindeki tanımladığımız her şey anlamına geliyor. State.cartItems diyerek bu initialState içerisindeki cartItems özelliğine ulaşmış oluyoruz aslında. Buradaki tanımladığımız metotlara başka componentlerden ulaşabilmek için export yapmamız gerekiyor. Reducers içerisinde yazdığımız metotlar actions olarak geçiyor. Daha sonra başka bir componentten bu metota ulaşmak için import {addProduct} from “../../redux/cartSlice gibi bir kullanım yapıyorum. Birde react redux ı import etmem lazım bu slice ı react ile kullanabilmek için aracı bir şeye ihtiyacım oluyor. İmport { useDispatch } from “react-redux” diyerek bir dispatch oluşturup bu dispatch ile slicer içerisindeki metotuma ulaşacağım direkt olarak metota ulaşım sağlayamıyorum bir aracıya ihtiyacım var işte burdaki aracı dispatch oluyor.
* Slicer içindeki fonksiyonları başka componentlerden çağırabildiğimiz gibi initialState özelliği içerisindeki tüm statelerimi, özelliklerimi vs de çağırabiliyorum. Bunu ise yine redux paketinden implement olacak şekilde useSelector hookunu kullanıyoruz.





* Bu sayede artık verilerimi propslar ile ordan oraya geçirmeme gerek kalmadı direkt nerde kullanacaksam bu şekilde çağırıp kolayca kullanabiliyorum statelerimi ve fonksiyonlarımı vs.
* Butona bastığımıza yazıcıdan yazdırma işlemi için kütüphane lazım. Npm install react-to-print paketini kurar kullanırız.
* userRef hooku ile dom içerisindeki bir elementin içerisindeki current target valuelerine ulaşabiliyoruz. Videoda kullanım detayı var basit bir işlem.
* Morgan paketi attığımız http isteklerinde bize log tutuyor. Bu paketi api klasörünnü terminaline npm install morgan diyerek kurabiliriz.Daha sonra server.js içerisinde require ettikten sonra. Middlewear olarak kullanabiliyoruz. app.use(logger("dev")); diyerek development için bana log tut diyebiliyoruz. Bu şekilde apinin terminalinden isteklerin logunu görebiliriz.
* Google konsolunda network simgesinden No thorttling yazan yerden 3g slow vs seçerek tarayıcımızın internet bağlantısını yavaşlatabiliyoruz bu sayede interneti yavaş olan kullanıcıların deneyimini test edebiliriz. Bir state ile from submit sırasındaki butonlara bu slowluk yüklenme anında spinner döndürebiliriz projede kullanım var.
* Ant design tablelarında    //text direkt içeriğin değerini veriyor, ikinci parmetre record ise elemanı etiketiyle birlikte vs.
* Localstroge ta giriş yapan kullanıcının bilgilerini tutabiliriz. Bunun yerine jwt kullanmak her türlü daha mantıklı ama kullanmadık. Localstroge veritabanı gibidir ama kullanıcının tarayıcısı üzerinde veri depolar. Bu uyguluma client side çalışacağı için bu yapılan işlemler geçerli ama daha profesyonel bir uygulama için jwt kullanmalıyız. Kullanıcı bilgisini localStorage da tuttuktan sonra bu bilgiyi kod tarafında çekerek kullanıcı bilgilerini kullanabiliriz.
* Ant designin tablo kolonlarında yazı yazarak ifadeleri aratmak için react-highlight-words paketine ihtiyaç var kullanım örneği ise ant design table compoennt dökümanlarında var. Ayrıca kolonlarda basitçe sıralama da yapabiliyoruz kullanım örneği yine var.
* Spin componenti ile antdesign sayesinde sayfa yükleme simgesi çıkartabiliyoruz kullanım detayı dökümanda ve projede var.
* Fetch işlemlerinde api urlsini belirtirken <http://localhost:5000> i bir .env dosyasına alıp oradan çekersek daha güvenli bir uygulama yapmış oluruz. Bu env dosyasını githuba pushlamayız bu dosyanın gizli olması gerekiyor kaynak kod olarak herhangi bir yerde paylaşmıyoruz.
* Render.com sayfasından uygulamamızı canlıya alabiliriz. Yapılışı son videoda var.