DUSUNCE TAHTASI UYGULAMASI

BACKEND AND DATABASE

CREATE PROJECT AND SETUP BACKEND

Projeyi olusturdum backend frontend dosyalarini olusturdum
Backend'in icine girip node.js'i kurdum. Bu projede backend'de Node.js ve MongoDb'yi
kullanacagim.

- backend dosyasinin icine girip server.js dosyamizi olusturduk. ilk olarak express'i import ettim daha sonra expressi bir variable'a atadim.
  Sonra 5001 portunu dinlemek icin app.listen yaptim bu sayede backendimiz 5001 portunu dinleyecek.

Daha sonra yapacagimiz notes uygulamasi icin get, post, put, delete metodlarini olustrudum
simdilik sadece message olusturuyor. daha sonra bunlarin iclerine herbirinin gorevine gore iclerini doldurucam.

Bunlari yaptikdan sonra refactoring yaparak separation of concerns prensibini uyguladim, kod ortamimi temizledim sadelestirdim. bu request metodlarini dosyalara bolucem cunku bunlar o sayfanin
yonetilmesini fazlasiyla zorlastiricak.

- Routes klasoru olusturdum bunun altina notesRoutes olusturdum ve bu routerlari ilk olarak bunu icine yerlestirdim..
  -Is mantigini controller klasorunu olusturup notesController dostasinda ayri metodlar halinde duzenledim.
  -Routerlar'da artik sadece controller metodlarini import edip parametre olarak kullaniyorum.

SETTING UP OUR DATABASE

Bu projede MongoDB No SQL Database kullanicam cunku yaptigim uygulama cok kompleksli bir uygulama degil ve hizli olmasini istedigim icin bunu tercih ediyorum

MongoDB Atlas'da bir cluster olusturuyorum. Daha sonra baglanmam icin gereken URI'i aliyorum ve bunu .env dosyamin icerisine koyuyorum bu sayede bilgilerimizi saklamis oluyoruz

src/config klasoru olusturp icine db.js dosyasini olusturuyorum bu dosyanin icerisinde mongoDB'ye baglanana fonksiyon olucak
ve bunu alip server.js'de import edip mongoDB'ye baglanmasini saglayacagim

Daha sonrasinda her bir notun ozelliklerini veri tabaninda tutucak Notes modelini olusturuyorum. Bunun icinde src/models altinda Notes.js dosyamda mongoose semasi olusturuyorum ve icine title ve content objelerini olusturuyorum birde ayri bir obje olarak timestamp olusturuyorum bu createdAt updatedAt bilgilerini tutucak database.

COMPLETING OUR CONTROLLERS

Controlleri olusturken async olarak olusturuyorum cunku promise olarak donmesini bekliyoruz ve bunu koyarak bize bu sozu her turlu yerine getirmesini bekliyoruz.

Bu adimda backendimizi database ile konusturuyor olucam. get, post, delete, update routerlarini tamamlayacagim

Ilk olarak getAllNotes controllerini ayarlayip butun notlari getirmesini sagliyacagim. Bunun icin Note.find() kullanarakk ardindan bunu bir degiskende sakliyorum daha sonra json formatinda geri donduruyorum. Ve bunu postman'de GET kullanarak yaptigimda butun notlari benim karsima getirmis oldu.

Sonrasinda createNote controllerini tamamliyorum bunun icin body'den title ve contenti almam gerekiyor bunun icin ilk olarak server.js dosyama gidip app.use(express.json())
kod blogunu ekleyerek title ve content'e erisebilir hale getiriyorum. Title ve contenti req.body'den alip note degiskeninde tutuyorum daha sonra bunu kaydetmesini awaitle bekliyorum.
Daha sonra sonucu statusle birlikte json formatina cevirerek kaydediyorum.

Update controller'ini tamamlamak icin yine title ve contenti req.body'den aliyorum variable'da sakliyorum. Daha sonra awaitle birlikte(her seferinde yaptigim icin artik cok teknik kisimlara girmiyecegim) findByIdAndUpdate
metodunu kullanarak id'ye gore update yapmasini sagliyorum. Birde bir kontrolcu ekliyorum eger id yoksa 404 donduruyorum.

Delete'de de ayni seyleri yapiyorum hemen hemen sadece findByIdAndDelete kullaniyorum farkli metod olarak daha sonra veri tabanini guncelliyorum.

MIDDLEWARE (ARA YAZILIM VE HIZ SINIRLAMASI)

Middleware; ara yazilim olarakda gecen attigimiz requestin responsu gelirken araya koydugumuz yazilimdir
Bunu kotu niyetli kisiler kotuye kullanabilir direkt olarak databaseimize erisebilirler o yuzden bu araya gecisler
guvenlik onlemleri koyabiliriz. Ben bu projede authentacation kullanmadigim icin sadece ratelimit ekledim.
Ratelimit'de bir kisinin belirledigimiz sure icersinde kac defa istek atacagimizdir o sinir asilirsa 429
Too many request hatasini verdirmis oluyoruz

Ben ratelimit verilerini hizli sekilde alabilmek ve depolayabilmek icin upstash kullanacagim.

Projeme upstash'i kurduktan sonra config klasoru icinde upstash ayarlarimi yapiyorum Ratelimeter'i 60 saniyede 100 istek alicak sekidlde
ayarliyorum. Daha sonra Middleware klasorumun icinde rateLimiter dosyamin icinde upstash dosyamin icindeki metodu cagirarak kullaniyorum
Eger bu limit asilirsa res.status(429).json({message: Too many request please try

again}) donduruyorum eger bir sikinti yok ise yani limit asilmiyoe ise
next() diyerek uygulaminin beklnene gibi calismasini soyluyorum

Daha sonra server.js'de ikinci middlewareim olarak app.use(rateLimiter) ekliyorum ve bunuda projeme eklemis oluyorum. 1. middleware ise title
ve content'i Json formatindan objeye js formatina donusturen express.json metoduydu.

FRONTEND

Bu adimda frontend'e geciyorum. Ilk olarak projeme react'i vite ile, react router, tailwind kurup Sayfalarimi olusturuyorum. Pages klasoru altinda sayfalarimi olusturdukdan sonra App.jsx'de bu sayfalarimi. Routes altinda
Route ile birlikte path'leriyle birlikte yerlestiriyorum. Bu sayede her sayfanin farkli bir pathname'i olmus oluyor.

Diger adimlarda ise sirayla sayfalarimi tamamlayip backend'e baglayacagim.

HOMEPAGE

Homepage sayfasinda ilk olarak Navbar componenetsini olusturucam bunun icin Navbar dosyamin icinde gerekli JSX yapisini TailwindCSS kullanalarak olusturutorum
Ve bir adet Link olusturup to'suna /create sayfasini veriyorum.
Daha sonra bu componenti Homepage'de import ederek icine koyuyorum.

Daha sonra RateLimited icin yani cok fazla istek geldiginde ekrana gelicek kismi yapiyorum bununda bir component olarak yapiyorum. Bununda Homepage'de JSX kisminda
kosullu olarak eger rateLimited true olursa gelecek sekilde ayarliyorum.

Isleme gecmeden once cors ayarlarini yapiyorum. Eger bunu backend'de yapmazsam diger farkli adreslerden gelen istekleride kabul eder ve bu hic guvenli degil bu yuzden server dosysamda cors
kullanip sadece kendi frontend url'mi yaziyorum bu sayede sadece benim isteklerimi kabul edicek

UseEffect kullanarak verileri cekme islemini ve rateLimited islemini ayarliyorum.
-Ilk olarak icinde bir async fonksiyon olusturuyorum axios un get metoduyla birlikte 5001/api/notes'den notlari cekip bir variable'da tutuyorum.
-Not stateini gelen response'u ile guncelliyorum ve rateLimited'i false yapiyorum cunku veriler geldiyse zaten false olmasi gerekir.
-Cath blogunda ise eger response.status = 429 yani too many request ise rateLimit degerini true'ya cekiyorum boylece UI ekrana gelmis oluyor kosullu blogumuz sayesinde.
-Finally olarak setLoading stateini false olarak ayarliyorum;
-En sonda ise fetchNotes() fonksiyonunu useEffect'in sonunda cagiriyorum boylece sayfa her yenilendiginde verilerin gelmesini saglamis oluyorum.

Daha sonra HomePage'de notes stateini mapleyerek her bi not icin NoteCard bileseni olusturuyorum.
NoteCard componentini olusturup icinde JSX'ile title, content ve createdAt'i yerlestiriyorum createdAt icin istedigim formata ayarlayabilmek icin bir fonksiyon kullaniyorum bu sayede
tarihi 13 May 2024 seklinde gosterebilmemi sagliyorum.
HomePage'de her bir not icin ozel key'i olacak sekilde mapleyerek butun notlari ekrana bastirmis oluyorum bu sayfayida bu sekilde tamamliyorum.

CREATEPAGE

Bu sayfada not olusturma kismini yapicam. HomePage'de yeni not butonuna basilinca kullanicinin karsisina cikacak olan sayfayi ayarliyacagim. Bu sayfada title content
alarak yeni bir not olusturucam. Bununda backend'deki createNote ile bagdastiricam. Bu sefer get istegi degil post istegi atarak yeni not olusturmus olucam.

Bu sayfada ilk olarak inputlardan aldigim bilgileri tutacagin stateleri olusturuyorum.
Daha sonra iki tane input bir tane button olusturup inputlarin degerlerini statelere bagliyorum. Formun onsubmitine olusturacagim handleSubmit fonksiyonunu veriyorum.
Daha sonra handleSubmit metodumu olusturuyorum bu metod'da axios ile post istegi atacagiz.

-Ilk olarak inputlar bos ise hata verdirtiyorum
- trycatch blogu icinde try'in icinde post request atiyorum ve parametre olarak title ve content statelerimi veriyorum. Daha sonra basarili bildirimi verip HomePage'e yonlendiriyorum
-Catch blogunda eger api'ma cok fazla istek gelirse hata bastiriyorum bu sayede kotu niyetli kisilerden korunmus oluyorum.

> Daha sonra 2 sayfadada axios isteklerim oldugu icin bunlari kisaltmak ve clea n kod icin bir dosyada topluyorum BASE URLL vererek bu metodu baska sayfa lardan import ederek daha kisa yonetmle kullanabiliyorum sdece parametre ol arak /notes gibi sonra kisimi verilmesini sagliyorum

NOTE DETAIL PAGE

Burada ana sayfada tiklanan notu detay sayfasina goturucem. Orada degisiklikler yapabilir.

Daha once route'u ayarladigim icin zaten NoteDetailPage'in path'i /:id olarak ayarliydi. Yani anasayfadan herhangi
bir not'a tiklandiginda direkt olarak o notun url'sine notun idsien gore gidiyordu bu sayede her not icin farkli sayfa olusturmus oldum.

Direkt olarak NoteDetailPage'e gidip useParams hooku ile url'deki idyi alip o sayfa icerisinde kullanabiliyorum cok kullanisli bir hook.
Statelerimi ayarladim.

Daha sonra notu cekebilmek icin useEffect'i id degisince yenilenecek sekilde ayarladim. Bu sayede sadece id degistiginde not verisini cekicek uygulama.
Sonra JSX yapimi ayarladim Inputlarimi ve butonumu koyup gerekli statelerle bagladim.
Handle fonksiyonlarimi yazdim. handleSave'de axios.put delete'de yine delete kullanarak ilgili not'un idsine gore notu guncelleme veya silme islemini gerceklesitrmesii
butonlara vererek sagladim.

Projenin son asamasinada boylece gelmis oldum.
Okuyanin gozune benimde elime saglik...

EFE OZEL