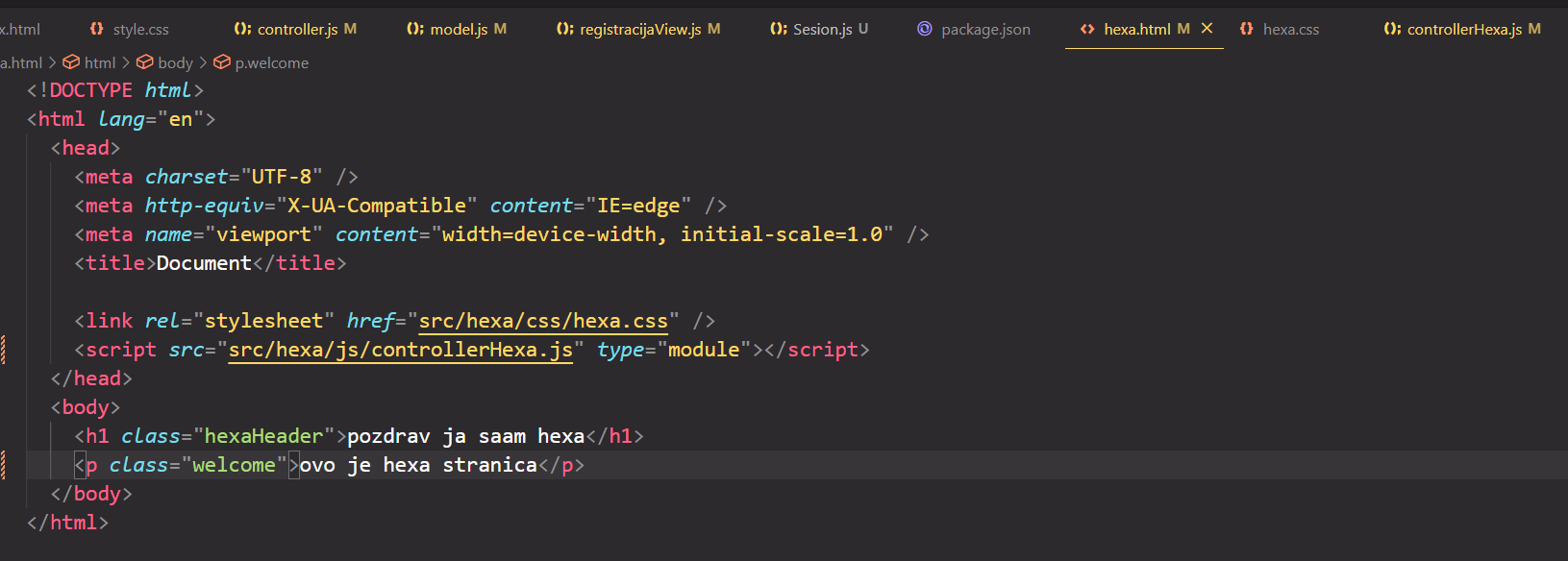
HEXA HTML – HEXA SCRIPTA

Sada kada nas window.location.href=hexa.html prenese sa index.html na hexa.html straniu moramo kreirati kako ce nam ta hexa.html stranica izgledati.Znaci moramo da napravimo hexa.html stranicu:  


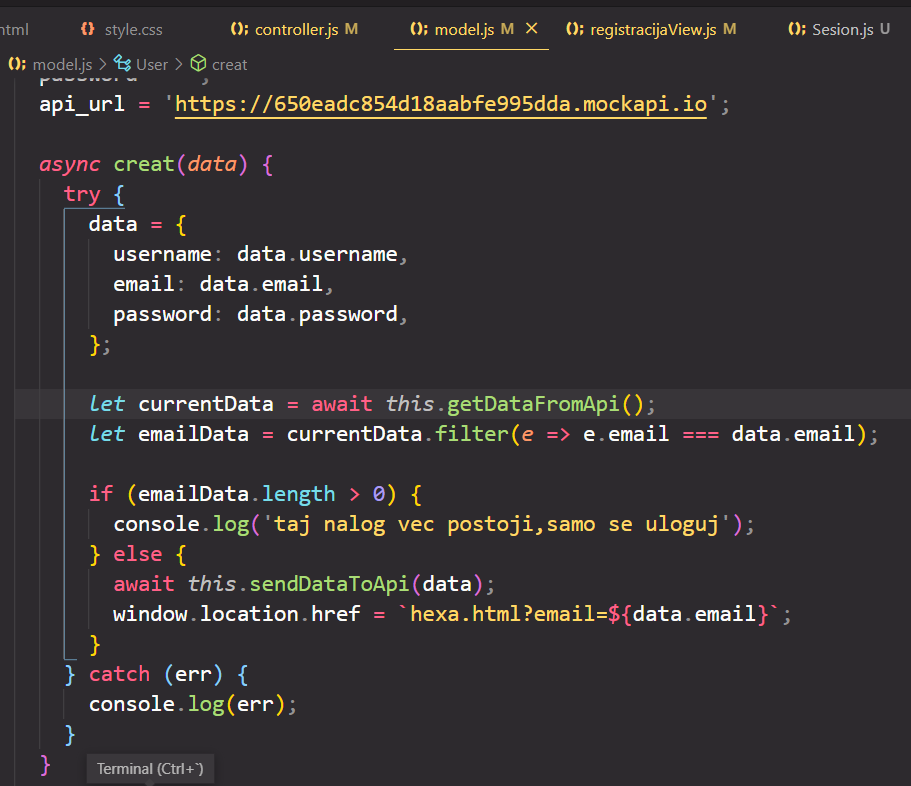
Ovde vidimo da smo u ovoj hexa.html stranici takodje napravili poziva hexa.css i controllerHexa.js koji nam se sada pokrecu iz hexa.html a ne iz index.html stranice.Dakle sada upravljamo sa stajlom posebnim css-som napravljenim saamo za ovu hexa.html stranicu .Takodje nam je scripta za hexa.html odvojena od skripte za index.html stranicu.

-------------------------------provjera da li nalog iz forme vec postoji u mockAPI-ju-----------------------------------

Dakle sada zelimo da zabranimo nasoj aplikaciji da kreira iste naloge ( da pravi duple naloge – iste naloge ) . To cemo uraditi tako ssto cemo postaviti program da prodje kroz svve mockAPI podatke i da provjeri da li se tu nalazi ijedan nalog sa unijetim podacima u formi.Ako se ne nalaze onda unesene podatke iz forme sacuvaj na mockAPI-ju, a ako podaci postoje napisi nam da ovi podaci vec postoje.

Dakle ovo cemo sve uraditi u creat() metodi u model.js fajlu:





vidimo da smo ovde u creat(data) metodi pod if uslov stavili da ispita da li nas emailData( podatak iz getDataFromAPI -> podaci iz mockAPI-ja ) ima vidse od 0 data.email( da li je email iz forme jednak emailu iz mockAPI-ja) te ako postoji vec ispisemo ovaj console.log, a u suprotnom ako ne postoji onda taj podatak posaljemo i sacuvamo na mockAPI-ju, te poslije toga da nam se dese ono nase cuveno prebacivanje sa index.html na hexa.html stranicu.

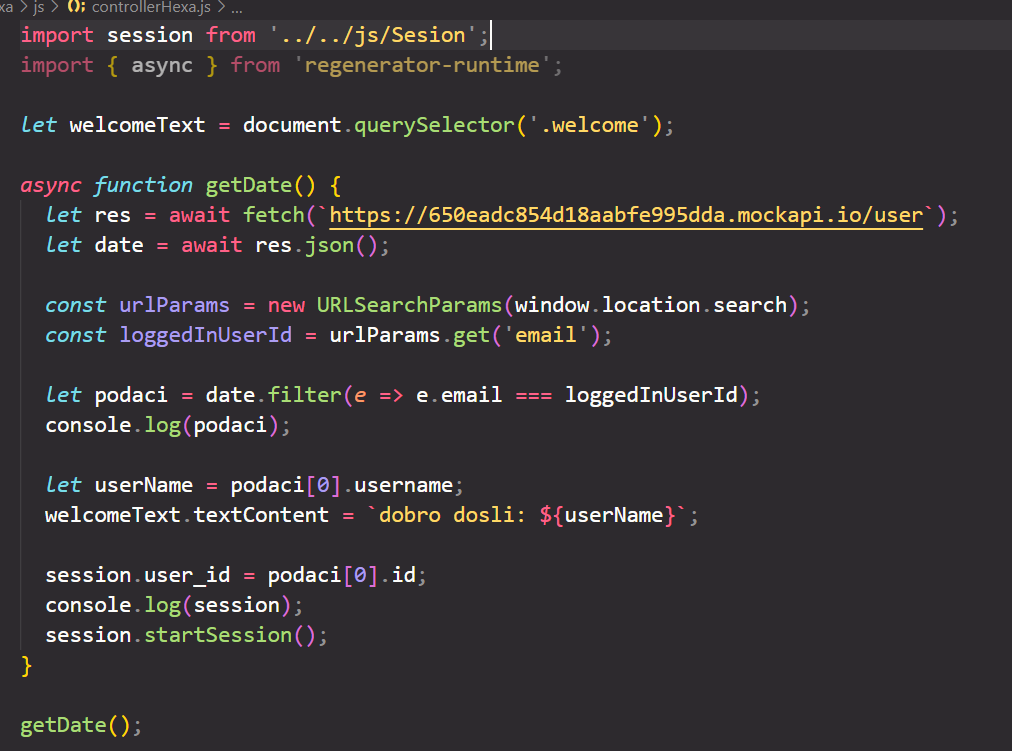
......................................................rad u controllerHexa.js falu sa Sessijama i podacima iz mockAPI-ja -> ali iz controllerHexa.js fajla...................................................................

Dakle zelimo da kreiramo jednu sessiju koju cemo koristiti za autodentifikaciju korisnika.Zelimo da nam provjerava da li je korisnik prijavljen na svojju stranicu ako je prijavljen da ostane na njoj a ako nije da ne moze preko URL-a da dodje do te stranice kucanjem npr u url dodatak da nas prebaci u hexa.html stranicu.Nas pocetni url u index.html stranici je : http://localhost:1234 , a kada nas prebaci sa indexa na hexa.html (prilikom ulogovanja) onda nas url izgleda ovako: http://localhost:1234/hexa.html, u zavisnosti pomocu kojek emaila smo se prijavili.

Sada da ne bi mogli ovo rucno uraditi dok se nalazimo u index.html da rucno upisemo ovo i da dospijemo doci do nekog profila i izvuci podatke sa tog profila bez da smo prosli autodentifikaciju to moramo sprijeciti. To radimo pomocu Sessija i kolacica.  
Dakle kreiracemo Session.js fajl u kome cemo da kreiramo kolacic:  
Dakle ovde vidimo da smo kreirali klasu Session u Session.js fajlu.U toj klasi cemo da definisemo startSession() funkciju.Takodje cemo tu kasnije da definisemo i kada zelimo da ubijemo kolacic(da se odjavimo sa kolacica).  
Ovde u ovom startSession() funkciji smo efinisali da nas kolacic vazi 2 dana : d.setTime(d.getTime() +2\*24....) => ovo definise 2 dana , dok smo to postavili na document.cookie.

Ovu klasu smo exportovali i sada zelimo da je uvezemo u controllerHexa.js fajl i da je tu koristimo.

..........................................OBJASNJENJE KAKO MOZEMO KORISTITI DIJELOVE URL-A (ovo nam nije potrebno za nas zadatak)............................................................

*controllerHexa.js:*

Ovde vidimo da smo importovali session klasu. Vidimo da smo kreirali funkciju getData() koja dohvaca podatke iz mockAPI-ja i sacuvava ih u data varijabli.

Vidimo da u urlParams smo kreirali novu instancu na nas URL.Objasnjenje iz chatCPT:  
Kod koji ste naveli kreira novu instancu **URLSearchParams** objekta na osnovu trenutnog URL-a veb stranice. Ovaj objekat se koristi za jednostavno rukovanje parametrima (query string) koji se nalaze u URL-u veb stranice.

Evo kako funkcioniše:

1. **window.location** predstavlja objekat koji sadrži informacije o trenutnom URL-u veb stranice.
2. **window.location.search** sadrži query string deo URL-a. Query string je deo URL-a koji sledi nakon znaka '?' i obično sadrži različite parametre u obliku **ključ=vrednost** koji su odvojeni znakom '&' (npr. **?param1=vrednost1&param2=vrednost2**).
3. **new URLSearchParams(window.location.search)** kreira novu instancu **URLSearchParams** objekta sa sadržajem query stringa trenutnog URL-a.

Nakon što se instancira **URLSearchParams** objekat, možete koristiti njegove metode za čitanje i manipulaciju parametrima iz query stringa. Na primer, možete koristiti **get** metodu da biste dobili vrednosti određenih parametara ili **set** metodu da biste postavili nove vrednosti parametara.

Evo primera kako biste mogli da koristite **URLSearchParams** objekat:

javascriptCopy code

// URL: https://www.example.com/page?param1=value1&param2=value2

const urlParams = new URLSearchParams(window.location.search);

// Dobijanje vrednosti parametara

const param1Value = urlParams.get('param1'); // 'value1'

const param2Value = urlParams.get('param2'); // 'value2'

// Postavljanje nove vrednosti parametra

urlParams.set('param3', 'value3'); // Dodaje param3=value3 u query string

Ovaj kod olakšava rukovanje parametrima iz URL-a veb stranice, što je korisno u mnogim situacijama, kao što su prenos parametara između različitih stranica ili obrada parametara u JavaScript kodu.

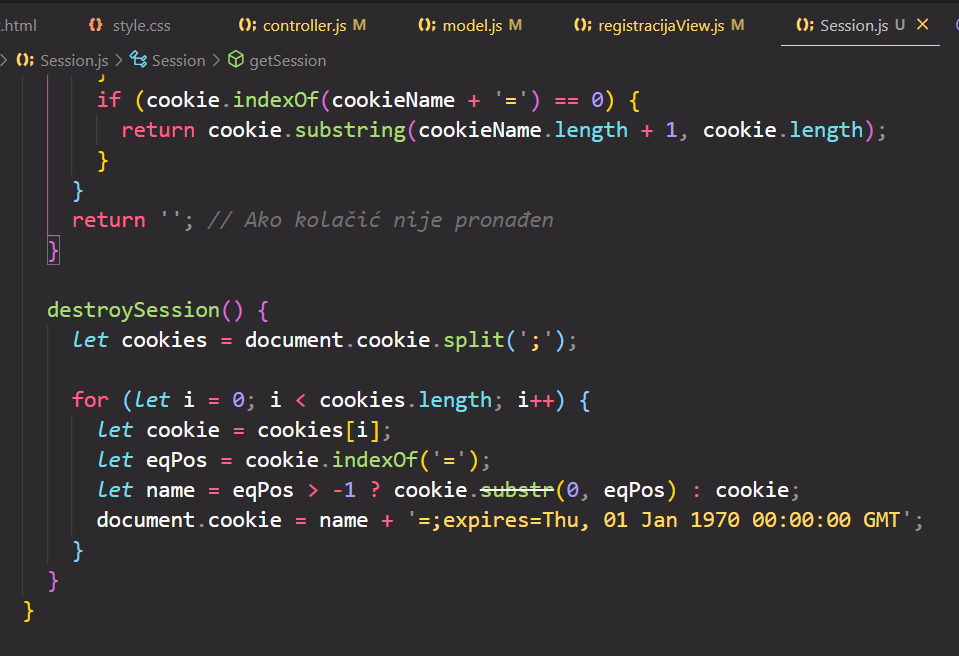
Mi u nasem primjeru koda zelimo da dobimjemo email iz URL-a ( za sada nam ne treba ali ga ovako dobijamo ) ovo sto smo sacuvali u loggedInUserId varijabli.

Onda smo uradili da pomocu filter () metode prodjemo kroz podatke iz api-ja i zelimo da nam vrati one podatke iz mockAPI-ja koji se poklapaju sa podacima tj. emailom iz URL-a:

*let* podaci = date.filter(*e* => e.email === loggedInUserId);

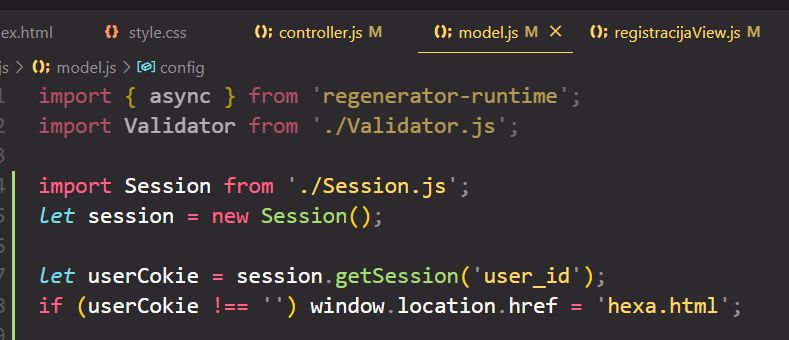
i ovo sada iskoristimo da bi dobili userName te da bi mogli da ubacimo ovaj userName kao poruku dobrodoslice u hexa.html stranicuu.  
  
  
  
.....................................................................................POKRETANJE SESSIJE (pravljenje kolacica => cookie-ja) ...........................................................................................

Kao sto smo gore rekli => zelimo da napravimo cookie koji se pravi onog trenutka kada se podaci iz forme posalju na mockAPI.  
Da bi smo to napravili najlakse nam je napraviti Session.js fajl sa metodama za pravljenje cookie-ja( startSession), metodu kojom cemo birati napravljeni cookie( getSession) ii metoda kojom cemo bristi cookie-je ( destroySession):



Dakle ovde sada imamo Session.js fajl kojeg cemo da exportujemo na mjesta gdje nam treba.

Sada cemo da u model.js fajl uvezemo ovu Session.js fajl i njegovu funkciju:

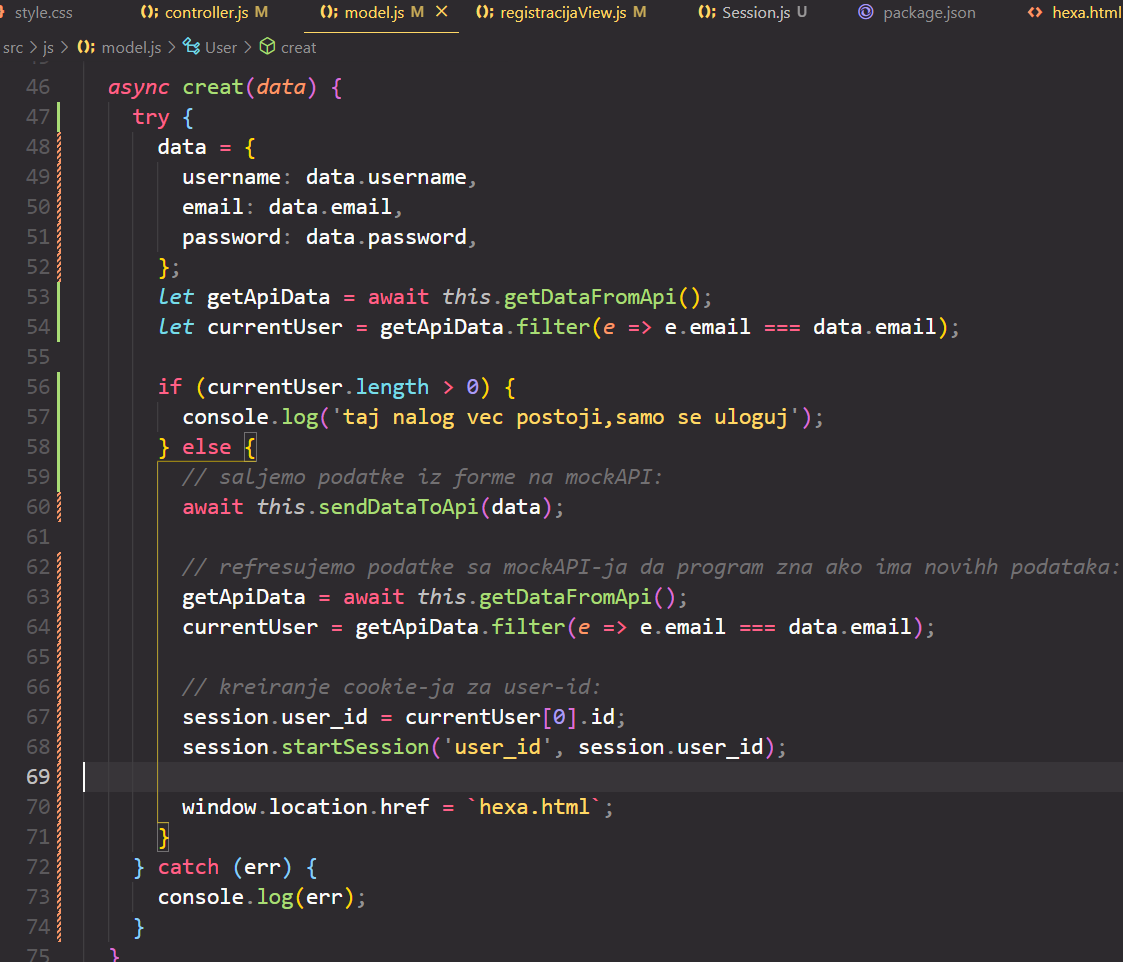


Pored toga sto smo uvezli import Session from ‘./Session.js’;

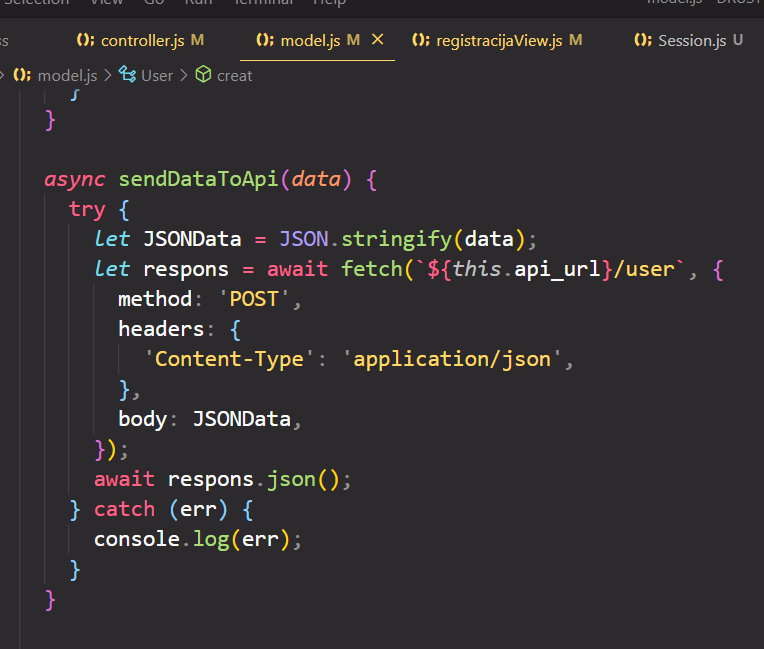
takodje smo uradili i sledeci korak koji nam je jako bitan za vertifikaciju korisnika.Vertifikacija korisnika je onaj dio kada provjeravamo da li je korisnik na stranici i ako jeste onda moze da otvori svoj profil na hexa.html stranici (kasnije cemo objasniti kako smo to uradili pomocu id-ja kojeg smo poslali na mockAPI, tj. pomocu cookie-ja) , a ako ne postoji cookie => to znaci da korisnik nije prijavljen i da se ne moze ulogovati na hexa.html stranicu . Da ovo nismo uradili neko bi jednostavno mogao da promjeni URL sajta i samo da na root url doda ‘/hexa.html’ i pretrazivac bi ga automacki prebacio na hexa.html iako cookie ne postoji i korisnik nije prijavljen.To zabranjujemo ovim if uslovom koji kaze ako :

session.getSession(‘user\_id’) cookie postoji (tj. ako je razlicito od praznog stringa ‘ ‘) onda nas drzi na ‘hexa.html’ stranici.

Sada treba da razmisljamo na ovaj nacin,kada se desi form ‘submit’ dogadjajj’ zelimo sem sto u tom trenutku salje podatke na mockAPI ,takodje zelimo i da kreiramo cookie u tom trenutku i to kreiracemo cooki koji ce nam cuvati podatake trenutno ulogovanog korisnika tj. cuva ce nam broj njegovog id-ja:

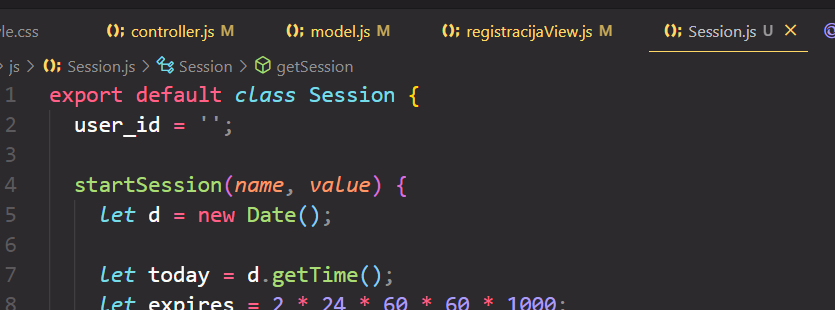
Dakle kreirali smo u model.js fajlu u creat() funkciji => jedan splet dogadjaja u if() dijelu koda gdje smo rekli da ako je if uslov ispunjen tj. ako je currentUser.length>0 (ako ovaj currentUser vec postoji ) onda zelimo da nam se ispise samo poruka u konzoli da ovaj user vec postoji a ne da mozemo kreirati dupli user.

A ako takav user ne postoji onda smo rekli u ‘else’ dijlu koda da podatke iz forme posaljemo na mockAPI funkcijom koja se naziva sendDataToApi( data ) => ova funkcija salje podatke na mockAPI:



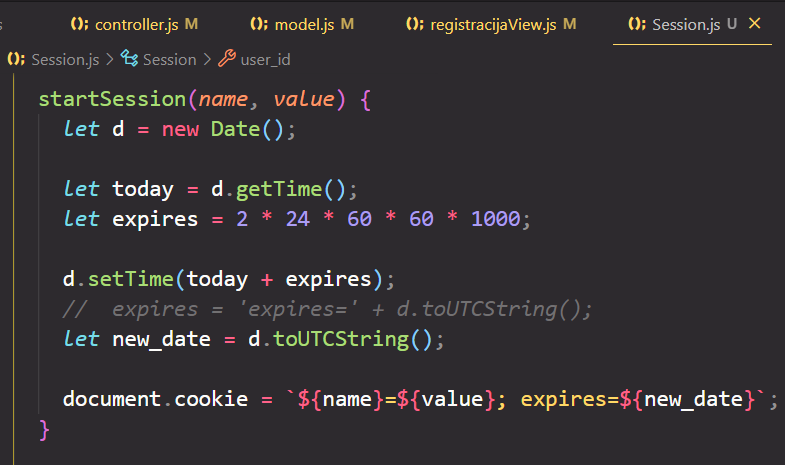
Sada kada smo poslali podatke iz forme na mockAPI , zelimo da nam se ti podaci azuriraju u nasoj funkciji da nas kod ima pristup tek dodatim podacima na mockAPI. To radimo tako sto smo samo restartovali getApiData i currentUser varijable u else dijlu koda.

Sada kada ovaj nas kod ima pristup restartovanim podacima iz mockAPI-ja sada mozemo da kreiramo ‘user\_id’ cookie. Kreiramo ga pomocu Session objekta kojeg smo ranije kreirali u Session.js fajlu (pa smo taj objekat tj. tu clasu exportovali ovde u model.js fajl i sada ga koristimo ovde kao session).Dakle session.user\_id=currentUser[0].id => ovo nam cuva vrijednost trenutnog id-ja u Session.js fajlu gdje smo kreirali:

Vidimo da smo u session.js fajlu kreirali svojstvo user\_id.E upravo tu smo sacuvali ovu nasu vrijednost iz model.js fajla.

Te sada pomocu session.startSession() metode kreiramo navu sessiju (novi cookie sa 2 parametra: ime cookija i vrijednost tog imena);

        session.startSession('user\_id', session.user\_id);

zato sto nam startSession trazi ova 2 parametra:  


Te sada kada smo kreirali cookie u model.js fajlu sada jos u istom fajlu i u istoj creat() funkciji govorimo nasem programu window.location.href=’hexa.htm’ da zelimo da nas prebaci na hexa.html stranicu.