აბსტრაქტი

მრავალი თანამედროვე ონლაინ პლატფორმის მიბაძვით, ჩვენ დავწერეთ ერთმანეთისგან სრულიად დამოუკიდებელი ორი პროექტი, ე.წ. სერვერი და ფრონტი. Homeworker-ის სერვერი საუბრობს JSON ენაზე. გამოძახებებს შორის სერვერი არ ცვლის STATE-ს და ყოველი გამოძახება მოიცავს ყველა საჭირო ინფორმაციას, იმისთვის რომ მოთხოვნა დაკმაყოფილებულ იქნას. Homeworker-ის ფრონტი კი მუშაობს React-ის გამოყენებით NODEJS-ის framework-ით და საუბრობს სერვერთან JSON-ის საშუალებით.

აღსანიშნავია ასევე რომ ჩვენი დიდი ძალისხმევის წყალობით, სერვერული კოდის > 95% არის ტესტირებადი (და დატესტილი), 100% არის კომენტირებული და სერვერული აპლიკაციის API არის Scenario-Test-ტიპის ტესტებით შემოწმებული გარდა ჩვეულებრივი JUNIT-ტესტებისა.

მიუხედავად ამისა, ჩვენ მაინც ვთვლით რომ ამ პროექტში მეტის გაკეთება იყო შესაძლებელი, ოდნავ უფრო მეტი დრო რომ გვქონოდა (და მეტი მეილსტოუნი).

სერვერული API-ს მოკლე Walk-though

პირველ რიგში, ვიძახებთ სესიის ასაღებ url-ს.

POST /sessions

რომელიც დაგვიბრუნებს JSON-ს ასეთ ფორმატში :

{STATUS : “OK”, sessionId : “123-abcd-123”}

მიღებული ცვლადი sessionId არის შენახული სერვერზეც, და ყოველ შემდეგ ჯერზე, როდესაც დაგვჭირდება ჩვენს ინფორმაციაზე წვდომა, შეგვიძლია გამოვიყენოთ ის ხელახლა.

მასე მერე რაც სესია დავჯავშნეთ, შეგვიძლია რომ დავლოგინდეთ:

POST /hasSession/login

ჰედერით:

sessionId - “123-abcd-123”

მოთხოვნით:

{email : “example@asda”, password : “pass”}

ყურადღება მიაქციეთ რომ ამ შემთხვევაში გავდივართ /hasSession ფილტრს, რომელიც ამოწმებს რომ HTTP-request-ის ჰედერში გვაქვს პარამეტრი sessionId ვალიდური სესიის იდენტიფიკატორით.

წარმატებული დალოგინების შემდგომ შეგვეძლება რომ გამოვიძახოთ პოსტის შექმნის მოთხოვნა, შესაბამისი JSON-ით:

POST /hasSession/isLoggedIn/posts

ჰედერით:

sessionId - “123-abcd-123”

მოთხოვნით:

{post : { contents :“example@asda”}, categories : [1,2,3] }

ეს შექმნის პოსტს მომხმარებლის სახელზე და დაამატებს მას 1, 2 და 3-ე კატეგორიებში.

ვთქვათ ახლა იუზერმა მოინდომა პოსტის დალაიქება.

ის იძახებს შემდეგ მოთხოვნას:

POST /hasSession/isLoggedIn/likeThePost

ჰედერით:

sessionId - “123-abcd-123”

მოთხოვნით:

{postId : 2}

ეს მოთხოვნა დაამატებს მომწონებელთა სიაში ჩვენს იუზერს. ხელახალი გამოძახება გადააქცევს ამ მოწონებას - არ-მოწონებაში.

თუ მომხმარებელმა საკუთარი პოსტის შეცვლა, ის გამოიძახებს:

PUT /hasSession/isLoggedIn/posts

ჰედერით:

sessionId - “123-abcd-123”

მოთხოვნით:

{id : 2, contents : „ გთხოვთ A გამოგვაყოლეთ “}

და პოსტი ნომერი 2 - თუ ის შექმნა ამ მომხმარებელმა - მაგიურად შეიცვლის თავის contents-ს, წინააღმდეგ შემთხვევაში, მომხმარებელი მიიღებს ამ მესიჯს :

{STATUS : ERROR, ERRORMESSAGE : “ This is not your post! ” }

და ბოლოს, მომხმარებელს უნდა რომ გამოვიდეს სისტემიდან :

POST /hasSession/isLoggedIn/logout

ჰედერით:

sessionId - “123-abcd-123”

და

ცარიელი მოთხოვნით.

ეს მოთხოვნა მონიშნავს სესიას როგორც დამთავრებულს, რის შემდეგაც უკვე შეიძლება მისი წაშლა.

ამ და სხვა მრავალ Feature-ს წარმოგიდგენთ ჩვენი პატარა პროექტის ფარგლებში.

რაც შეეხება ფრონტს

ფრონტი დაიწერა სრულიად REACT-ში NODEJS-ის გამოყენებით. გამომდინარე იქიდან რომ JS-ს ნაკლებად ვიცნობთ, ტესტები არ გვიწერია ფრონტისთვის. მიუხედავად ამისა, ჩვენი ფრონტ აპლიკაცია არის ეფექტურად ერთ-გვერდიანი ვებ-საიტი, რომელიც იძახებს ბექ-ენდის URL-ებს და თავად ინახავს მინიმალურ ინფორმაციას მხოლოდ.