

PREDMET IT381 ZAŠTITA I BEZBEDNOST INFORMACIJA

Jsenji semestar 2016/2017

Bezbednost tehnologije autentifikacije i čuvanja sesija u veb aplikacijama

Dušan Stanojević 1940

dusan.stanojevic.1940@metropolitan.ac.rs

Beograd, 2017

# Abstrakt

Iako je u prošlosti delovalo kao da će aplikacije budućnosti biti sve bezbednije, problem bezbednosti podataka i pristupa u veb aplikacijama postaje sve češći. Propusti prilikom izrade veb aplikacija imaju sve veće i veće posledice kako sistemi preuzimaju ozbiljnije delove poslovanja. Preventivne kontrole su jedne od četri osnovnih kontrola bezbednosti aplikacija koje rade u oblaku i na vebu generalno. Postoji puno načina da se autentifikacija kao deo preventivne kontrole dizajnira i implementirza kao i puno protokola autentifikacije ali nisu svi podjednako efikasni i bezbedni. Ovaj rad analizira osnovne greške dizajna autentifikacije, implementacije autentifikacije, protokola autentifikacije i analizira njihove prednosti i mane. U radu je dat konkretn primer implementacije Jason Web Token protokola na serverskoj strani sa Java programskim jezikom.

Ključne reči: Bezbednost veb aplikacija, Java, Autentifikacija, Validacija, Sesija, Napad

Sadržaj

[Abstrakt 1](#_Toc484109186)

[1. Uvod 3](#_Toc484109187)

[1.1 Opis problema 3](#_Toc484109188)

[2. Česte greške prilikom dizajna autentifikacije 4](#_Toc484109189)

[2.1 Loše šifre 4](#_Toc484109190)

[2.2 Neograničenost pokušaja 4](#_Toc484109191)

[2.3 Detaljne greške 5](#_Toc484109192)

[2.4 Nebezbedno prenošenje kredencijala 6](#_Toc484109193)

[2.5 Promena šifre 7](#_Toc484109194)

[2.6 Zapamti me 7](#_Toc484109195)

[2.7 Predvidljiv Password 7](#_Toc484109196)

[3. Primer dobrog dizajna autentifikacije 8](#_Toc484109197)

[4. Česte greške prilikom implementacije autentifikacije 9](#_Toc484109198)

[4.1 Mehanizam - Otvoreno prilikom greške 9](#_Toc484109199)

[5. Čuvanje stanja aplikacije 10](#_Toc484109200)

[5.1 Podaci u cookie-u sa hash-om i privatan ključ na serveru 10](#_Toc484109201)

[5.2 Podaci na serveru i ključ u cookie-u klijenta 11](#_Toc484109202)

[5.3 Podaci na serveru i ključ kao GET parametar 12](#_Toc484109203)

[5.4 Podaci na serveru i ključ kao deo zaglavlja zahteva 12](#_Toc484109204)

[6. Protokoli autentifikacije veb aplikacija 13](#_Toc484109205)

[6.1 Protokol autentifikacije ime i šifra 13](#_Toc484109206)

[6.2 Protokol autentifikacije oAuth 15](#_Toc484109207)

[6.3 Protokol autentifikacije oAuth2 16](#_Toc484109208)

[6.4 Protokol autentifikacije Json Web Token (JWT) 16](#_Toc484109209)

[7. Implementacija protokola JWT za autentifikacije veb aplikacija 19](#_Toc484109210)

[7.1 Implementacija JWT-a 19](#_Toc484109211)

[7.2 Testiranje implementacije JWT-a 23](#_Toc484109212)

[5. Literatura 26](#_Toc484109213)

# 1. Uvod

## 1.1 Opis problema

Na prvi pogled, potvrda identiteta je konceptualno među najjednostavnijim od svih sigurnosnih mehanizama korišćenih u veb aplikacijama. U tipičnom slučaju, korisnik snabdeva svoje korisničko ime i lozinku, a aplikacija mora da potvrdi da su ovi podaci dobri i ako jesu da mu omogući pristup određenim delovima sistema.

Authentication takođe leži na srcu zaštite aplikaciju protiv zlonamernih napada. To je linija fronta odbrane od neovlašćenog pristupa. Ako napadač može da pobedi ovu odbranu, on će često dobiti punu kontrolu nad funkcionalnosti aplikacije i neograničen pristup podacima koji se u njoj nalaze. Bez snažnog dizajna i implementacije autentifikacije, nijedna drugi sigurnosni mehanizam (kao što je upravljanje sesijom i kontrola pristupa) ne mogu biti efikasni.

U stvari, uprkos očiglednoj jednostavnosti, osmišljavanje sigurne provere identiteta je težak posao. U stvarnom svetu veb aplikacijama provera identiteta često je najslabija karika, što omogućava napadaču neovlašćen pristup.

U daljem teksu će biti sagledane greške prilikom dizajna autentifikacije kao i različiti načini implementacije čuvanja stanja aplikacije kao i protokoli autentifikacije.

Poseban fokus biće bačen na Jason Web Token protokol i biće konkretno implementiran.

# 2. Česte greške prilikom dizajna autentifikacije

Dizajn autentifikacije je jako bitan prilikom izrade veb aplikacija. U ovom poglavlju su sagledani neke od čestih grešaka prilikom dizajna autentifikacije u veb aplikacijama.

## 2.1 Loše šifre

Loše šifre, to jest šifre koje je lakše pogoditi, u veb aplikacijama su česte ukoliko aplikacije ne podržavaju stroga pravila za dozvoljene šifre.

Najčešća pravila čiju restrikciju aplikacije ne podržavaju su:

1. Krate ili prazne šifre
2. Česte reči iz rečnika
3. Iste lozinke kao reči
4. Podešene na istu početnu vrednost

Ukoliko aplikacije ne podržavaju stroga pravila prilikom unosa lozinke veće su šanse da neki od njihovih korisnika imaju slabe lozinke što čini šanse da ne autorizovana osoba prođe autentifikaciju pogađanjima lozinki veća.

## 2.2 Neograničenost pokušaja

Izrada prijave na sistem gde aplikacije ne limitiraju broj neuspešnih pokušaja je poziv za napadače da pokušavaju kombinacije imena i lozinke koje će im omogućiti ne autorizovan pristu. Ovo predstavlja veliku ranjivost sistema. Nakon analiza velikih količina lozinki koje su ukradene sa velikih sajtova ovo su bile najčešće korišćene lozinke[1]:

1. password
2. *website name*
3. 12345678
4. qwerty
5. abc123
6. 111111
7. monkey
8. 12345
9. letmein

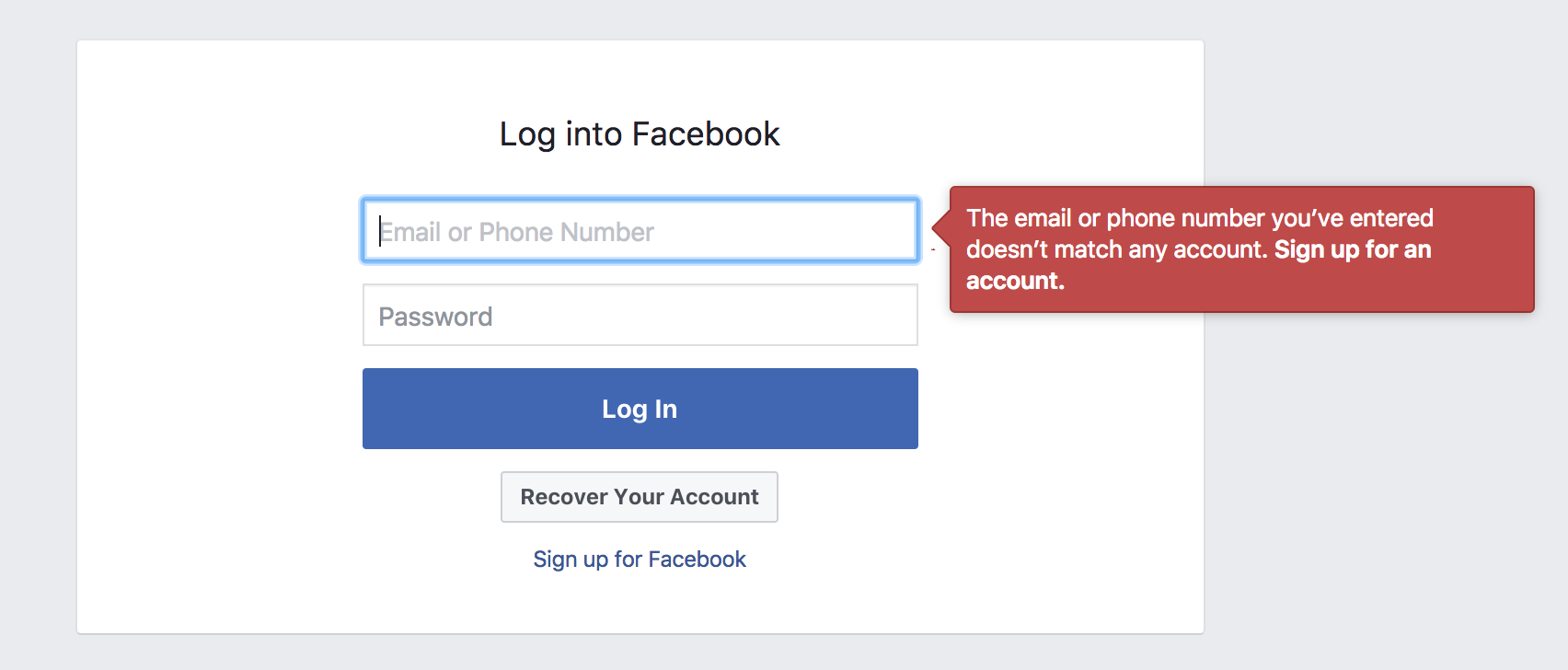
Ukoliko napadač koristi automatizovan pristup pogađanju šifre brzina kojom će pogoditi šifru se povećava.

## 2.3 Detaljne greške

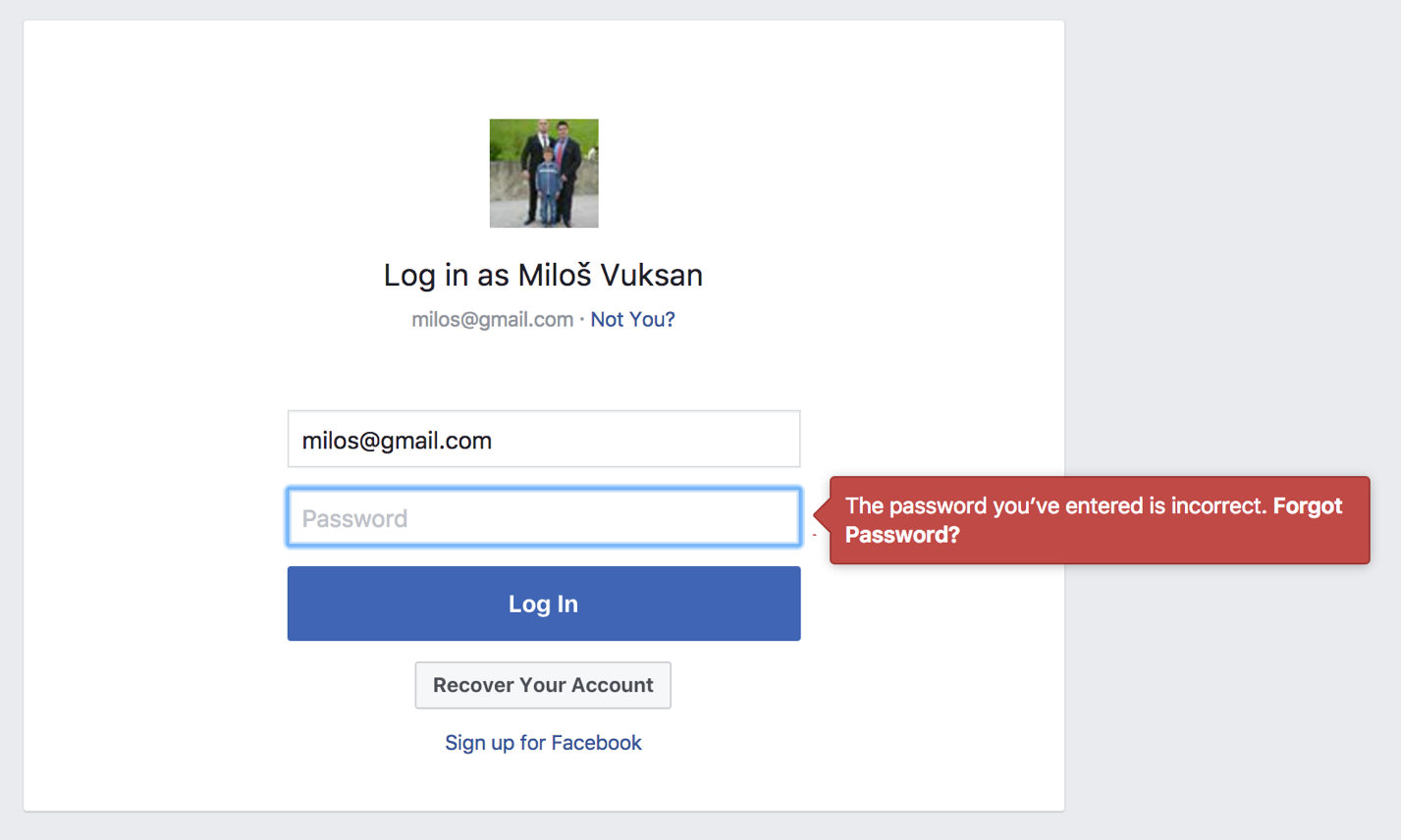
Tipiče login forme zahtevaju da korisnik unese bar ime i lozinku. Mnoge aplikacije zahtevaju i dodatne podatke u vidu datuma rođenja ili PIN koda.

Kada autentifikacija ne prođe može se zaključiti da jedan od parametara nije dobar, ali ukoliko postoji detaljna poruka o grešci može da se zaključi da su određeni podaci tačni i time da se znatno olakša napad pogađanjem.

Primer ovakve greške može se videti na slikama 2.3.1 i 2.3.2.



Slika 2.3.1



Slika 2.3.2

## 2.4 Nebezbedno prenošenje kredencijala

Ukoliko veb aplikacija podržava http protokol radije nego https protokol moguće je uraditi napade “man in the middle” i prisluškivati na protok saobraćaja na mreži.  
  
Parametari zahteva u http standardu nisu enkriptovani stoga je moguće pročitati kredencijale korisnika.

Jedan primer ovakvog sajta je LAMS Univerziteta Metropolitan.

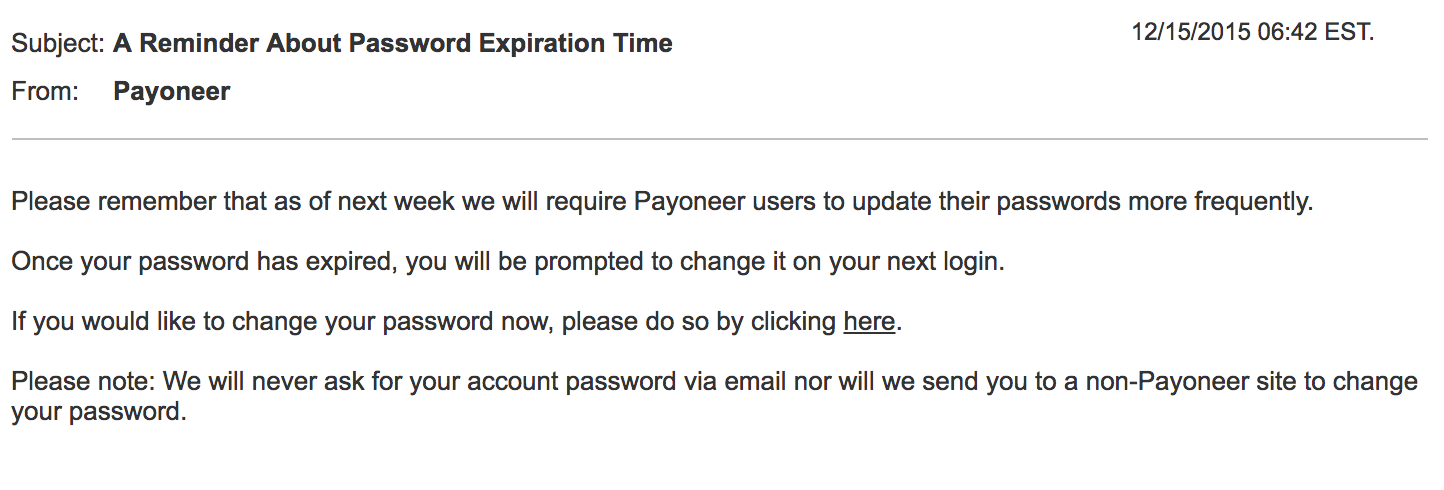
Primer URL-a login-a:

<http://lams.metropolitan.ac.rs:8080/lams/index.do?username=dusan.stanojevic.1940&password>

Iako je lozinka enkriptovana napadač može da uzme enkriptovanu vrednost i uradi login koristeći ga.

## 2.5 Promena šifre

Promena šifre je bitan deo veb aplikacija jer omogućava korisnicima da brzo promene svoju lozinku ukoliko misle da je neko dobio pristup istoj. Mnoge veb aplikacije ne podržavaju promenu šifre što je samo jedan deo problema. Dobar način da se poveća bezbednost je periodična obavezna promena šifre. Mnogi bankovni sistemi poput payoneer-a teraju korisnika da jednom godišnje promeni šifru, što se može videti na slici 2.5.1.



Slika 2.5.1

## 2.6 Zapamti me

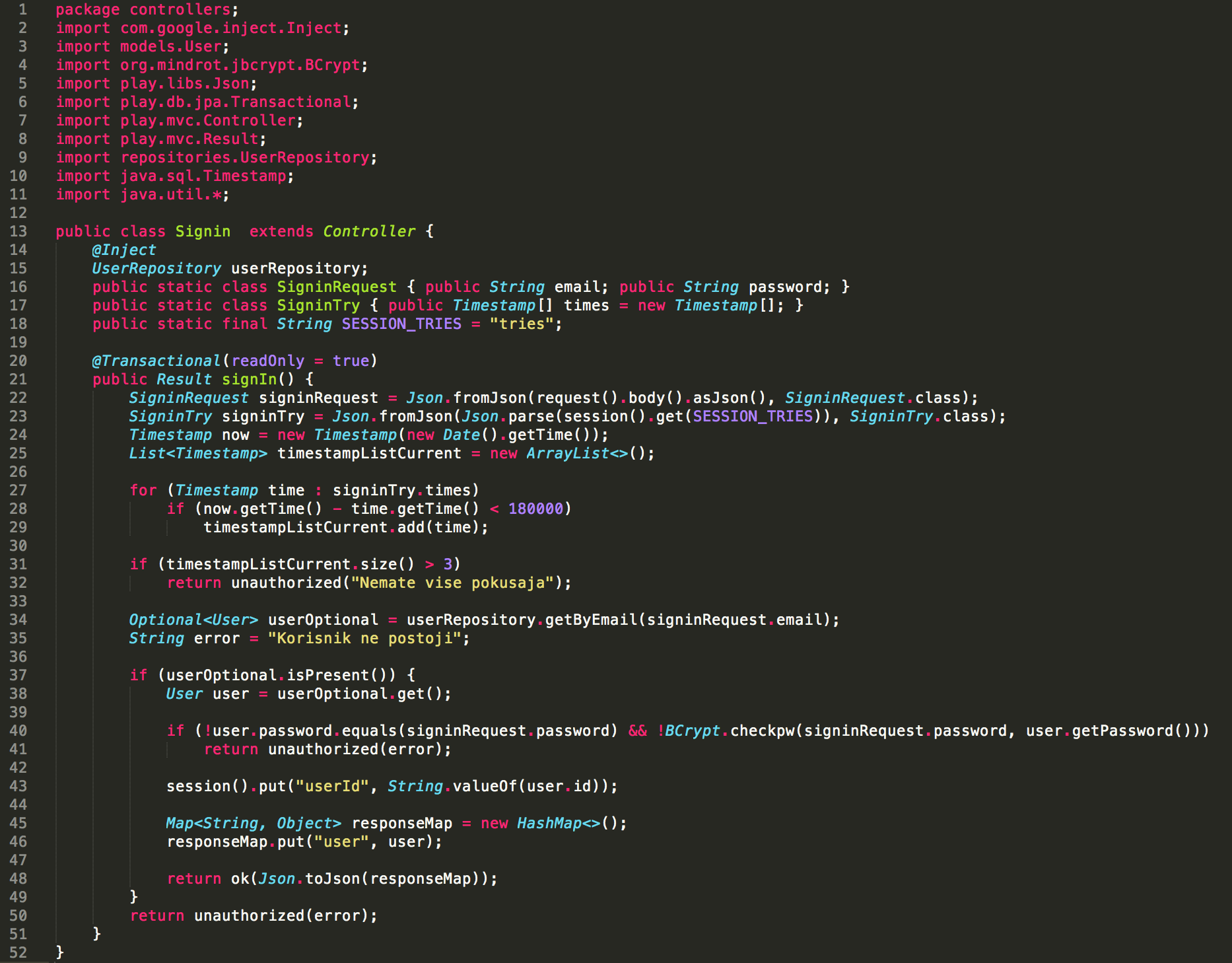
Aplikacije često implementiraju zapamti me funkcionalnost tako što ne rade ponovnu proveru šifre korisnika i samo zapamte korisnika po prosleđenom korisničkom imenu. Ukoliko napadač naiđe na ovakav sistem i poznaje korisnikovo ime može odmah da se uloguje.

## 2.7 Predvidljiv Password

Neke aplikacije prilikom kreiranja korisnika dodeljuju automatski generisane šifre. Ukoliko ove šifre budu otkrivene moguće je dobiti pristup nalogu pre nego što pravi korisnik promeni šifru.

# 3. Primer dobrog dizajna autentifikacije

Korišćenjem Play Framework-a i Java programskog jezika daćemo primer REST login forme koja prati stavke iz poglavlja 2.



Slika 3.1 Primer Java koda za autentikaciju korisnika

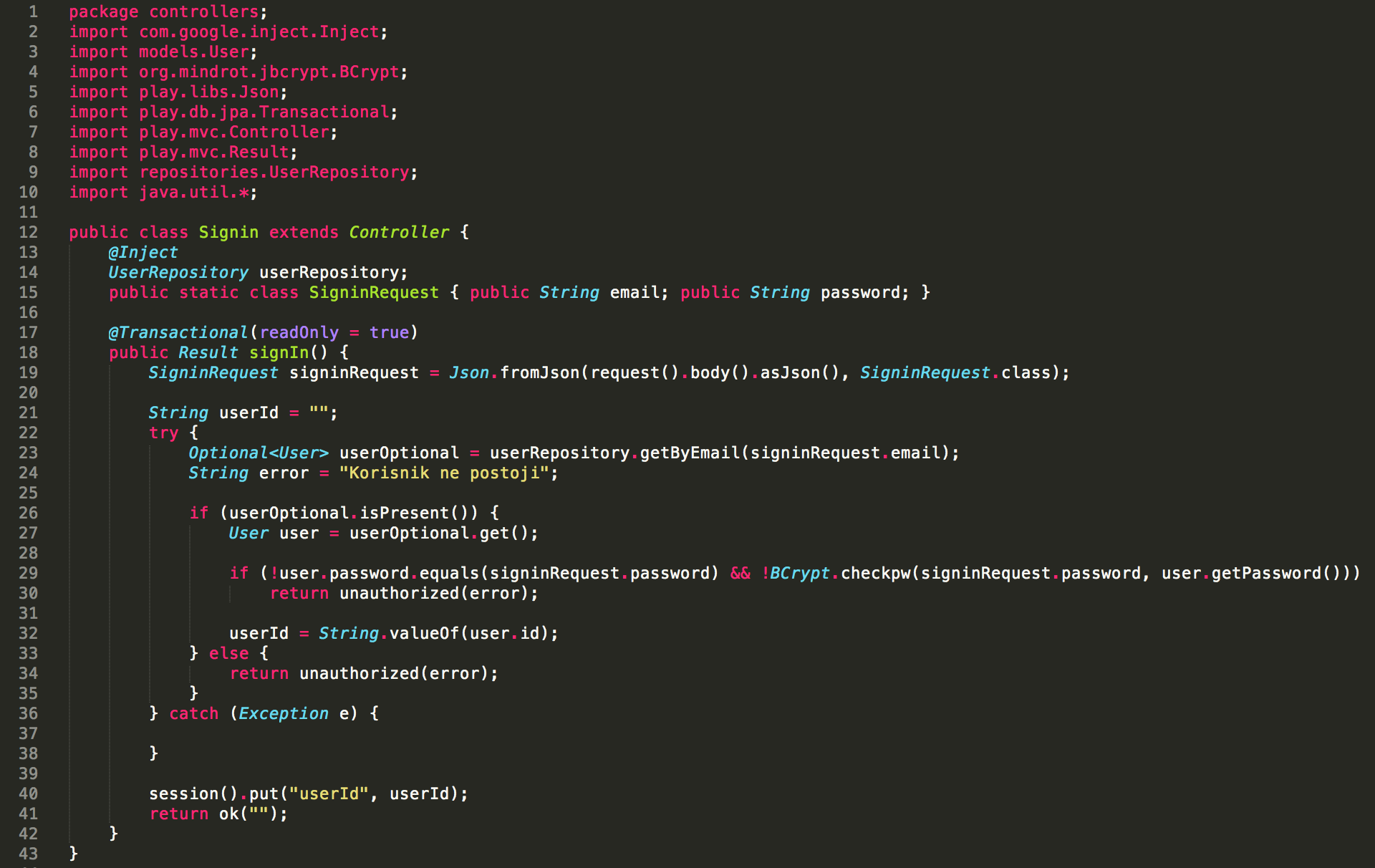
# 4. Česte greške prilikom implementacije autentifikacije

Čak i dobro dizajnirani sistemi za autentifikaciju mogu da budu ranjivi. Osnovni problemi prilikom autentifikacije korisnika prilikom dobrog dizajna autentifikacije dolaze iz načina na koji je autentifikacija sa serverom implementirana.

## 4.1 Mehanizam - Otvoreno prilikom greške

Otvoreno prilikom greške je implementaciona greška implementacije gde se autentifikacija korisnika dozvoljava ukoliko dođe do greške prilikom implementacije autentifikacije.

Delovi koda za hvatanje grešaka se obično koriste radi enkapsulacije poziva ka bazi podataka, pošto autentifikacija korisnika često zahteva pristup bazi ovakve greške su česte.  
  
Primer java koda koji daje ovakvu grešku u Play framework-u se može videti na slici 4.1.1



Slika 4.1.1

# 5. Čuvanje stanja aplikacije

Čuvanje podataka u sesiji je jedan od osnovnih mehanizama bezbednosti većine veb aplikacija. Čuvanje podataka u sesiji je ono što omogućava aplikacijama jedinstevno identifikovanje korisnika preko više različitih zahteva.

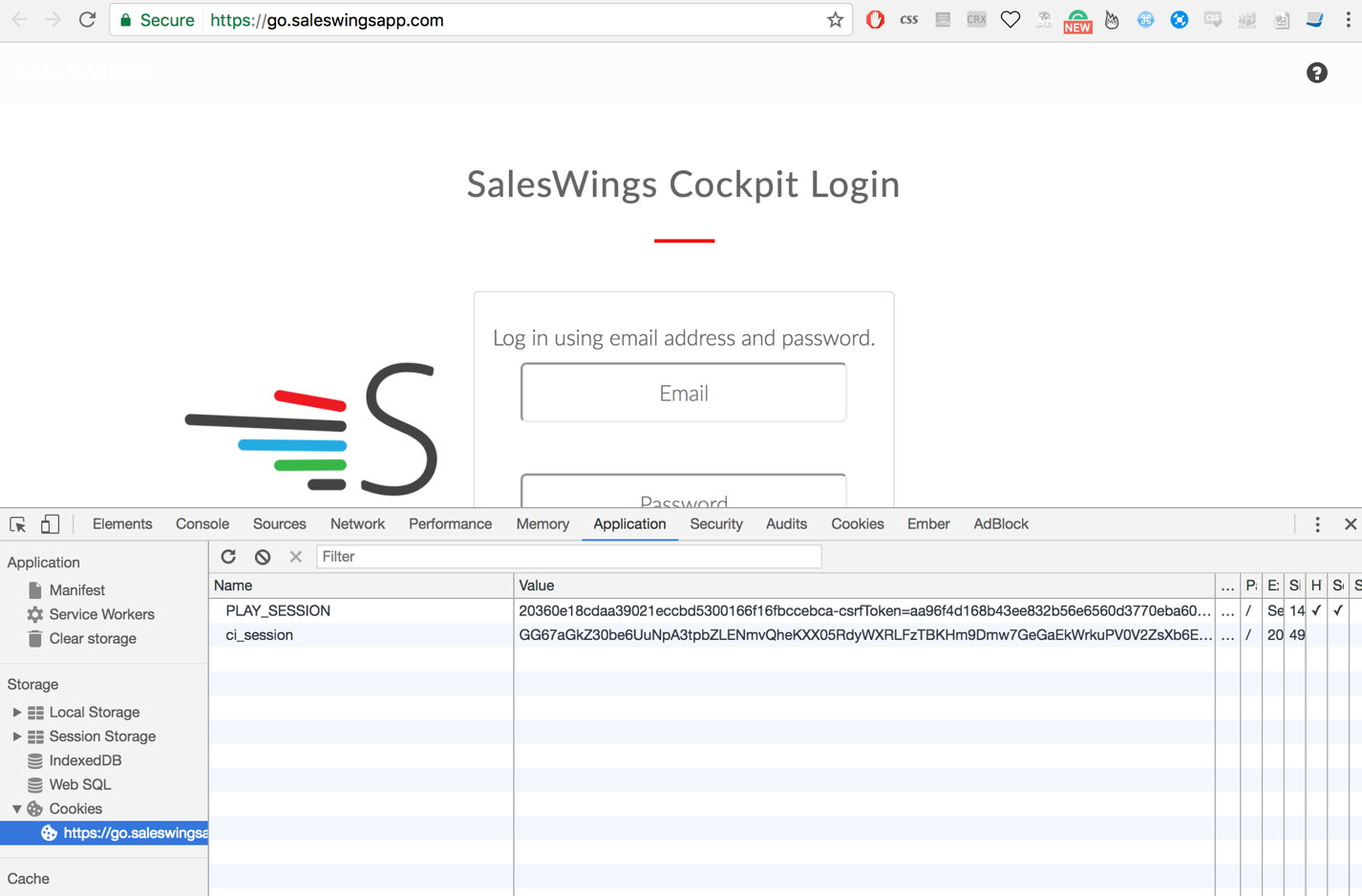
Mnogi napadači biraju da napadnu kako aplikacija čuva stanje. U nastavku poglavlja će biti analizirani neki od mehanizama za čuvanje podataka u aplikaciji.

## 5.1 Podaci u cookie-u sa hash-om i privatan ključ na serveru

Čuvanje podataka sesije u Cookie polju na klijentu i praćenje hash-a enkriptovanog sa tajnom sa server-u. Ovakav pristup čuvanju podataka veb aplikacija vezanih za jednu interakciju korisnika sa sajtom je korišćen od strane mnogih radnih okvira kao očekivan, jedan ovakav radni okvir je i Play Java radni okvir koji smo koristili za primere iz poglavlja 3 i 4.

Ovaj pristup ima prednosti u tome što server čuva samo jedan podatak radi validacije hash-a same sesije dok klijentska aplikacija mora da čuva i šalje podatke vezane za sesiju. Ovaj pristup ima manu u tome što parametri Cookie polja mogu da budu pročitani za http zahteva jer nisu enkriptovani. Jako sličan pristup ovome je JWT pristup koji je opisan u poglavljima 6.4 i implementiran u poglavlju 7.

Ovaj pristup možemo videti iz pokretanja aplikacija iz poglavlja 2, 3 i 4 kao i na slici 5.1.1

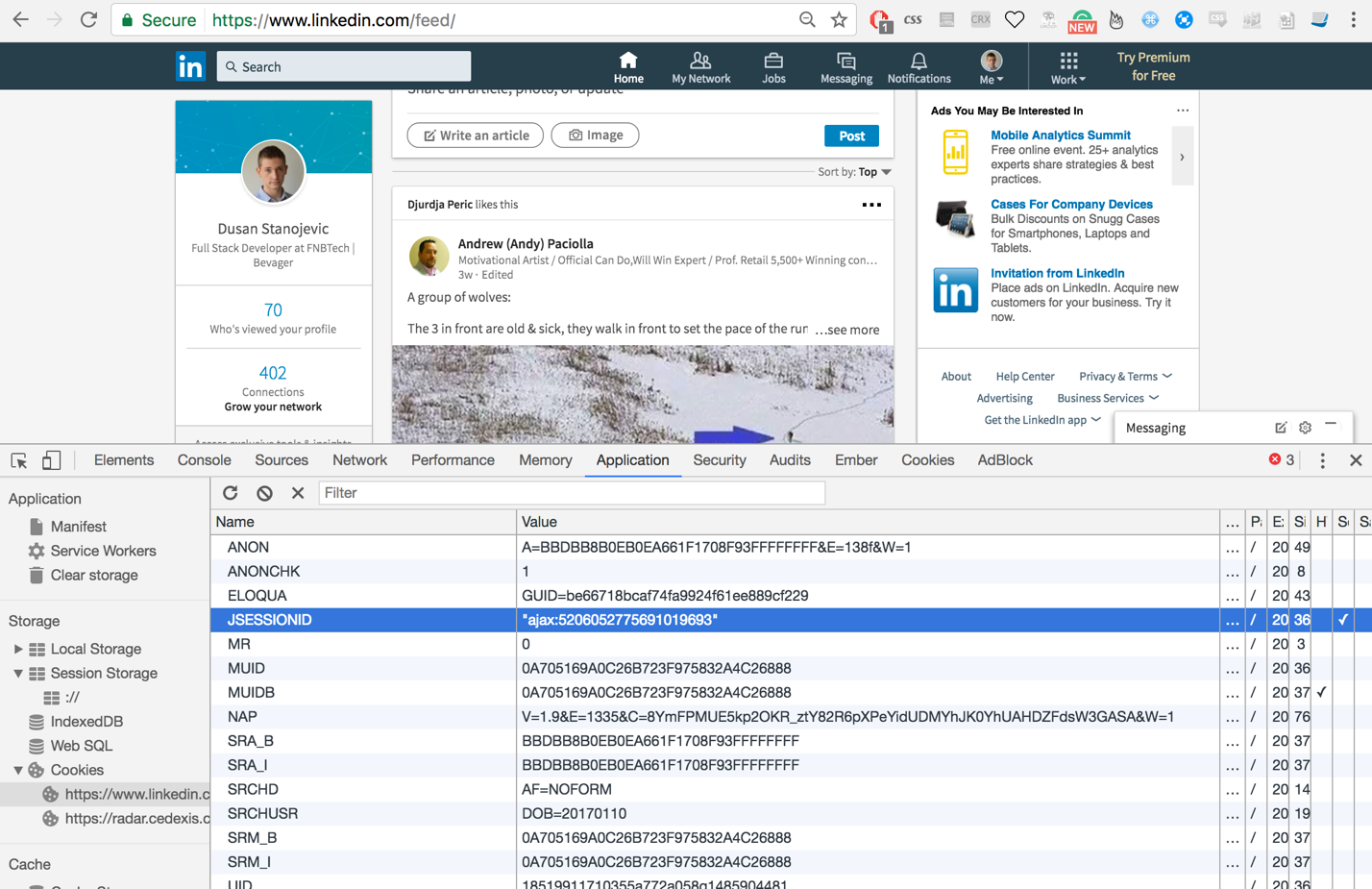


Slika 5.1.1

## 5.2 Podaci na serveru i ključ u cookie-u klijenta

Svi podaci za sesiju bivaju sačuvani u sesiji na serverskoj strani a na klijentskoj strani čuvamo samo vrednost sessije. Ovaj način čuvanja sesija korsite mnogi veb sajtovi jer ima prednost u tome što su zahtevi manji, jer se šalje samo jedan parametar i dobar je pristup za veb sajtove koji šalju i primaju puno podataka.

Problem sa ovim pristupom je u tome što ukoliko ne postoje dodatne mere zaštite napadač može izmenom vrednosti token-a na klijentskoj strani da uradi napad pod nazivom “session hijacking” i preuzme sesiju validnog korisnika. U ovom napadu napadač mora da zna ključ sesije, ali ukoliko koristi “brute force” pristup moći će relativno lako do njega da dođe.  
  
Jedan od veb sajtova koji koristi ovaj pristup (uz najverovatnije dodatnu zaštitu) je linkedIn što se može videtu sa slike 5.2.1.



Slika 5.2.1

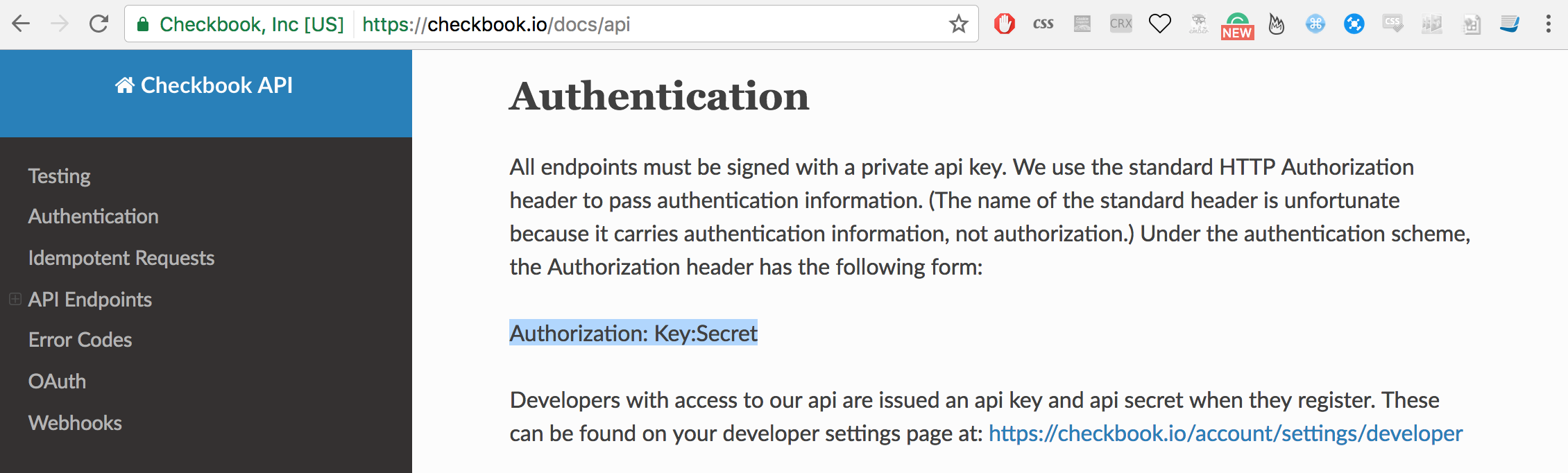
## 5.3 Podaci na serveru i ključ kao GET parametar

Pristup sličan pristupu iz 5.2 sa time što se umesto kao deo Cookie-a ključ sesije šalje kao deo get parametra. Ovim se pojavljuje još jedan problem u tome što neko ko posmatra korisnika kako radi na kompjuteru može da vidi sam ključ sesije.

## 5.4 Podaci na serveru i ključ kao deo zaglavlja zahteva

Slanje ključa kao deo potpisa zahteva je čest i omogućava čuvanje podataka po tom ključu na serverskoj strani. Ovaj pristup se koristi za implementaciju mnogih protokola autentifikacije.

Na slici 5.4.1 možete videti dokumentaciju jednog programskog interfejsa uspešne firme koja koristi ovaj pristup.



Slika 5.4.1

# 6. Protokoli autentifikacije veb aplikacija

Protokoli autentifikacije će u ovom poglavlju biti sagledani najviše u kontekstu autentifikacije korisnika ne samo na nivou veb klijenta već i na nivou autentifikacije korisnika između veb aplikacija i sistema koji ne rade strogo u okruženju veb pregledača. Ovakvi sistemi mogu biti mobilni uređaji kao i udaljeni serveri koji se integrišu sa postojećim aplikacijama. U kontekstu ovog poglavlja biće implementirani neki od ovih protokola sa strane korisnika kao i sa strane provajdera.

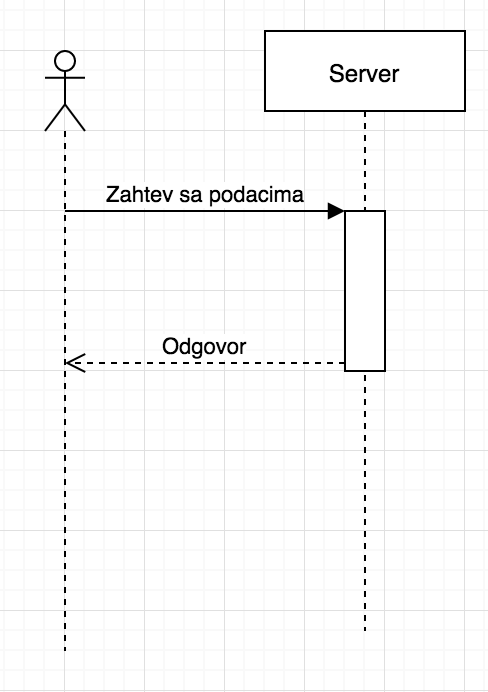
## 6.1 Protokol autentifikacije ime i šifra

Prilikom slanja zahteva zahteva se šalju i osnovni podaci korisnika, njegovo ime i password.

Ovakav protokol autentifikacije zahteva da onaj ko pušta zahteve u svakom slanju šalje korisnikove podatke i da ih čuva na svojoj strani. Pored ovoga parametri zahteva na http protokolu nisu enkriptovani te je moguće uraditi “man in the middle” napad.

Mnogi stari sistemi zahtevaju ovakav način autentifikacije koji je zbog prethodno navedenih razloga ne preporučljiv.

Primer sekvencijalnog dijagrama ovog protokola može se videti na slici 6.1.1

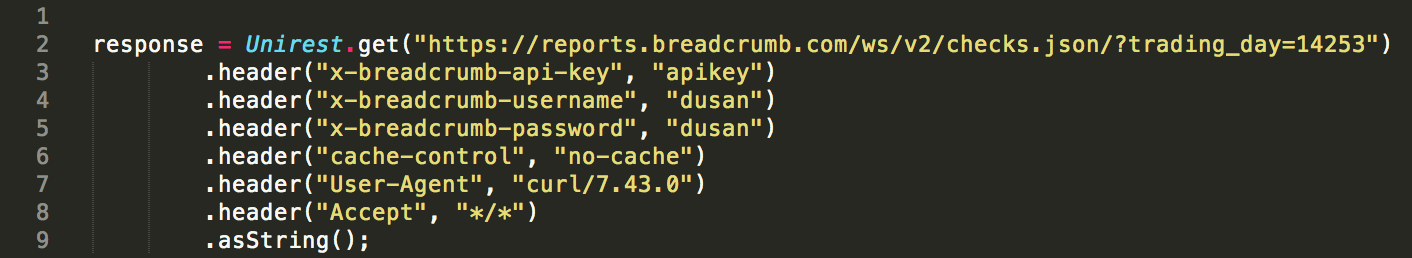


Slika 6.1.1

Osnovna dva entiteta u ovoj komunikaciji su klijent i server. Klijent pravi zahteve potpisane podacima i šalje ih ka serveru.

Jedan od sistema koji koriste ovakav pristup je Breadcrumb point of sale sistem koji funkcioniše na nivou restorana.

Sa slike 6.1.2 možemo videti primer koda koji se konektuje na Breadcrumb programski intefejs korišćenjem GET zahteva koji je potpisan username-om i password-om korisnika.

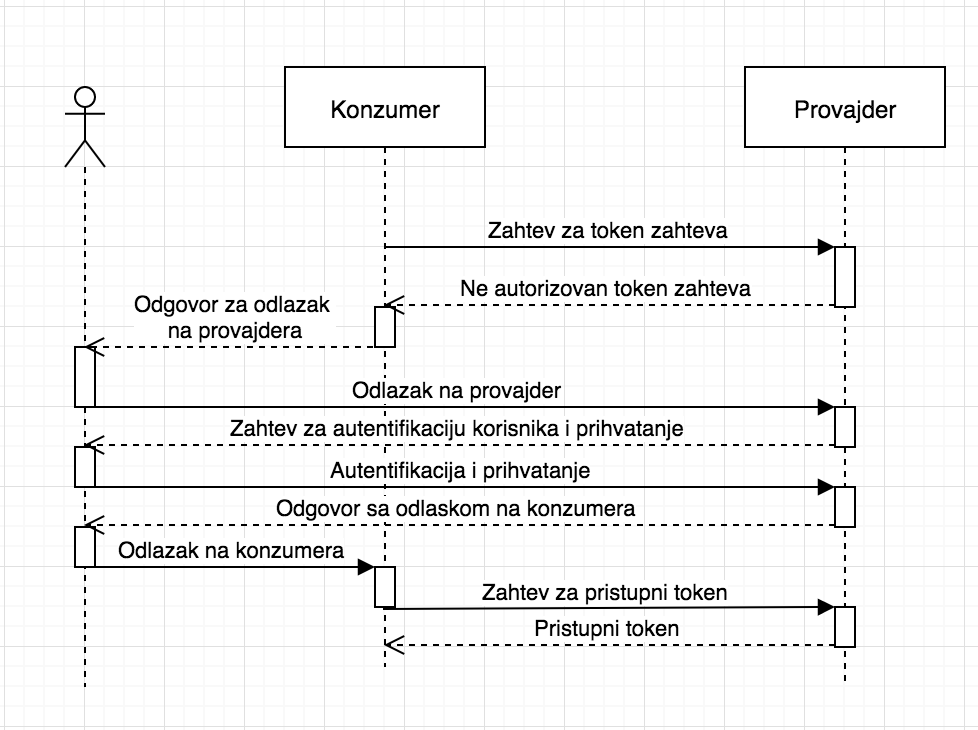


Slika 6.1.2

## 6.2 Protokol autentifikacije oAuth

Protokol oAuth je jedan od poznatijih protokola koji omogućavaju korisnicima da dozvole jednom sistemu koji koriste da pristupi drugom sistemu koji koriste u njihovo ime. Ovaj protokol se sastoji iz tri osnovna aktera: korisnika, konzumera i provajdera.

Primer sekvencijalnog dijagrama koji predstavlja implementaciju oAuth protokola je na slici 6.2.1.



Slika 6.2.1

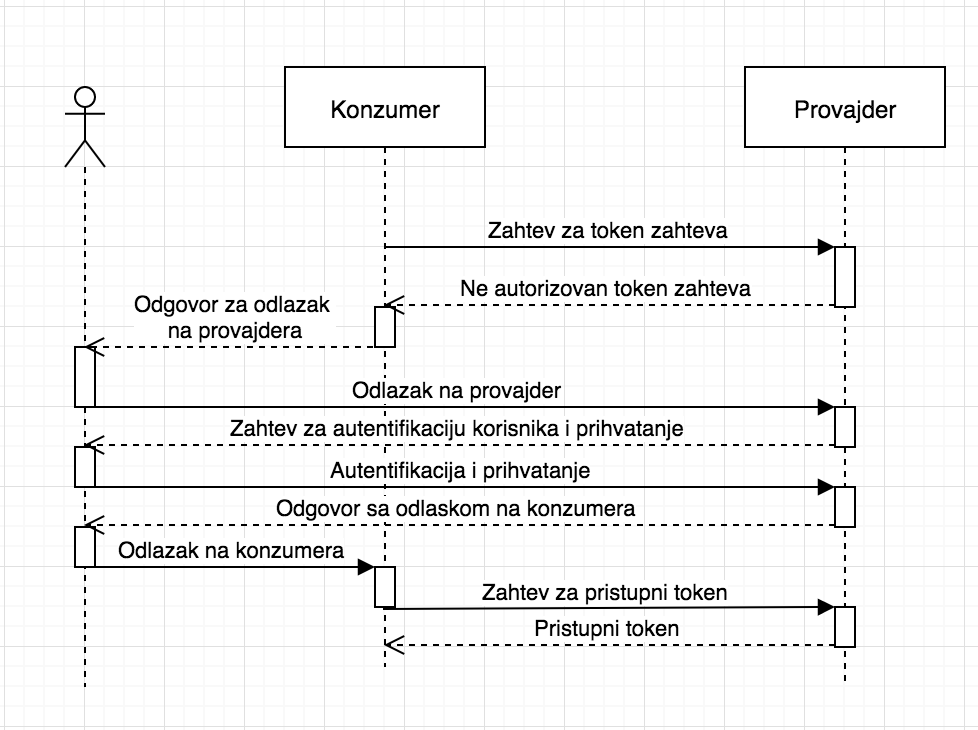
Interakcija oAuth korisnika:  
Konzumer šalje zahtev provajderu za token zahteva koji prosleđuje korisniku. Korisnik sa ovim tokenom odlazi kod provajdera i vrši autentifikaciju i odobravanje na njemu, nakon toga korisnik dobija odgovor od provajdera da je sve uspelo i konzumer onda pravi novi zahtev ka provajderu sa zahtevnim tokenom i kao odgovor dobija pristupni token kojim potpisuje zahteve nakon toga.

## 6.3 Protokol autentifikacije oAuth2

Protokol oAuth2 je naredna verzija oAuth protokola koja ne donosi prava poboljšanja u vidu sigurnosti. oAuth2 protokol donosi mnoga poboljšanja na nivou aplikativne logike.

Najveće mane oAuth protokola su se ogledale u tome što uređaji na kojima je radio su se oslanjali na postojanje veb pretrazivača, korisnici su morali ručno da kopiraju token pristupa i mogao je da radi samo na sistemima koji su podržavali ekriptovanje.

Sa serverske strane (konzumer-a) problem se javljao na nivou rasta aplikacija sa povećanim brojem korisnika bilo je sve zahtevnije održavati i proveravati ekriptovanje bez bezbedonosne vrednosti.



Slika 6.3.1

## 6.4 Protokol autentifikacije Json Web Token (JWT)

Json Web Token (JWT) je standard koji definiše protokol za bezbedno slanje podataka između klijenta i servera na bezbedan način tako da je moguće proveriti validnost JWT-a.

JWT koristi pristup sličan onome koji smo prethodno opisali u poglavlju 5.1 ali ne koristi cookie-je već je agnostičan u pogledu kako JWT biva dostavljen serveru.

Za enkripciju sa JWT-em je moguće koristiti HMAC algoritam ili public i private ključ sa RSA algoritmom.

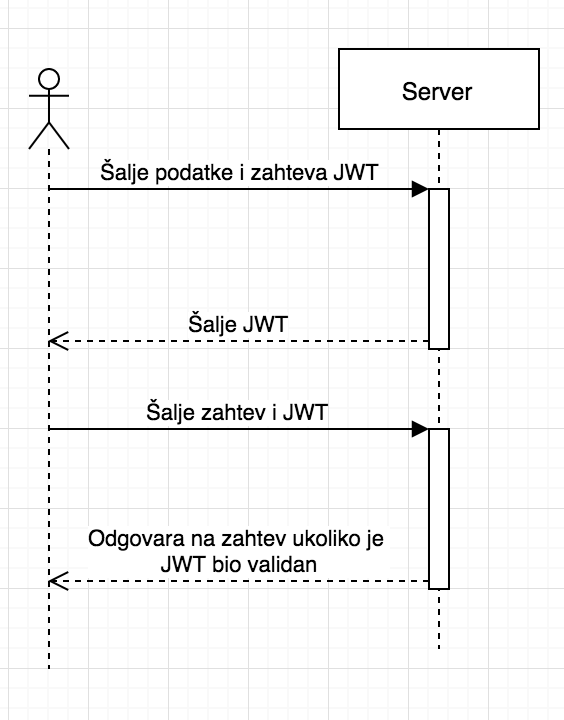
Prednosti JWT-a:

Moguće ga je slati kroz URL, kroz parametar zahteva ili kao deo zaglavlja zahteva.

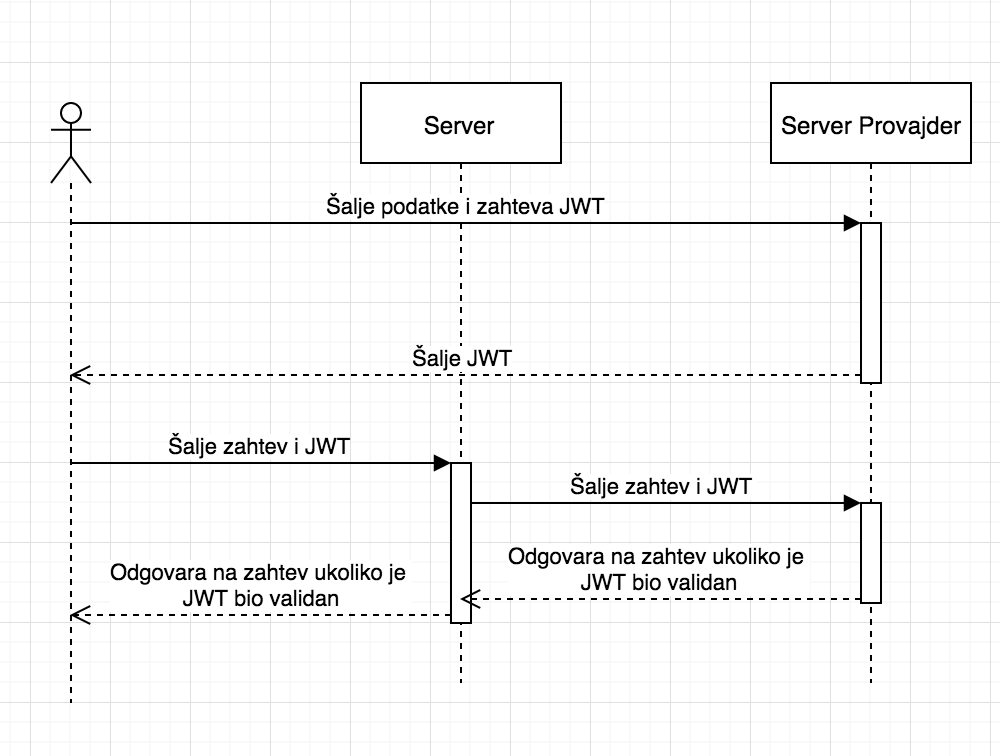
Može da sadrži podatke o sesiji korisnika.

JWT nije i ne definiše enkripciju već samo enkoding podataka. Ovaj protokol definiše samo kako poslati podatke i verifikovati njihov izvor.

Na slici 6.4.1 možemo videti sekvencijalni dijagram korišćenja JWT-a za autentifikaciju a na slici 6.4.2 možemo videti korišćenje JWT-a za prosleđivanje kredencijala pristupa udaljenim aplikacijama.



Slika 6.4.1



Slika 6.4.2

JWT se sastoji iz 3 osnovna dela, zaglvalja tela i verifikacije. Ova tri dela su odvojena tačkom. Zaglavlje zahteva sadrži algoritam korišćen za enkripciju verifikacije i tip token-a (u ovom slučaju JWT). Zaglavlje biva enkodovano u base64 format pre nego što postane deo JWT-a. Telo samog JWT-a je fleksibilno i predstavlja JSON podatke koji bivaju poslati, takođe je enkodovano u base64 format. Za generisanje poslednjeg dela JWT-a bitno je da uzmemo zaglavlje i telo samog JWT-a da ih enkodujemo u base64 i konkateniramo tačkom zatim ovaj podatak dodajemo samom JWT-a.

Prvi deo JWT-a je base64 enkodovano zaglavlje

Drugi deo JWT-a je base64 tela (sadržaj)

Treći deo je hash kombinacije tačkom tela i zaglavlja

Primer JWT-a:  
eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZCI6MSwibmFtZSI6IkR1c2FuIiwicGFzc3dvcmQiOiJTZUN1cjNQYXNzMSJ9.5PT\_X8quTH8fxW3DrB0vHWgCY1jzF0m-bhog7IWlOfU=

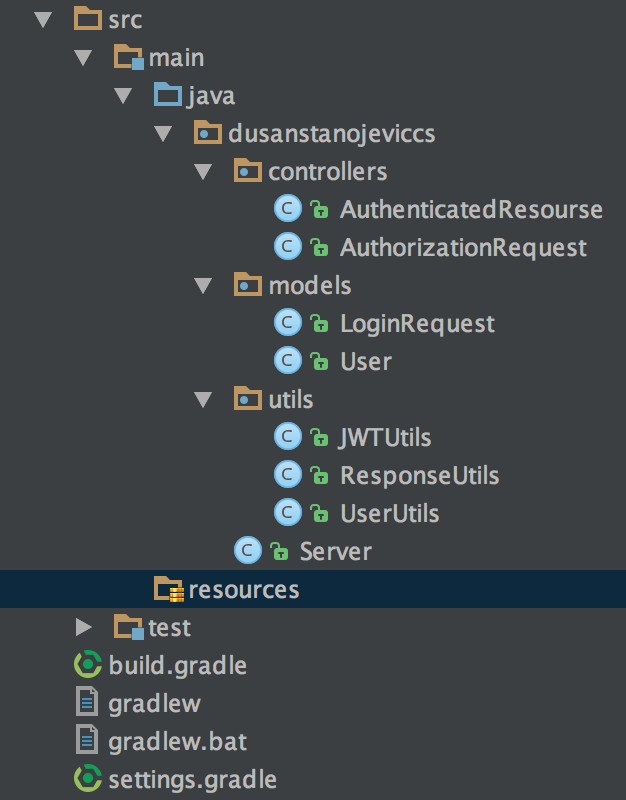
# 7. Implementacija protokola JWT za autentifikacije veb aplikacija

Za implementaciju JWT protokola koristićemo Java programski jezik i biblioteku niskog nivo-a Vert.x.

## 7.1 Implementacija JWT-a

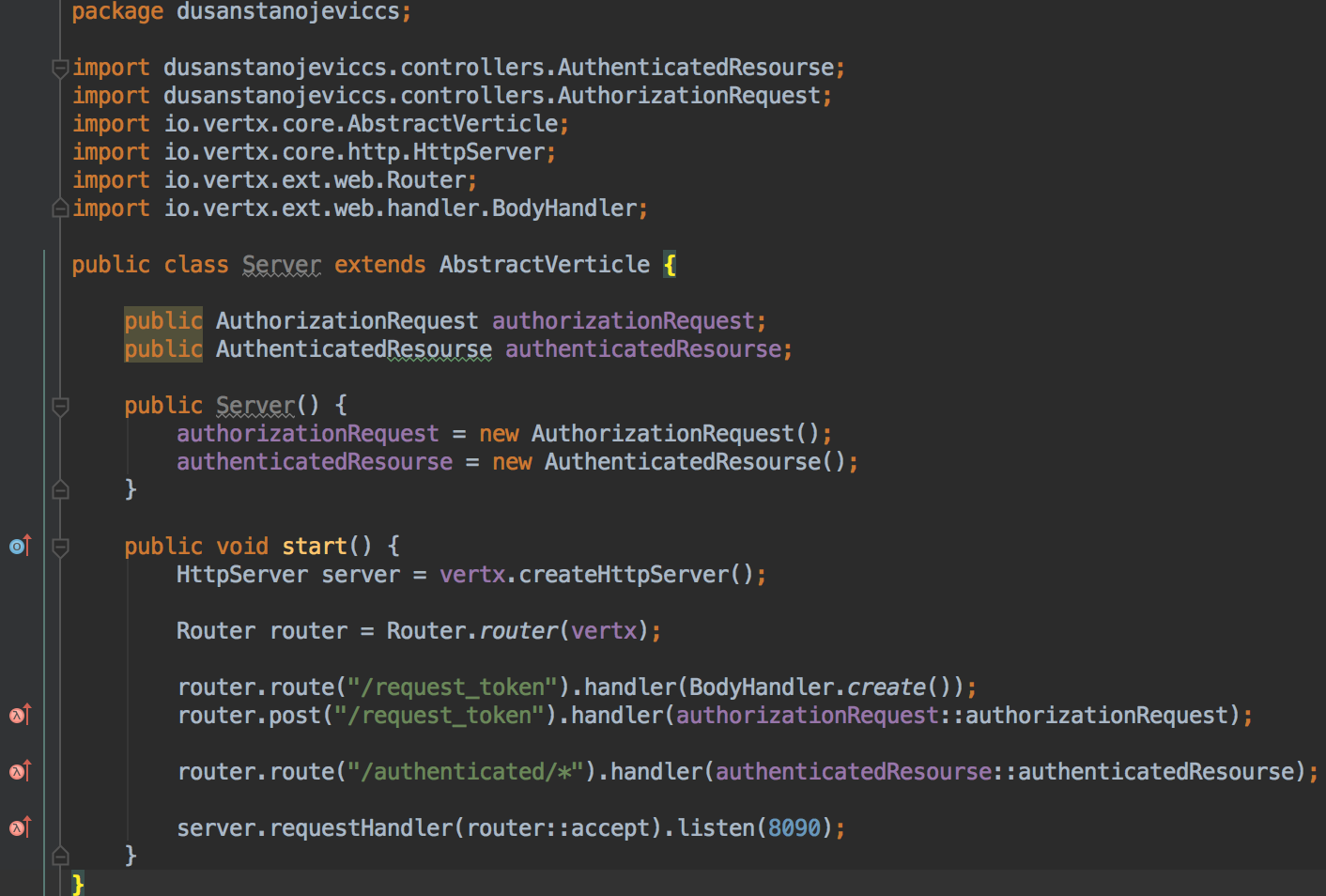
Za izradu koda korišćeni su i Gradle kao alat za dobavljanje biblioteka, git kao alat za kontrolu verzija koda i InteliJ kao IDE za razvoj aplikacije.

Struktura projekta je data na slici 7.1



Slika 7.1

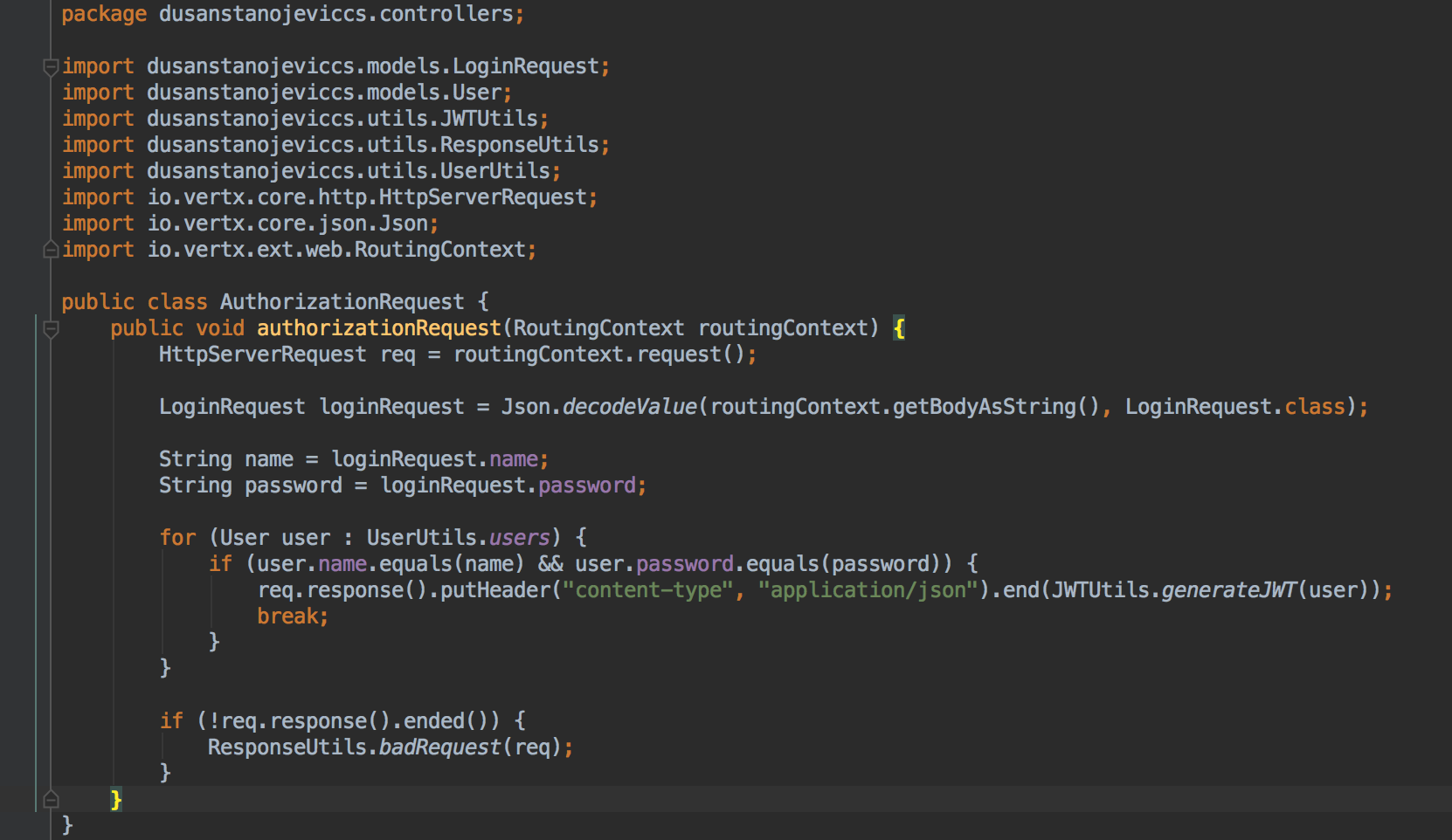
Za osnovno generisanje interfejsa ka klijentima preko http protokola definisali smo klasu Server koja je zadužena da kreira server koji sluša na određenom portu. Kod klase Server može se videti na slici 7.2. Ova klasa otvara 2 rute request\_token i authenticated na portu 8090. Ruta request\_token reaguje na post tip zahteva i prosleđuje obradu zahteva AuthorizationRequest klasi dok ruta authenticated prosleđuje zahteve AuthenticatedResourse metodi.



Slika 7.2

Klasa AuthorizationRequest je zadužena da proveri da li korisnik sa određenim kredencijalima postoji i da uzvrati generisani JWT. Pošto je ova klasa deo demo projekta ona vrši iteraciju predefinisanih korisnika i ukoliko odgovarajućeg korisnika nađe uzvraća njegov JWT.

Kod ove klase možemo videti na slici 7.3.



Slika 7.3

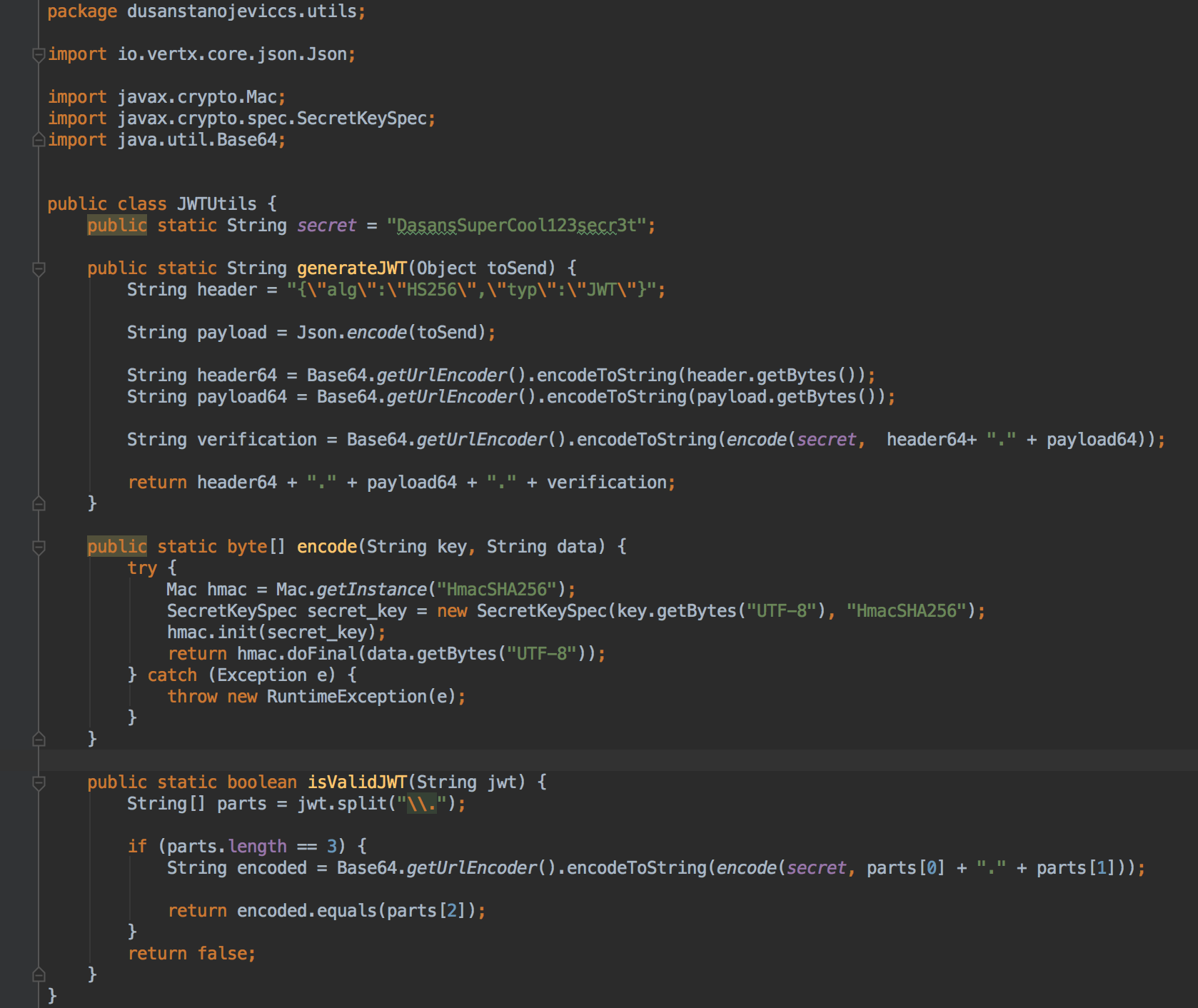
Klasa AuthorizationRequest prosleđuje kontrolu klasi JWTUtils za generisanje JWT token-a. Kod za generisanje JWT token-a će biti analiziran u daljem tekstu.

Kada korisnik želi da pristupi resorsima za koje je potrebna autentifikacija on pravi zahtev i kao header zahteva postavlja autorizacioni parametar. Ovaj zahtev se obrađuje u klasi AuthenticatedResourse koja bi u pravom svetu predstavljala zaštićeni endpoint. Kod ove klase možemo videti na slici 7.4.



Slika 7.4

Klasa sa slike 7.4 prosleđuje kontrolu JWTUtils klasi koja je data na slici 7.5 a u daljem tekstu i detaljno analizirana.



Slika 7.5

Klasa JWTUtils je zadužena za generisanje naših JWT-a i njihovu validaciju, ova klasa u sebi sadrži privatnu tajnu koja se nikad ne šalje klijentima. Metoda generateJWT uzima u sebe objekat koji predstavlja payload koji treba da bude poslat korisnicima, ova metoda ima header napisan u vidu statičkog string-a jer pretpostavlja da se algoritam HmacSHA256 neće menjati. Po pravilima iz poglavlja 6.4 ova metoda generiše JWT za klijenta.

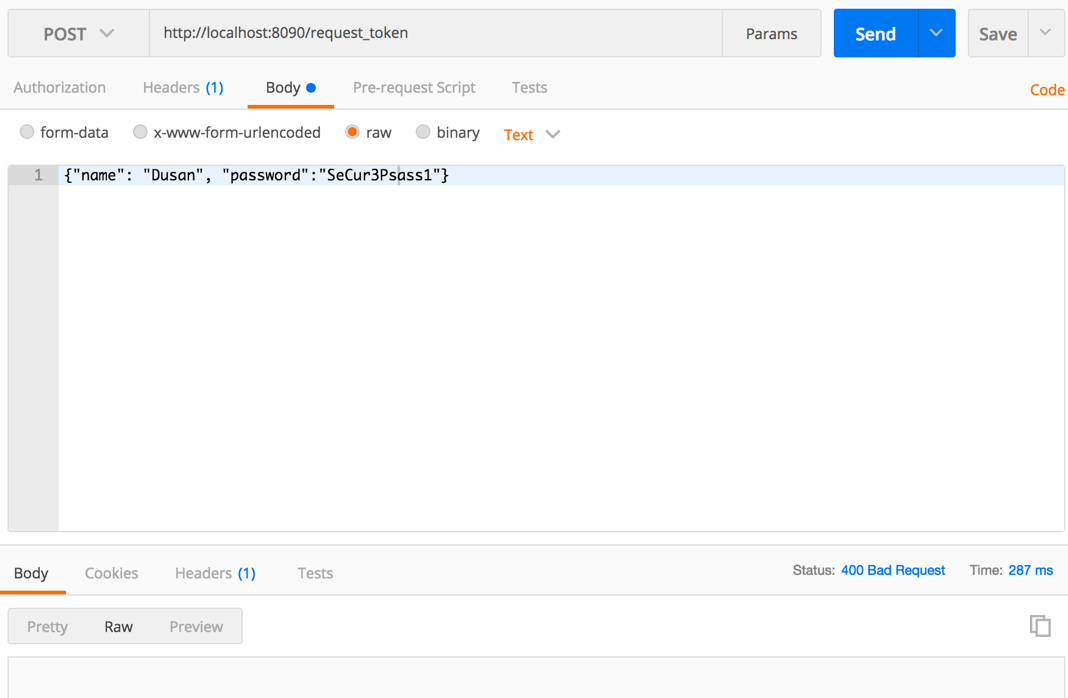
Metoda isValidJWT klase JWTUtils je zadužena da proveri validnost jwt-a dobijenog od klijent-a. Ova metoda deli JWT na 3 dela i zatim proverava da li je enkripcija kombinacije prva dva jednaka trećem, ukoliko jeste JWT je validan na datom serveru.

Pored ovih korišćeno je još dve klase za predstavljanje model-a User i LoginRequest kao i dve pomoćne klase ResponseUtils i UserUtils.

## 7.2 Testiranje implementacije JWT-a

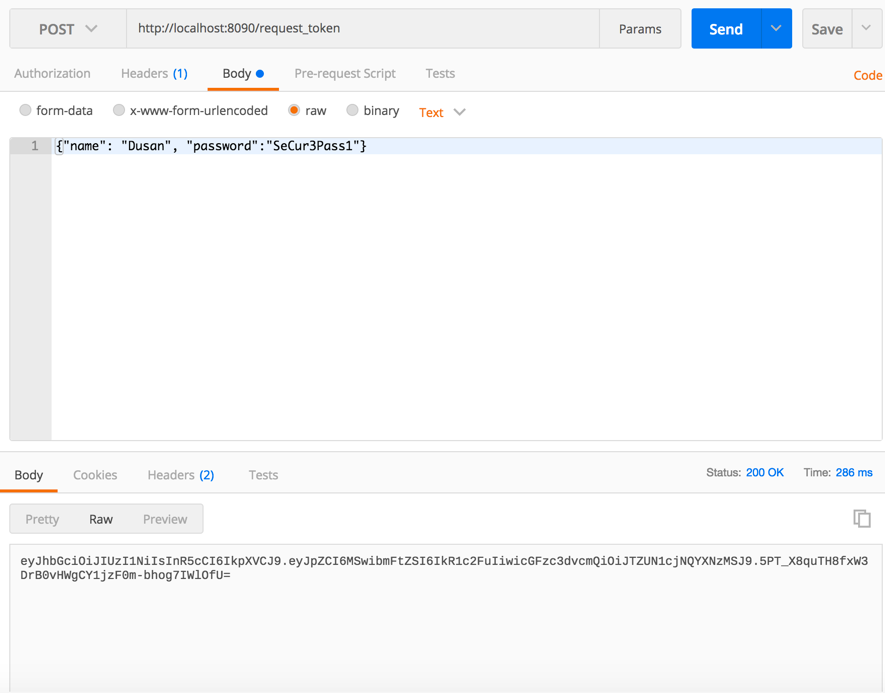
Implementaciju i funkcionalan rad priloženog koda testirali smo korišćenjem postman alata koji nam omogućava da generišemo http zahteve.

Test korišćenja ne validnih kredencijala možemo da vidimo sa slike 7.1.1. Server je postupio korektno i vratio status 400, što se vidi u donjem desnom uglu slike.



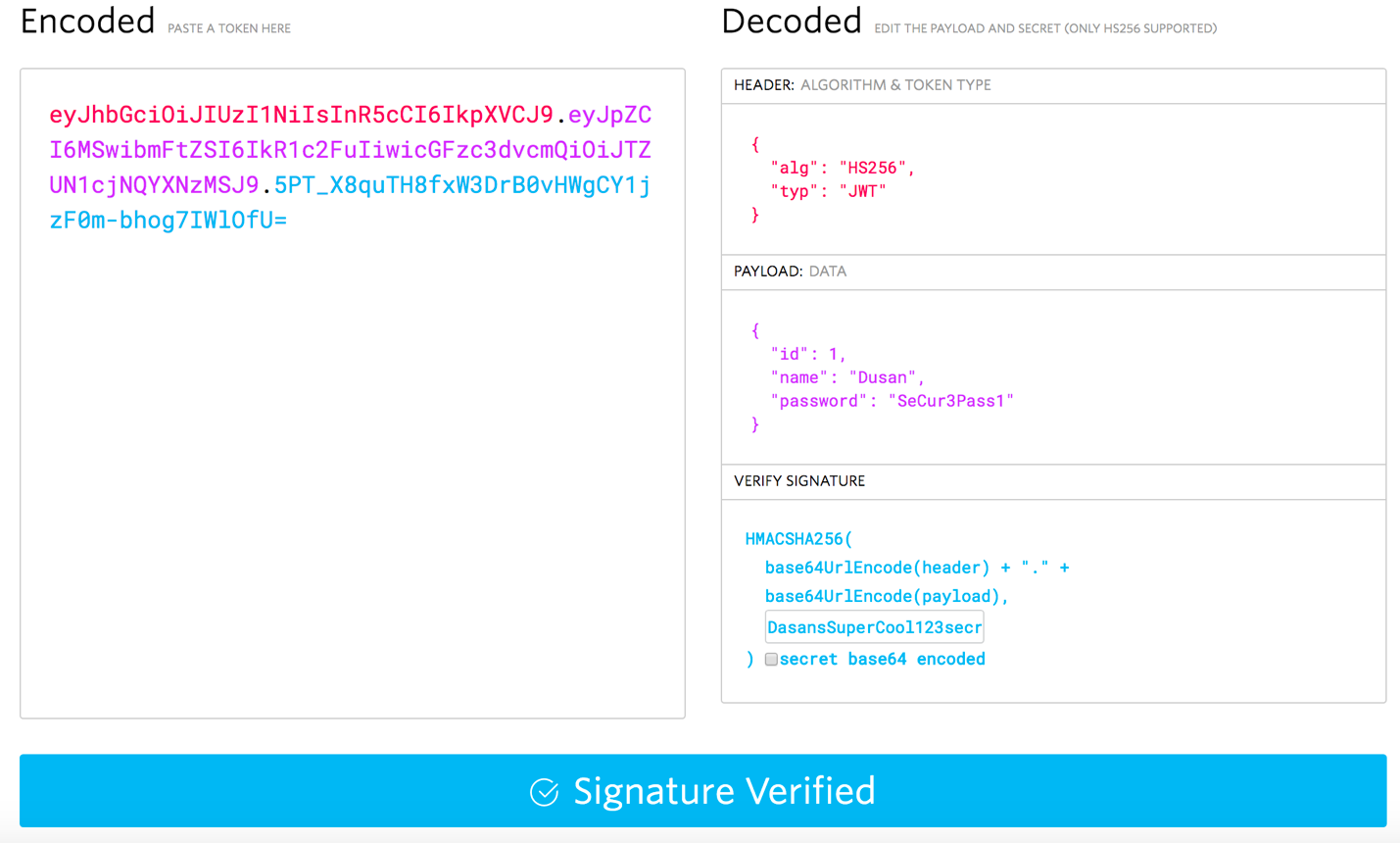
Slika 7.1.1

Test uspešne autentifikacije dat je na slici 7.1.2



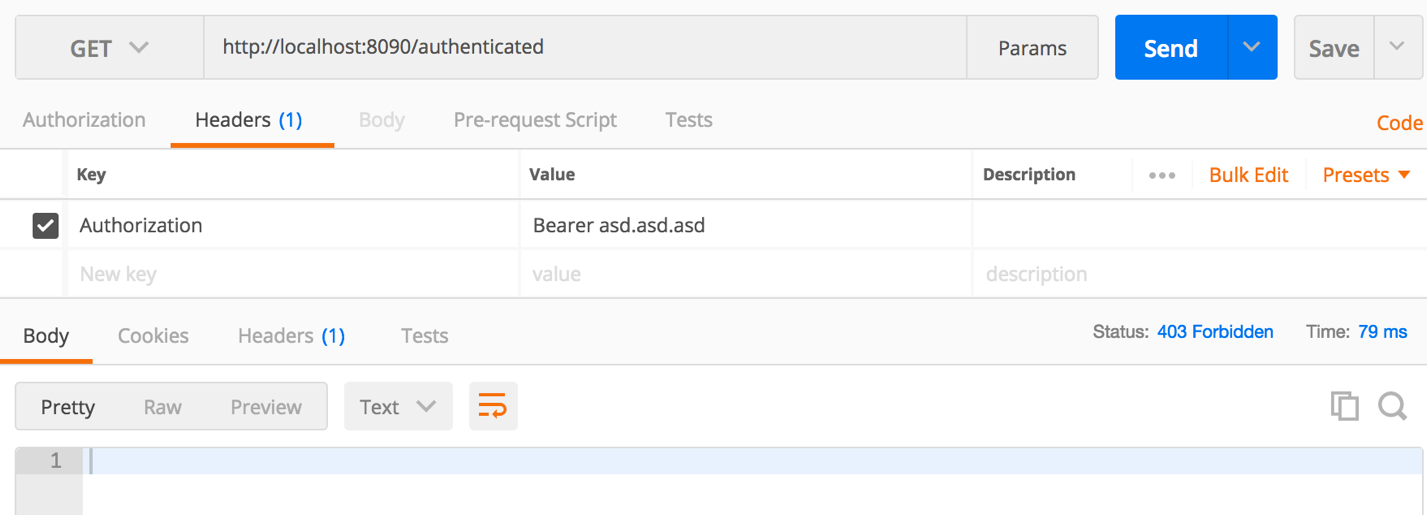
Slika 7.1.2

Kako bismo testirali uspešnost generisanja JWT-a kopirali smo ga i nalepili na jwt.io delu za testiranje JWT-a. Uspešnost ovoga možemo videti na slici 7.1.3.

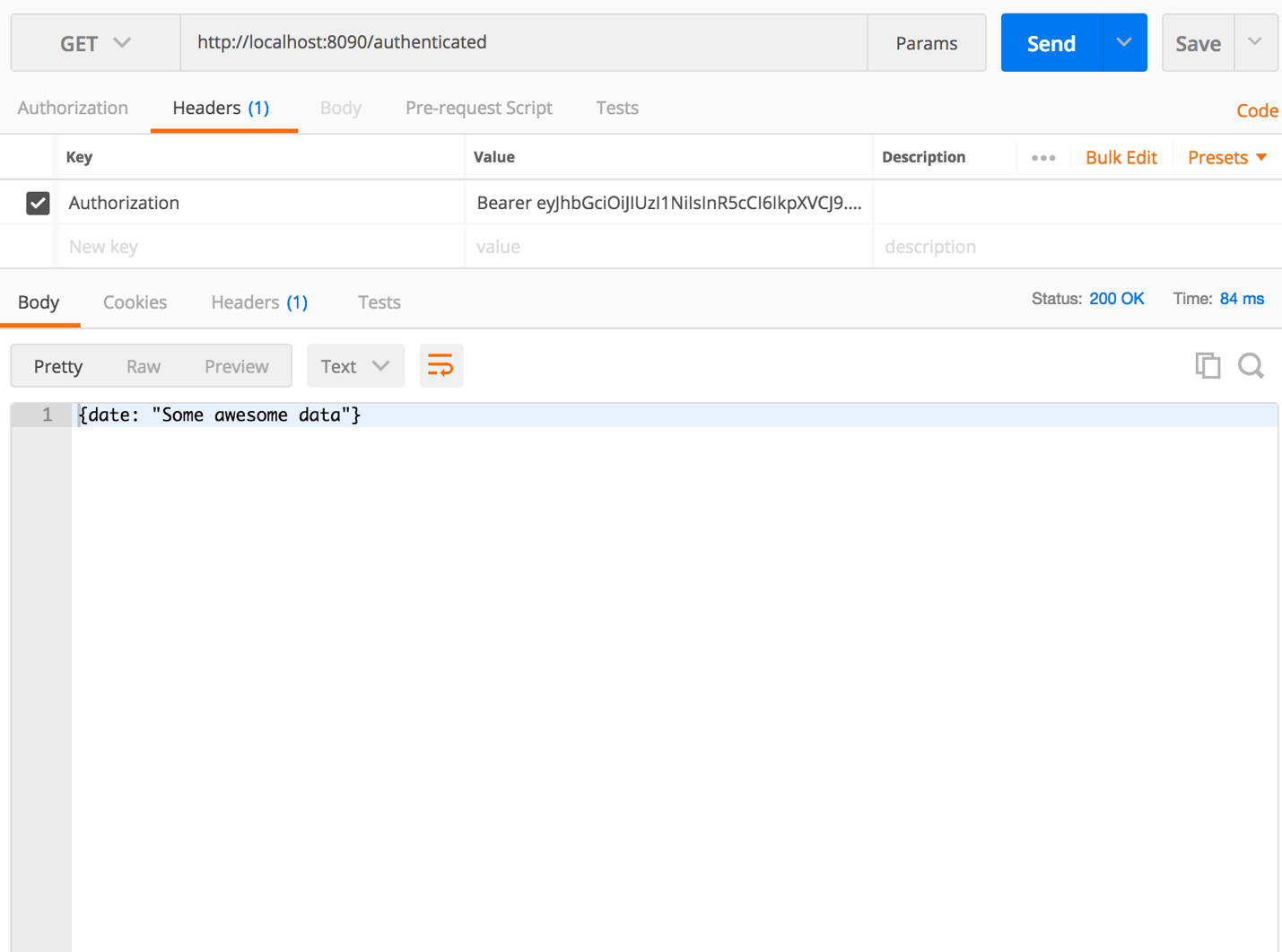


Slika 7.1.3

Nakon testiranja generisanog tokena trebamo da testiramo samu validaciju JWT-a na serveru. Neuspešan primer ovoga možemo videtu na slici 7.1.4 a uspešan na slici 7.1.5.



Slika 7.1.4



Slika 7.1.5

# 5. Literatura

[1] JWT - jwt.io

[2] Introduction to oauth - https://blog.varonis.com/introduction-to-oauth/

[3] Understanding oauth2 - http://www.bubblecode.net/en/2016/01/22/understanding-oauth2/

[4] 5 Easy steps to understand json web tokens jwt - https://medium.com/vandium-software/5-easy-steps-to-understanding-json-web-tokens-jwt-1164c0adfcec

[5] Vert.x - http://vertx.io/

[7] The Web Application - Hacker’s Handbook Second Edition - Dafydd Stuttard, Marcus Pinto