# Zadatak

* Verzionirati API sa prošlih vježbi, po želji koristiti jedan način verzioniranja za dohvaćanje tokena, a neki drugi za dohvat podataka iz baze podataka
* Koristiti swagger za kreiranje dokumentacije
* Napisati testove koji će testirati funkcionalnost API-a po želji

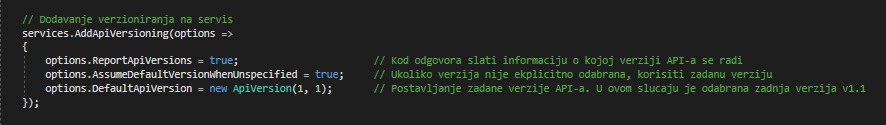
Vježbu poslati na email: [isinek@gmail.com](mailto:isinek@gmail.com)  
Ogledni projekt: <https://github.com/isinek/napredne_tehnike_programiranja_web_servisa>   
Rok za predaju: 31.01. u 00:00

# Verzioniranje API-a

Ukoliko se radi o javno dostupnom API-u koji će koristiti mnogo korisnika, a planiraju se veće izmjene u funkcionalnosti, dohvatu podataka ili naziva metoda, verzioniranje je gotovo obavezno. U protivnom će svi korisnici u trenutku izmjene API-a morati doraditi svoje klijente ili oni neće biti funkcionalni.

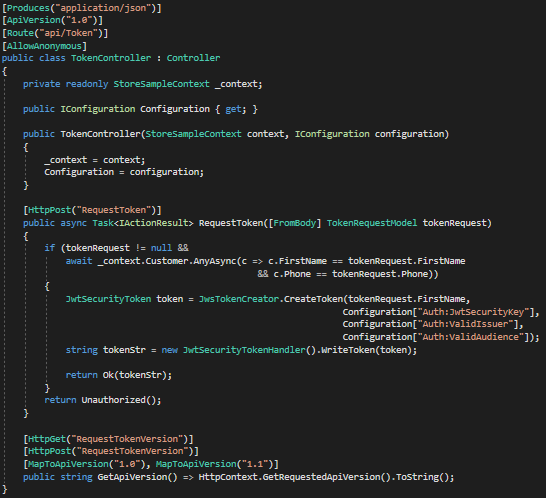
Kako bi se u projektu koristilo verzioniranje, u API projekt potrebno je dodati Microsoft.AspNetCore.Mvc.Versioning NuGet paket. Zatim u Startup.cs datoteku treba dodati verzioniranje.

Startup.cs

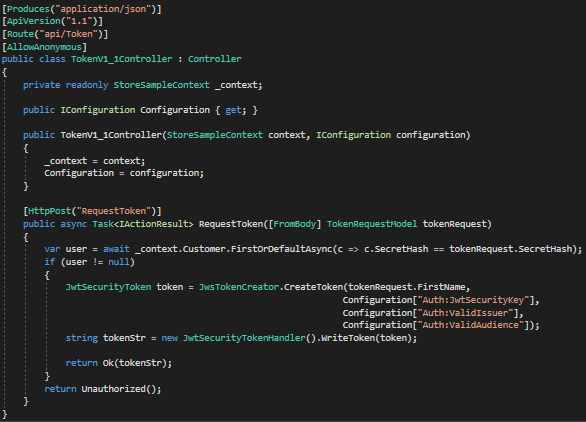


Na kontrolerima se koriste ApiVersion atributi za određivanje verzije. Ukoliko se kontroler treba moći koristiti kroz više verzija, moguće je dodati više verzija odjednom ili ApiVersionNeutral atribut kako bi se koristio u svim verzijama.

Ukoliko se jedna metoda unutar kontrolera koristi u više verzija, moguće ju je mapirati pomoću MapToApiVersion atributa.



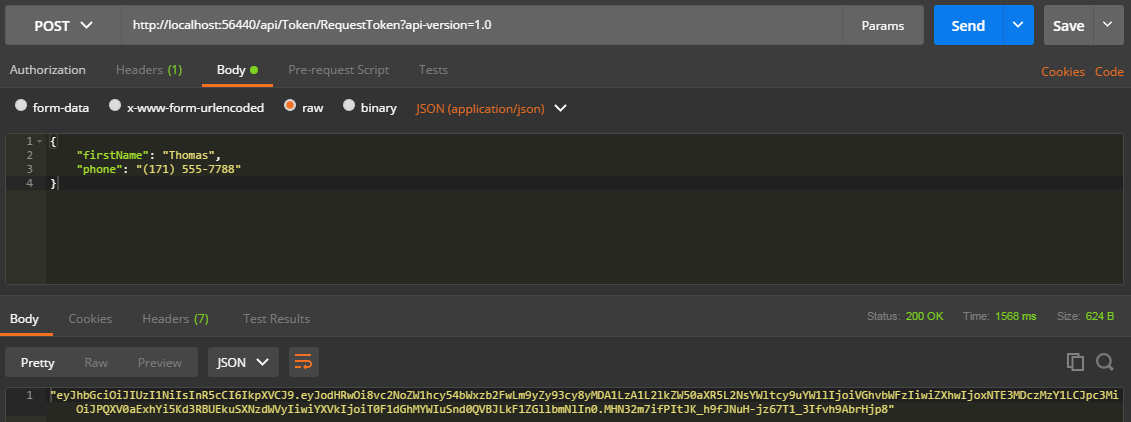
Slika 1: Token API v1.0



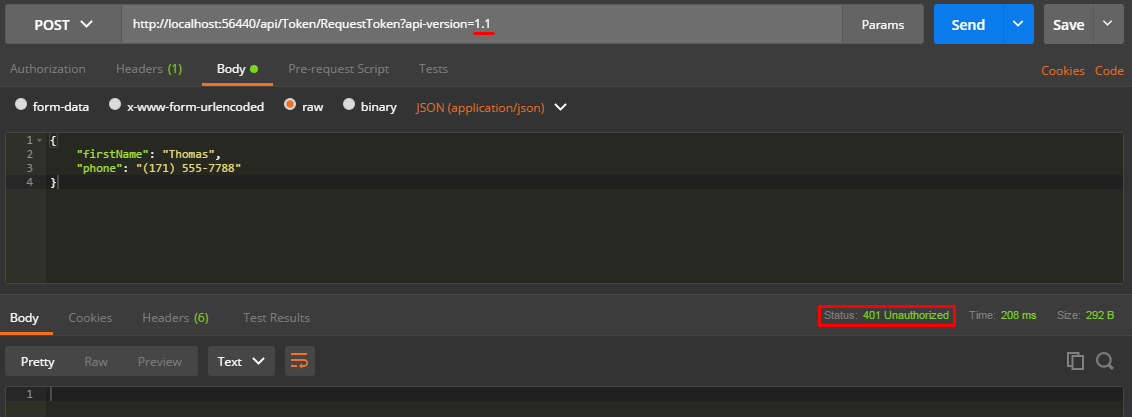
Slika 2: Token API V1.1

## Verzioniranje pomoću api-version parametra

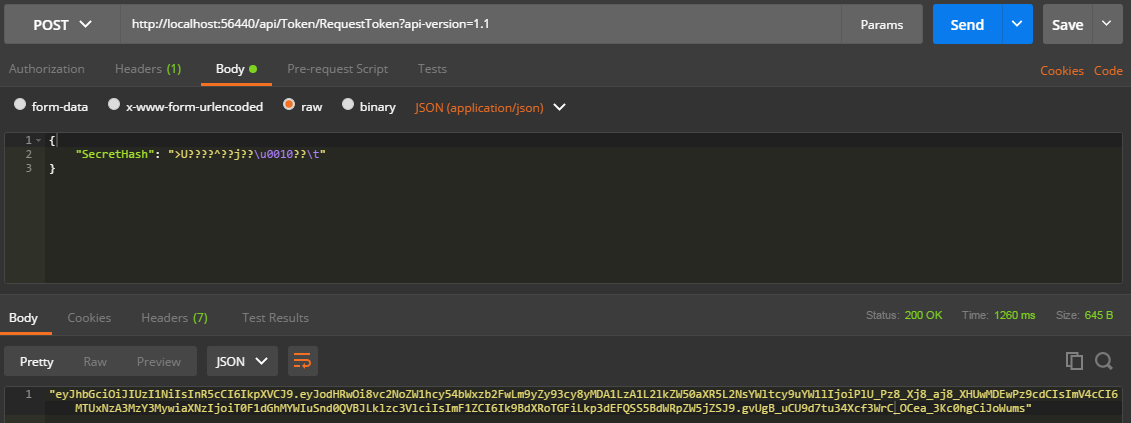
Dodavanjem verzioniranja omogućeno je korištenje api-version URL parametra. Primjeri poziva različitih verzija RequestToken metode:



Slika 3: Dohvat tokena preko stare verzije API-a



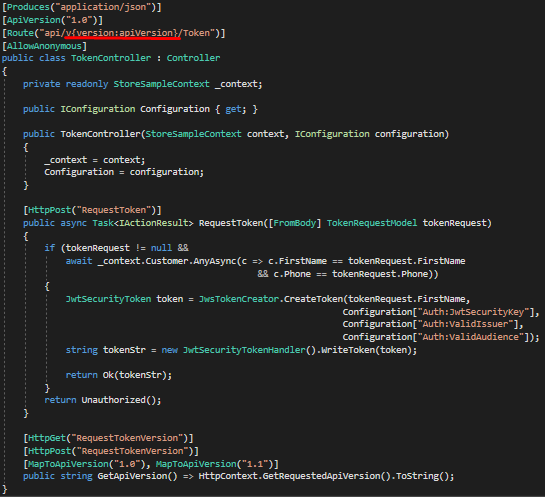
Slika 4: Verzioniranje API-a preko parametra v1.0



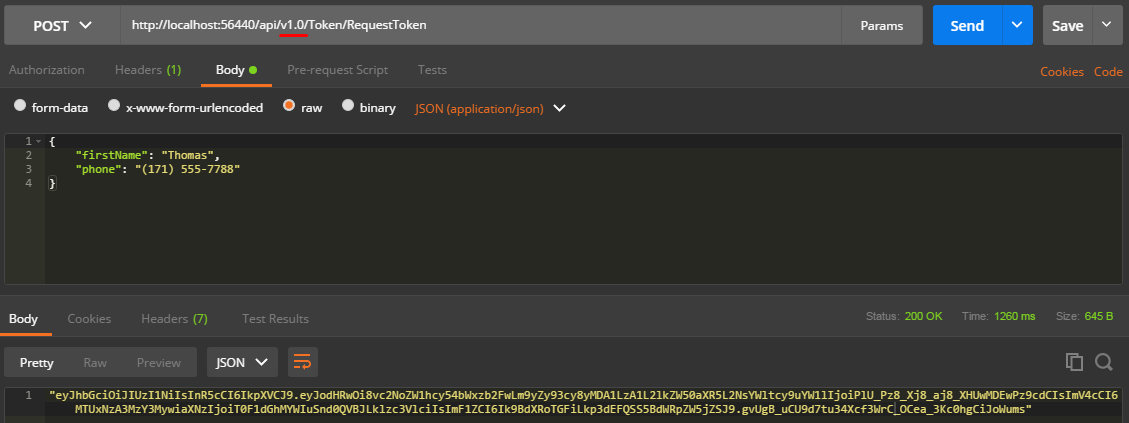
Slika 5: Verzioniranje API-a preko parametra v1.1

## Verzioniranje pomoću URL ruta

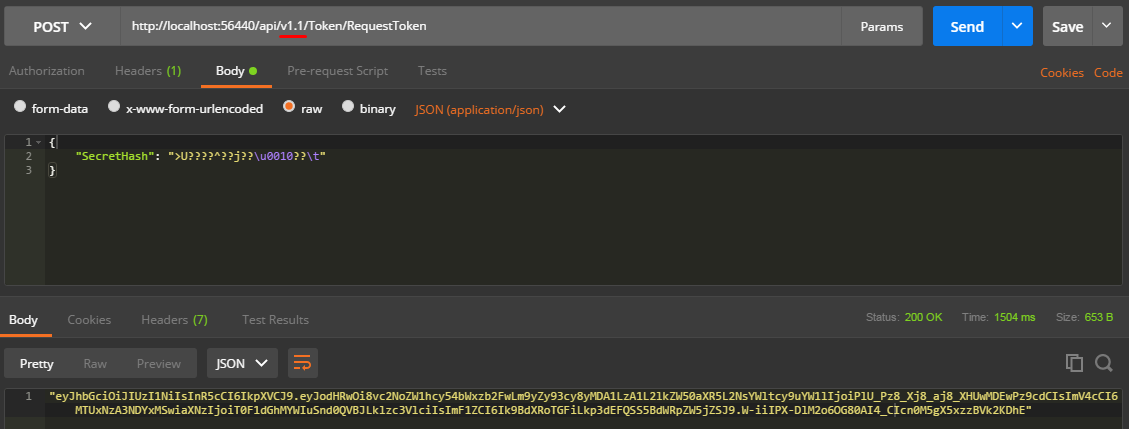
S obzirom na to da URL parametri nisu najbolje rješenje za verzioniranje, naročito kada su u pitanju pozivi POST, PUT i DELETE metoda, verzioniranje je moguće napraviti tako da se verzija API-a izvlači iz URL-a koji je pozvan.



Slika 6: Primjer definiranja rute za verzioniranje



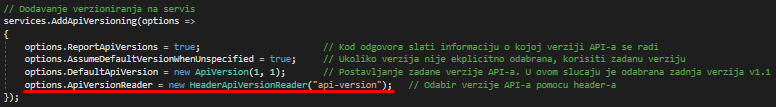
Slika 7: Verzioniranje API-a preko ruta v1.0



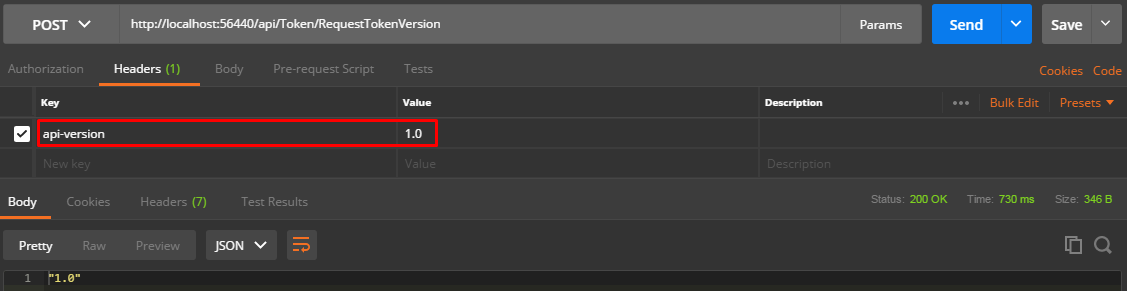
Slika 8: Verzioniranje API-a preko ruta v1.1

## Verzioniranje pomoću headera u pozivu

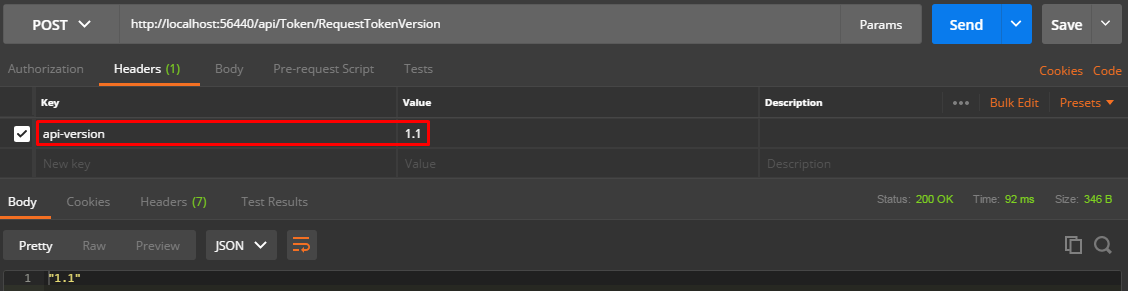
Za verzioniranje pomoću headera potrebno je to dodatno naznačiti u Startup.cs datoteci.



Slika 9: Dodavanje opcije odabira verzije API-a unutar headera



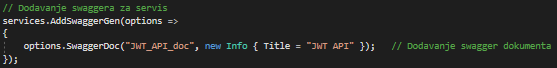
Slika 10: Dohvat verzije API-a pomoću headera v1.0



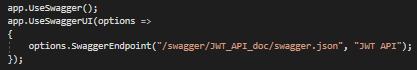
Slika 11: Dohvat verzije API-a pomoću headera v1.1

# Dokumentiranje pomoću swaggera

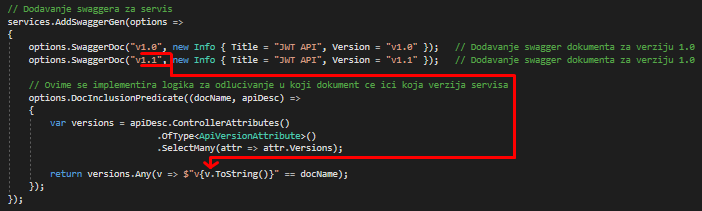
Da bi se kod mogao dokumentirati pomoću swagger frameworka u projekt je potrebno dodati Swashbuckle.AspNetCore NuGet paket. Nakon instalacije potrebno je definirati korištenje swaggera u Startup.cs datoteci.



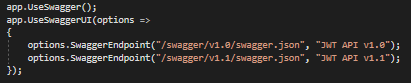
Slika 12: Generiranje swagger dokumenta



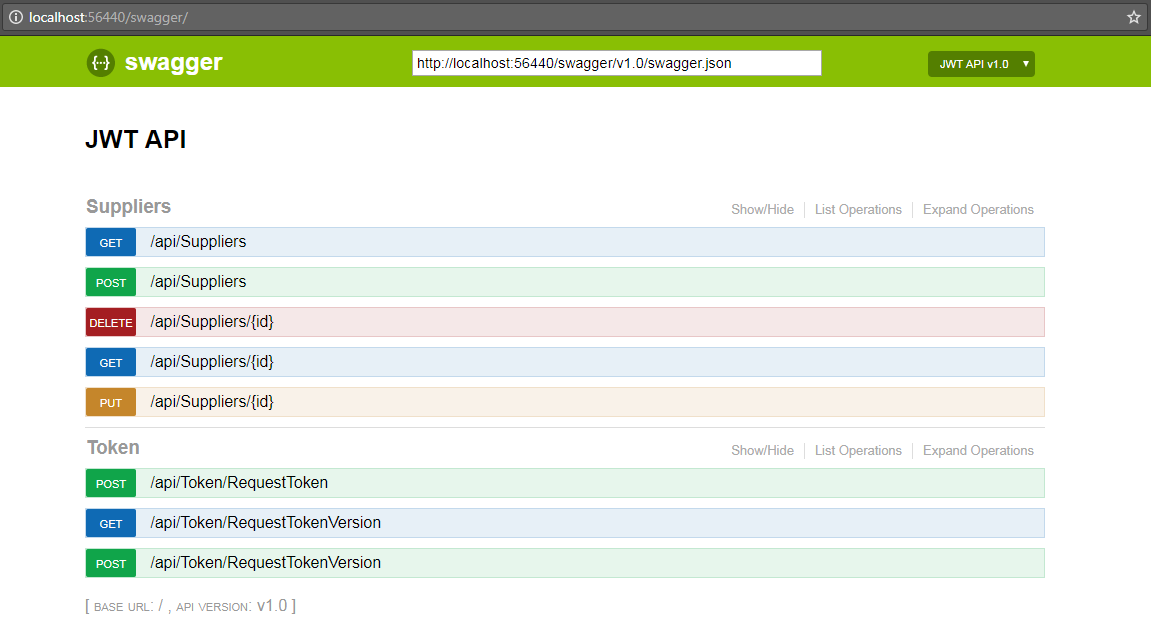
Slika 13: Dodijeljivanje URL-a swagger dokumentu



Slika 14: Generiranje swagger dokumenata i povezivanje sa verzijom servisa

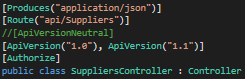


Slika 15: Dodijeljivanje URL-ova swagger dokumentima



Slika 16: Swagger UI

Kako bi swagger prepoznao SuppliersController u obje verzije API-a, potrebno je na SuppliersController klasi definirati sve API verzije.



Slika 17: Dodavanje API verzija za swagger

# Testiranje

Testiranje koda bi trebala biti obaveza svakog developera kako bi se osigurao da izmjenom ili dodavanjem koda nije narušena ili izmijenjena funkcionalnost koda. Testirati možemo funkcionalnosti pojedinih metoda unutar klasa, ali i servisa.

Za testiranje .NET Core servisa može se koristiti xUnit Test Project. Klase testova prema konvenciji završavaju riječi Should (npr. Customer\_Should i Suppliers\_Should). Metode označene Fact atributom su zapravo testovi koji se mogu izvršavati unutar projekta. Za validaciju podataka i objekata koristi se Assert klasa i njezine metode.

Pošto se za testiranje nikad ne bi trebala koristiti stvarna baza podataka, u ovom primjeru kotištena je InMemory baza podataka. Primjer korištenja InMemory baze podataka i kreiranje testnog contexta može se vidjeti unutar klase SuppliersService\_Should.