

Sigurnost operacijskih sustava i aplikacija

Laboratorijska vježba

Sveučilište u Zagrebu, Fakultet elektrotehnike i računarstva

UPOTREBA OpenID Connect ZA JEDINSTVENU PRIJAVU

MATIJA ALOJZ STUHNE

Zagreb, svibanj 2025.

Uvod

U suvremenim informacijskim sustavima krajnji korisnici često pristupaju većem broju aplikacija unutar iste organizacije. Upravljanje višestrukim korisničkim računima nije samo nepraktično, već i predstavlja sigurnosni rizik. Jedinstvena prijava (Single Sign-On – SSO) omogućuje korisnicima da se autentificiraju samo jednom i zatim pristupaju svim autoriziranim aplikacijama bez dodatne prijave. Ova laboratorijska vježba istražuje implementaciju SSO sustava korištenjem protokola OpenID Connect (OIDC), uz pomoć alata Keycloak, koji služi kao poslužitelj za upravljanje identitetima.

Zadatak

Zamišljena je organizacija koja razvija više web aplikacija i želi omogućiti korisnicima pristup svim aplikacijama korištenjem jedinstvene prijave. Potrebno je implementirati sustav koji koristi OpenID Connect kao protokol za autentifikaciju, pri čemu će Keycloak služiti kao centralni identitetski poslužitelj. U vježbi je potrebno razviti dvije jednostavne web aplikacije (npr. A i B), registrirati ih u sustavu Keycloak te omogućiti da korisnik, nakon prijave u jednu aplikaciju, automatski bude prijavljen i u drugu aplikaciju bez potrebe za ponovnom autentifikacijom.

Teorijski podloga

OpenID Connect (OIDC) predstavlja moderni identitetski protokol koji se temelji na protokolu OAuth 2.0 te omogućuje autentifikaciju korisnika i pouzdano prenošenje informacija o identitetu između različitih aplikacija. Dok OAuth 2.0 prvenstveno služi za autorizaciju pristupa resursima, OIDC proširuje tu funkcionalnost omogućujući identifikaciju korisnika, što ga čini prikladnim za implementaciju sustava za jedinstvenu prijavu (Single Sign-On – SSO).

OIDC uvodi nekoliko ključnih pojmova i mehanizama:

- **ID Token:** Struktura temeljena na JSON Web Token (JWT) formatu, koja sadrži skup atributa (tzv. „claims“) o autentificiranom korisniku. Primjeri atributa uključuju korisničko ime, e-mail adresu, vrijeme izdavanja tokena i njegov rok valjanosti. Ovaj token koristi se za dokazivanje identiteta korisnika pred klijent-aplikacijom.
- **UserInfo Endpoint:** Standardizirana HTTP točka putem koje klijent, nakon autentifikacije, može dohvatiti dodatne informacije o korisniku. Pristup ovoj točki obično zahtijeva prethodno izdani pristupni token (access token).
- **Authorization Code Flow:** Preporučeni tok autentifikacije za aplikacije koje se izvršavaju na poslužiteljskoj strani. Proces započinje preusmjeravanjem korisnika na identitetski poslužitelj gdje se odvija autentifikacija. Nakon toga klijent dobiva autorizacijski kod koji koristi za dohvat ID tokena i pristupnog tokena. Ovaj mehanizam omogućuje sigurno rukovanje osjetljivim podacima, budući da se razmjena tokena odvija između poslužitelja putem sigurnog kanala.

U kontekstu OpenID Connect protokola, sve aplikacije koje žele koristiti funkcionalnosti jedinstvene prijave moraju biti registrirane kod istog identitetskog poslužitelja, koji nadzire izdavanje i verifikaciju tokena. Ova centralizacija osigurava konzistentnost i sigurnost pri upravljanju korisničkim sesijama.

Keycloak je sustav otvorenog koda za upravljanje identitetima i pristupom (Identity and Access Management – IAM), koji pruža potpunu podršku za moderne standarde kao što su OpenID Connect, OAuth 2.0 i SAML 2.0. Osmišljen je za jednostavnu integraciju s aplikacijama te omogućuje upravljanje autentifikacijom, autorizacijom, korisničkim računima i ulogama.

Ključne funkcionalnosti koje Keycloak nudi uključuju:

- **Centralizirano upravljanje korisnicima:** Administratori imaju mogućnost upravljanja korisničkim računima, grupama, lozinkama i drugim atributima iz jednog sučelja.
- **Jedinstvena prijava (SSO):** Korisnik se autentificira samo jednom, nakon čega mu je omogućen pristup svim aplikacijama povezanim na isti identitetski sustav, bez potrebe za ponovnom prijavom.
- **Podrška za višestruke protokole:** Keycloak implementira više industrijskih standarda, čime omogućuje fleksibilnu integraciju s raznolikim sustavima i aplikacijama.
- **Mogućnosti upravljanja ulogama i pravima pristupa:** Aplikacije mogu delegirati provjeru uloga i pristupnih prava Keycloak-u, što pojednostavljuje logiku kontrole pristupa.

U sklopu jedinstvene prijave, sve aplikacije (npr. app-a, app-b) registrirane su unutar istog **realm-a** u Keycloak-u. Kada korisnik jednom uspješno obavi autentifikaciju putem Keycloak-a, isti token koji potvrđuje njegov identitet koristi se i za ostale aplikacije unutar sustava. Time se postiže korisničko iskustvo bez ponovljenih prijava, dok sigurnosne kontrole ostaju centralizirane.

Korištenje ovakvog modela omogućuje organizacijama sigurnu i efikasnu autentifikaciju korisnika, smanjuje složenost aplikacijskih kodova i poboljšava nadzor nad korisničkim sesijama i pristupom.

Postavke za vježbu

Okolina

- **Operacijski sustav:** Linux
 - Moguće je koristiti virtualni stroj s npr. Ubuntu distribucijom.
- **Docker:** Instaliran za pokretanje Keycloak poslužitelja.
 - Ako se koristi virtualni stroj, potrebno je Docker instalirati unutar njega.
- **Pristup Internetu:** Potreban za kloniranje Git repozitorija.
- **Napomena:** Ukoliko ne želite koristiti virtualni stroj, a na vašem računalu nije instaliran Linux operacijski sustav, potrebno je prilagoditi skriptu „./run_java_apps.sh“ u format prikladan za vaše okruženje (npr. PowerShell na Windowsima) kako biste pokrenuli obje Java aplikacije koje koriste konfiguracijske datoteke iz direktorija „./config“.
- **Napomena:** Sve navedene datoteke i direktoriji nalaze se unutar navedenog Git repozitorija.

Potrebni resursi

Prije početka rješavanja laboratorijske vježbe, potrebno je klonirati Git repozitorij s adrese: <https://github.com/masstuhne/fer-sosa-laboratory-exercise-openidconnect>.

Unutar repozitorija nalaze se svi resursi potrebni za izvođenje i testiranje laboratorijske vježbe:

- **Prevedeni Java program:** „openidconnect-0.0.1.jar“
- **Konfiguracijske datoteke** za dvije odvojene Java aplikacije
 - U njima se nalaze podatci koji se koriste prilikom registracije aplikacije na Keycloak poslužitelja.
- **Datoteka u JSON formatu** koja sadrži sve konfiguracijske vrijednosti potrebne za registraciju obje aplikacije na Keycloak poslužitelj te simulaciju registracije korisnika u Keycloak sustav (testni Keycloak račun s vjerodajnicama).
 - Unutar datoteke navedene su i vjerodajnice testnog računa.
- **Docker-compose datoteka** koja služi za pokretanje Keycloak poslužitelja.
- **Bash skripta** koja služi za pokretanje obje Java aplikacije.
 - Ako se vježba ne izvodi na Linuxu, skriptu je potrebno prilagoditi vašem operacijskom sustavu.

Navedeni repozitorij ne sadrži izvorni kod aplikacija, već samo gotovu .jar datoteku.

Izvorni se kod može pronaći u repozitoriju: <https://github.com/masstuhne/fer-sosa-laboratory-exercise-openidconnect-code>.

Pokretanje

Nakon pripreme okoline potrebno je:

1. Pokrenuti Keycloak poslužitelj (pričekajte da se inicijalizira prije daljnjih koraka – cca. 20tak sekundi).
2. Pokrenuti dvije Java aplikacije od kojih svaka koristi vlastitu konfiguracijsku datoteku. U tu se svrhu koristi priložena skripta.

3. Otvoriti adrese „http://localhost:8081“ i „http://localhost:8082“.

4. Ispitati ponašanje autentifikacije – ulogirati se u jednu aplikaciju i zatim provjeriti je li pristup drugoj aplikaciji moguć bez ponovne prijave.

Rješenje vježbe

1. Pokretanje Keycloak poslužitelja korištenjem docker-compose datoteke.

```
✖ ~ /f/f/s/l/sosa_laboratory_exercise on master ~ docker compose up
✓ Network sosa_laboratory_exercise_default Created
✓ Container keycloak Created
Attaching to keycloak
keycloak Updating the configuration and installing your custom providers, if any. Please wait
keycloak 2025-05-09 18:49:43,669 INFO [io.qua.dep.QuarkusAugmentor] (main) Quarkus augmentation completed in 5891ms
keycloak 2025-05-09 18:49:45,146 WARN [org.infinispan.CONFIG] (keycloak-cache-init) ISPN000569: Unable to persist Infinispan internal caches as no global state enabled
keycloak 2025-05-09 18:49:45,196 INFO [org.infinispan.CONTAINER] (keycloak-cache-init) ISPN000556: Starting user marshaller 'org.infinispan.jboss.marshalling.core.JBossUserMarshaller'
keycloak 2025-05-09 18:49:45,331 INFO [org.keycloak.quarkus.runtime.hostname.DefaultHostnameProvider] (main) Hostname settings: Base URL: <unset>, Hostname: <request>, Strict HTTPS: false, Path: <request>, Strict Backchannel: false, Admin URL: <unset>, Admin: <request>, Port: -1, Proxied: false
keycloak 2025-05-09 18:49:46,797 WARN [io.quarkus.agroal.runtime.DataSources] (JPA Startup Thread) DataSource <default> enables XA but transaction recovery is not enabled. Please enable transaction recovery by setting quarkus.transaction-manager.enable-recovery=true, otherwise data may be lost if the application is terminated abruptly
keycloak 2025-05-09 18:49:47,557 INFO [org.keycloak.broker.provider.AbstractIdentityProviderMapper] (main) Registering class org.keycloak.broker.provider.mappersync.ConfigSyncEventListener
keycloak 2025-05-09 18:49:47,577 INFO [org.keycloak.connections.infinispan.DefaultInfinispanConnectionFactory] (main) Node name: node_942128, Site name: null
keycloak 2025-05-09 18:49:48,573 INFO [org.keycloak.quarkus.runtime.storage.legacy.liquibase.QuarkusJpaUpdaterProvider] (main) Initializing database schema. Using changelog META-INF/jpa-changelog-master.xml
keycloak
keycloak UPDATE SUMMARY
keycloak Run: 124
keycloak Previously run: 0
keycloak Filtered out: 0
keycloak -----
keycloak Total change sets: 124
keycloak
keycloak 2025-05-09 18:49:51,102 INFO [org.keycloak.services] (main) KC-SERVICES0050: Initializing master realm
keycloak 2025-05-09 18:49:52,015 INFO [org.keycloak.exportimport.dir.DirImportProvider] (main) Importing from directory /opt/keycloak/bin/./data/import
keycloak 2025-05-09 18:49:52,016 INFO [org.keycloak.services] (main) KC-SERVICES0030: Full model import requested. Strategy: IGNORE_EXISTING
keycloak 2025-05-09 18:49:53,431 INFO [org.keycloak.exportimport.util.ImportUtil] (main) Realm 'sso-realm' imported
keycloak 2025-05-09 18:49:53,523 INFO [org.keycloak.services] (main) KC-SERVICES0032: Import finished successfully
keycloak 2025-05-09 18:49:53,618 INFO [io.quarkus] (main) Keycloak 24.0.3 on JVM (powered by Quarkus 3.8.3) started in 9.744s. Listening on: http://0.0.0.0:8080
keycloak 2025-05-09 18:49:53,619 INFO [io.quarkus] (main) Profile dev activated.
keycloak 2025-05-09 18:49:53,619 INFO [io.quarkus] (main) Installed features: [agroal, cdi, hibernate-orm, jdbc-h2, keycloak, logging-gelf, narayana-jta, reactive-routes, resteasy-reactive, resteasy-reactive-jackson, smallrye-context-propagation, smallrye-health, vertx]
keycloak 2025-05-09 18:49:53,874 INFO [org.keycloak.services] (main) KC-SERVICES0009: Added user 'admin' to realm 'master'
keycloak 2025-05-09 18:49:53,876 WARN [org.keycloak.quarkus.runtime.KeycloakMain] (main) Running the server in development mode. DO NOT use this configuration in production.
```

2. Pokretanje Java aplikacija pomoću skripte.

```
1:docker 2:fish
✖ ~ /f/f/s/l/sosa_laboratory_exercise on master ~ ./run_java_apps.sh
Starting app1...
app1 started with PID 12787
Starting app2...
app2 started with PID 12788
```

3. Što možemo vidjeti na “http://localhost:8081” i “http://localhost:8082”.



Everybody can see this!

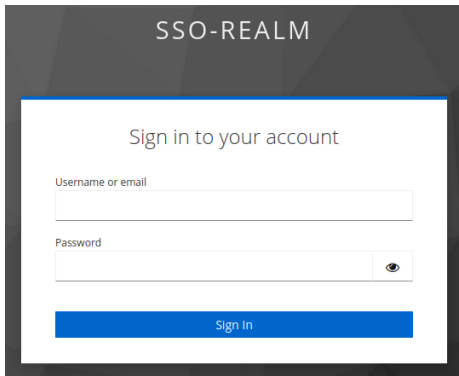
Click [here](#) to view personalized hello page.



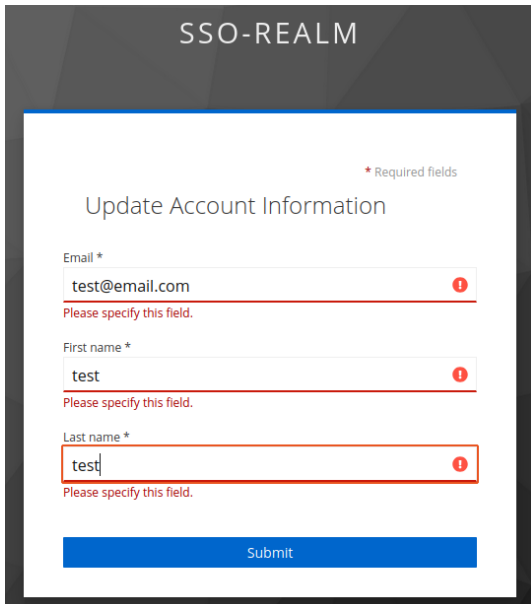
Everybody can see this!

Click [here](#) to view personalized hello page.

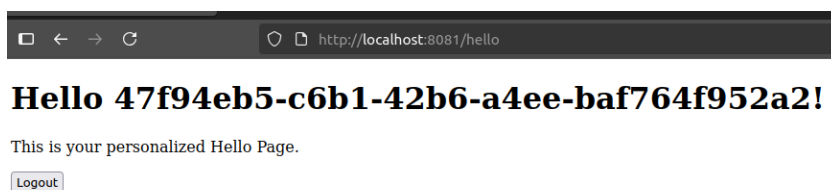
4. Odlučimo se ulogirati u aplikaciju koja se izvršava na “http://localhost:8081” te smo preusmjereni na središnju Keycloak stranicu za autentifikaciju. Koristimo vjerodajnice: “user1” i “password”.

The image shows the Keycloak SSO-REALM login page. At the top, it says "SSO-REALM". Below that, there's a heading "Sign in to your account". There are two input fields: "Username or email" and "Password". The "Password" field has a toggle icon (an eye) to the right of it. At the bottom, there is a blue "Sign In" button.

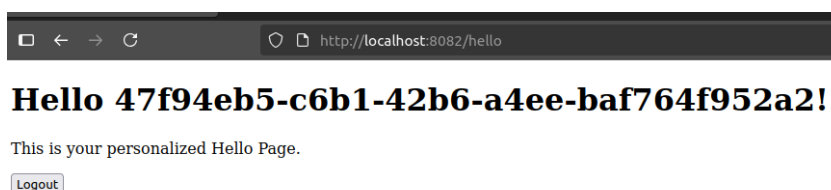
5. Popunimo potrebne podatke.

The image shows the Keycloak SSO-REALM "Update Account Information" page. At the top, it says "SSO-REALM". Below that, there's a heading "Update Account Information". There's a red asterisk icon and the text "Required fields". There are three input fields: "Email *", "First name *", and "Last name *". Each field has a red border and a red exclamation mark icon to its right. Below each field, there is a red error message: "Please specify this field." At the bottom, there is a blue "Submit" button.

6. Preusmjereni smo natrag na našu aplikaciju.



7. Odlučimo se ulogirati u aplikaciju koja se izvršava na “http://localhost:8082” te smo direktno preusmjereni na našu aplikaciju zahvaljujući aktivnoj sjednici kod Keycloak poslužitelja.



Zaključak

Kroz ovu vježbu stečeno je razumijevanje funkcioniranja sustava za jedinstvenu prijavu (SSO) i njegova implementacija korištenjem OpenID Connect protokola. Također, u praksi je demonstrirana integracija Keycloak-a s web aplikacijama temeljenim na Spring Boot-u. SSO se pokazao kao vrlo korisno rješenje koje olakšava korisničku autentifikaciju i povećava sigurnost.

Literatura

- Keycloak službena dokumentacija: <https://www.keycloak.org/server/containers>
- OpenID Connect službena dokumentacija: <https://openid.net/developers/how-connect-works/>
- Spring Security OAuth 2.0: <https://docs.spring.io/spring-security/reference/servlet/oauth2/index.html>