OCI | WE ARE SOFTWARE ENGINEERS.

WEBINAR

MICRONAUT SECURITY

SERGIO DEL AMO

objectcomputing.com

SERGIO DEL AMO

- MICRONAUT / GRAILS OCI TEAM
- GUADALAJARA, SPAIN
- CURATOR OF GROOVYCALAMARI.COM
- PODCAST HOST OF PODCAST.GROOVYCALAMARI.COM
- GREACH Conference organizer
- @SDELAMO
- HTTP://SERGIODELAMO.ES

CONTROLLER EXAMPLE

```
@Controller(“/books")
public class BookController {

    @Get
    public List<Book> index() {
        return Arrays.asList(new Book("1491950358", "Building Microservices"),
                             new Book("1680502395", "Release It!"),
                             new Book("0321601912", "Continuous Delivery"));
    }
}
```

INSTALLATION

SECURITY INSTALLATION

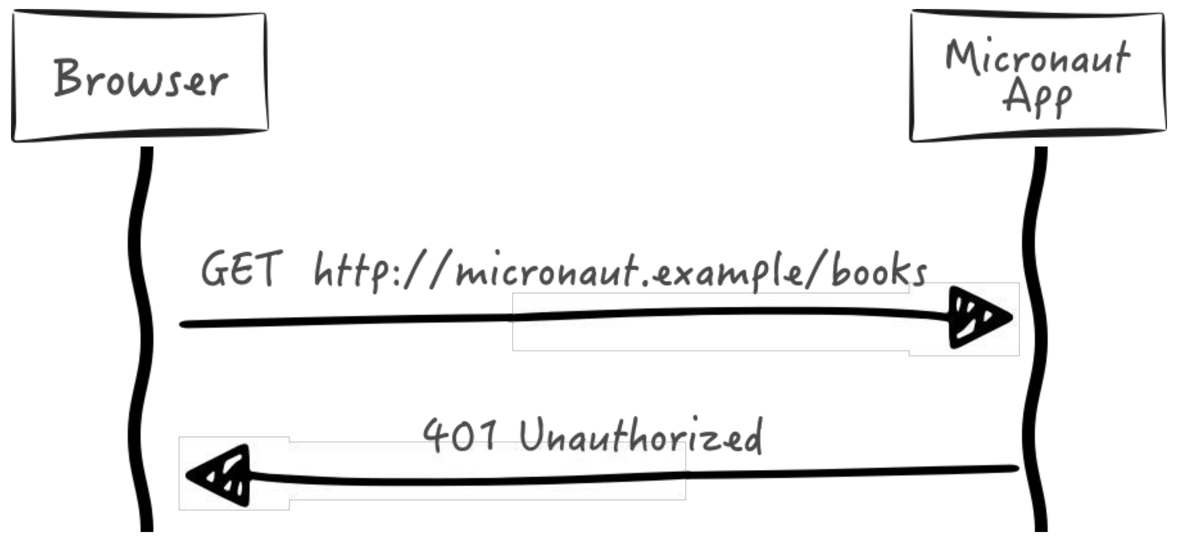
build.gradle

```
dependencies {
    ...
    ..
    .
    annotationProcessor "io.micronaut:micronaut-security"
    compile "io.micronaut:micronaut-security"
}
```

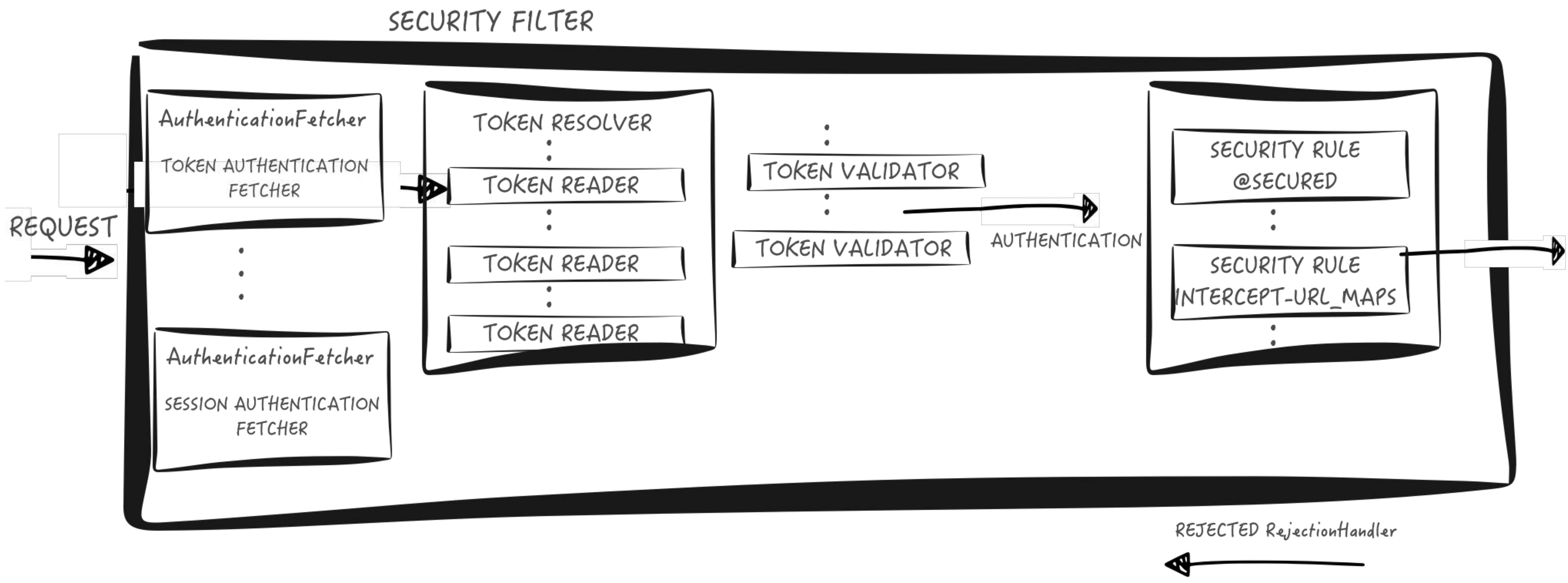
src/main/resources/application.yml

```
micronaut:
  security:
    enabled: true
```

SECURED BY DEFAULT



Security Filter



ANONYMOUS ACCESS

@Secured IS_ANONYMOUS

```java
import io.micronaut.security.annotation.Secured;

@Controller(“/books")
public class BookController {

    @Secured(SecurityRule.IS_ANONYMOUS)
    @Get
    public List<Book> index() {
        return Arrays.asList(new Book("1491950358", "Building Microservices"),
                             new Book("1680502395", "Release It!"),
                             new Book("0321601912", "Continuous Delivery"));
    }
}
```

@Secured IS_ANONYMOUS

```
import io.micronaut.security.annotation.Secured;

@Secured(SecurityRule.IS_ANONYMOUS)
@Controller(“/books")
public class BookController {

    @Get
    public List<Book> index() {
        return Arrays.asList(new Book("1491950358", "Building Microservices"),
                             new Book("1680502395", "Release It!"),
                             new Book("0321601912", "Continuous Delivery"));
    }
}
```

JSR_250 annotations

```
import javax.annotation.security.PermitAll;

@Controller(“/books")
public class BookController {

    @PermitAll
    @Get
    public List<Book> index() {
        return Arrays.asList(new Book("1491950358", "Building Microservices"),
                             new Book("1680502395", "Release It!"),
                             new Book("0321601912", "Continuous Delivery"));
    }
}
```

INTERCEPT URL MAP

src/main/java/example/micronaut/BookController.java

```
@Controller(“/books")
public class BookController {

    @Get
    public List<Book> index() {

        return Arrays.asList(new Book("1491950358", "Building Microservices"),
                             new Book("1680502395", "Release It!"),
                             new Book("0321601912", "Continuous Delivery"));
    }
}
```

src/main/resources/application.yml

```
micronaut:
  security:
    enabled: true
    intercept-url-map:
      -
        pattern: "/books"
        http-method: GET
        access:
          - isAnonymous()
```

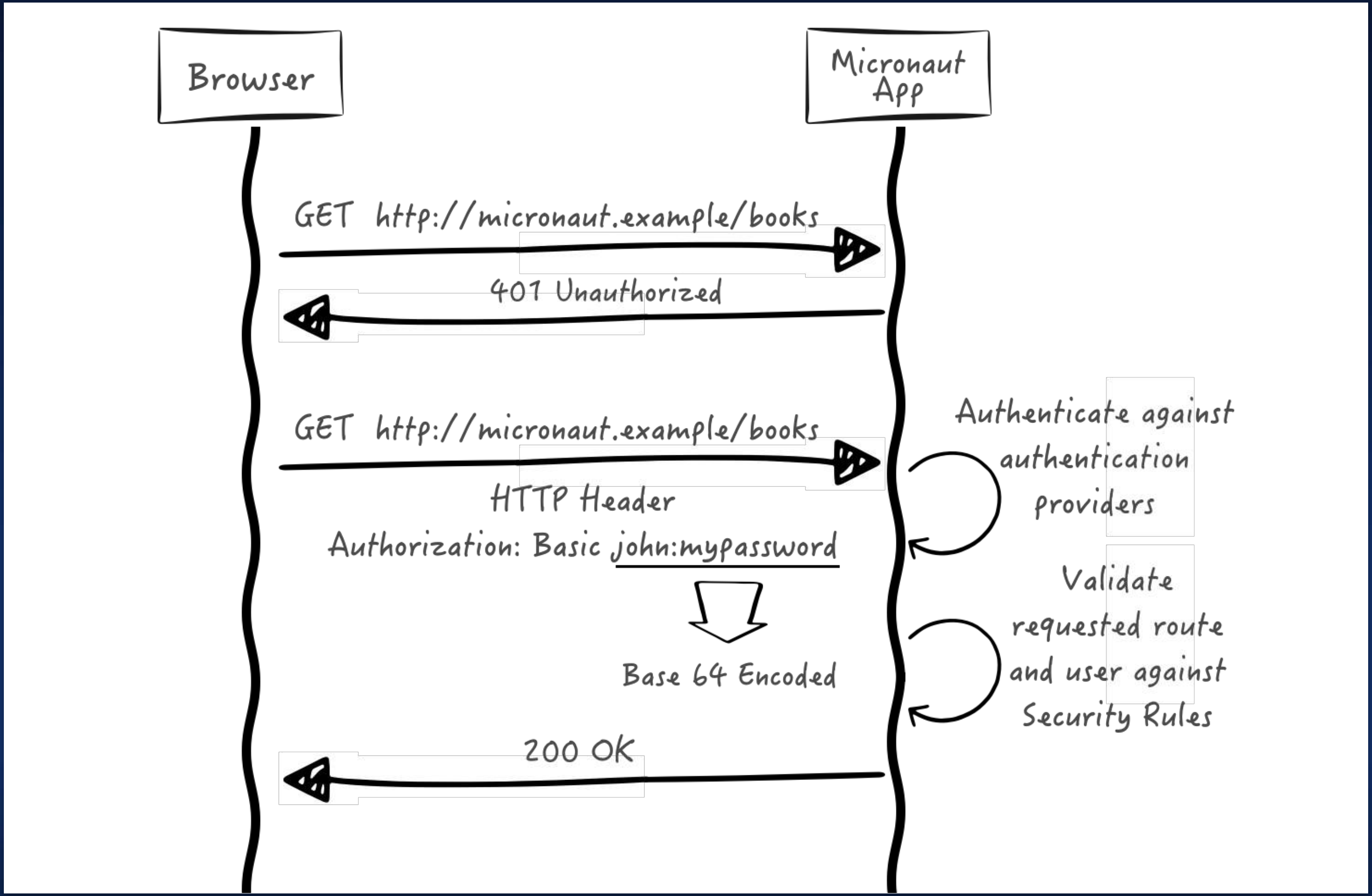
INTERCEPT URL MAP for STATIC RESOURCES

src/main/resources/application.yml

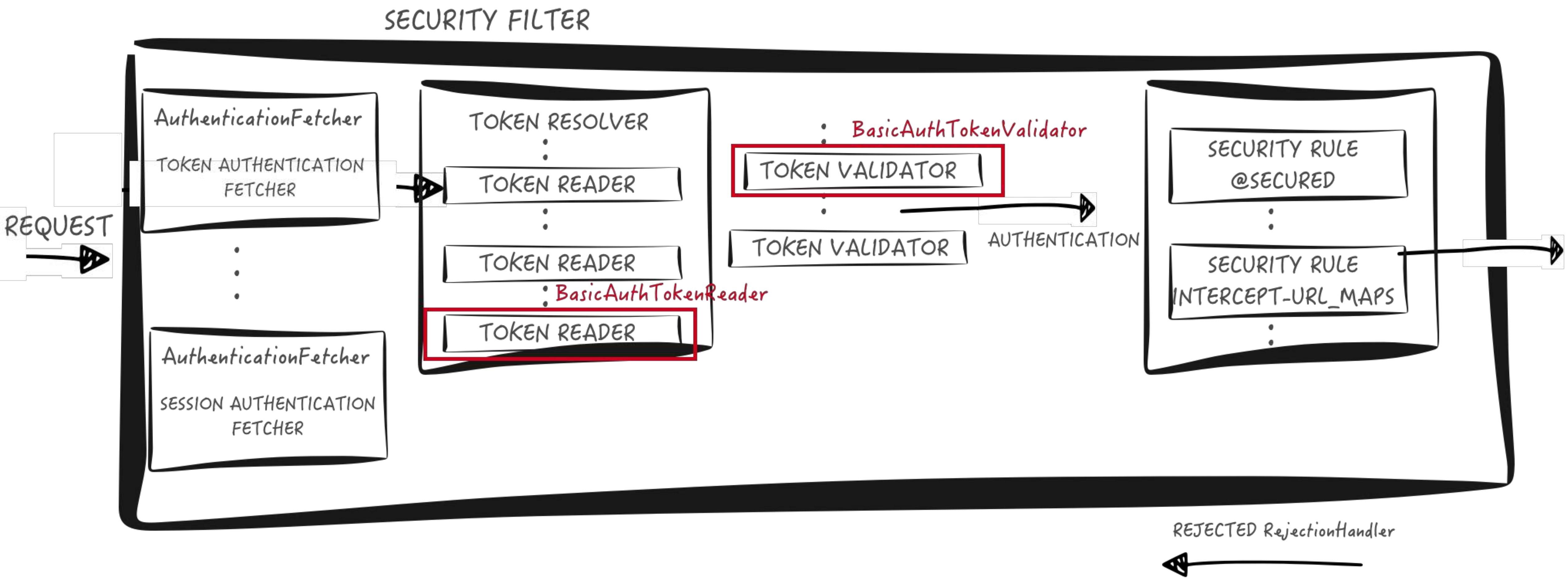
```
micronaut:
  router:
    static-resources:
      deafult:
        enabled: true
        mapping: /static/**
        paths:
          - classpath: public
  security:
    enabled: true
    intercept-url-map:
      -
        pattern: "/static/logo.png"
        http-method: GET
        access:
          - isAnonymous()
```

BASIC AUTH

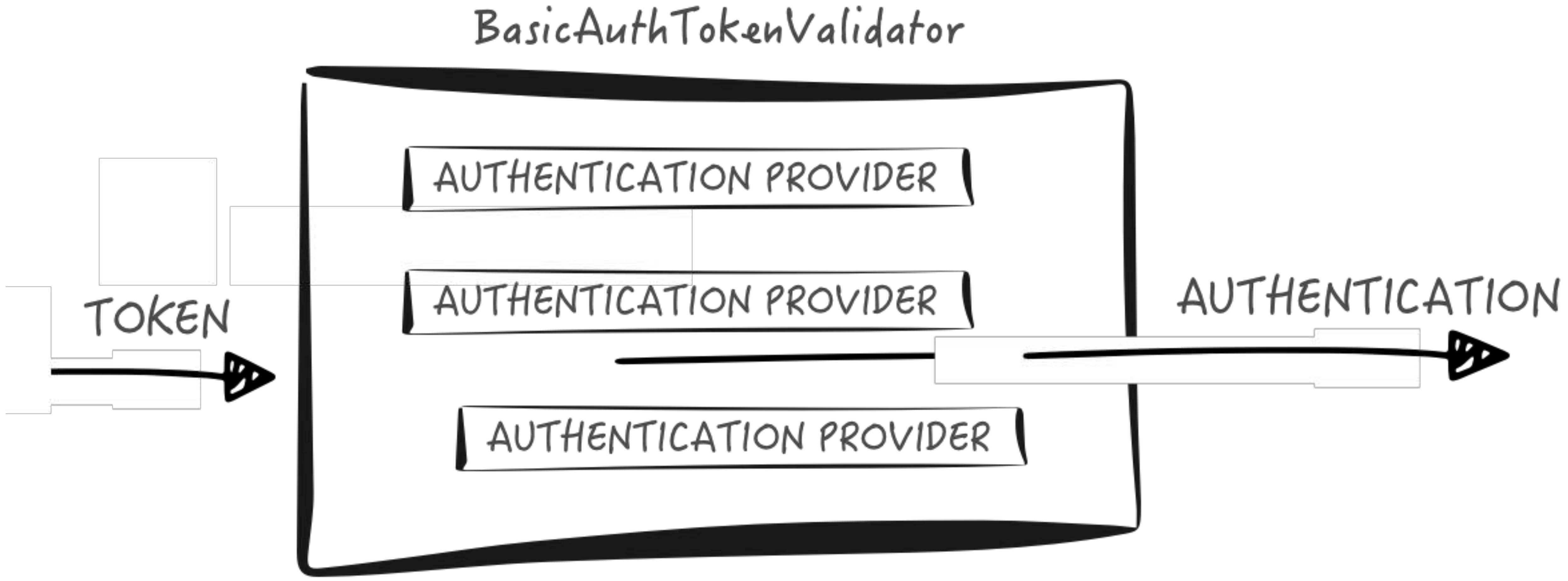
BASIC AUTH


Browser
Micronaut App
GET http://micronaut.example/books
401 Unauthorized
GET http://micronaut.example/books
HTTP Header
Authorization: Basic john:mypassword
Base 64 Encoded
Authenticate against authentication providers
Validate requested route and user against Security Rules
200 OK

Basic Auth


SECURITY FILTER
REQUEST
AuthenticationFetcher
TOKEN AUTHENTICATION FETCHER
AuthenticationFetcher
SESSION AUTHENTICATION FETCHER
TOKEN RESOLVER
TOKEN READER
TOKEN READER
BasicAuthTokenReader
TOKEN READER
BasicAuthTokenValidator
TOKEN VALIDATOR
TOKEN VALIDATOR
AUTHENTICATION
SECURITY RULE @SECURED
SECURITY RULE INTERCEPT-URL_MAPS
REJECTED RejectionHandler

Basic Auth


BasicAuthTokenValidator
AUTHENTICATION PROVIDER
AUTHENTICATION PROVIDER
AUTHENTICATION PROVIDER
TOKEN
AUTHENTICATION

BASIC AUTH

curl with basic auth

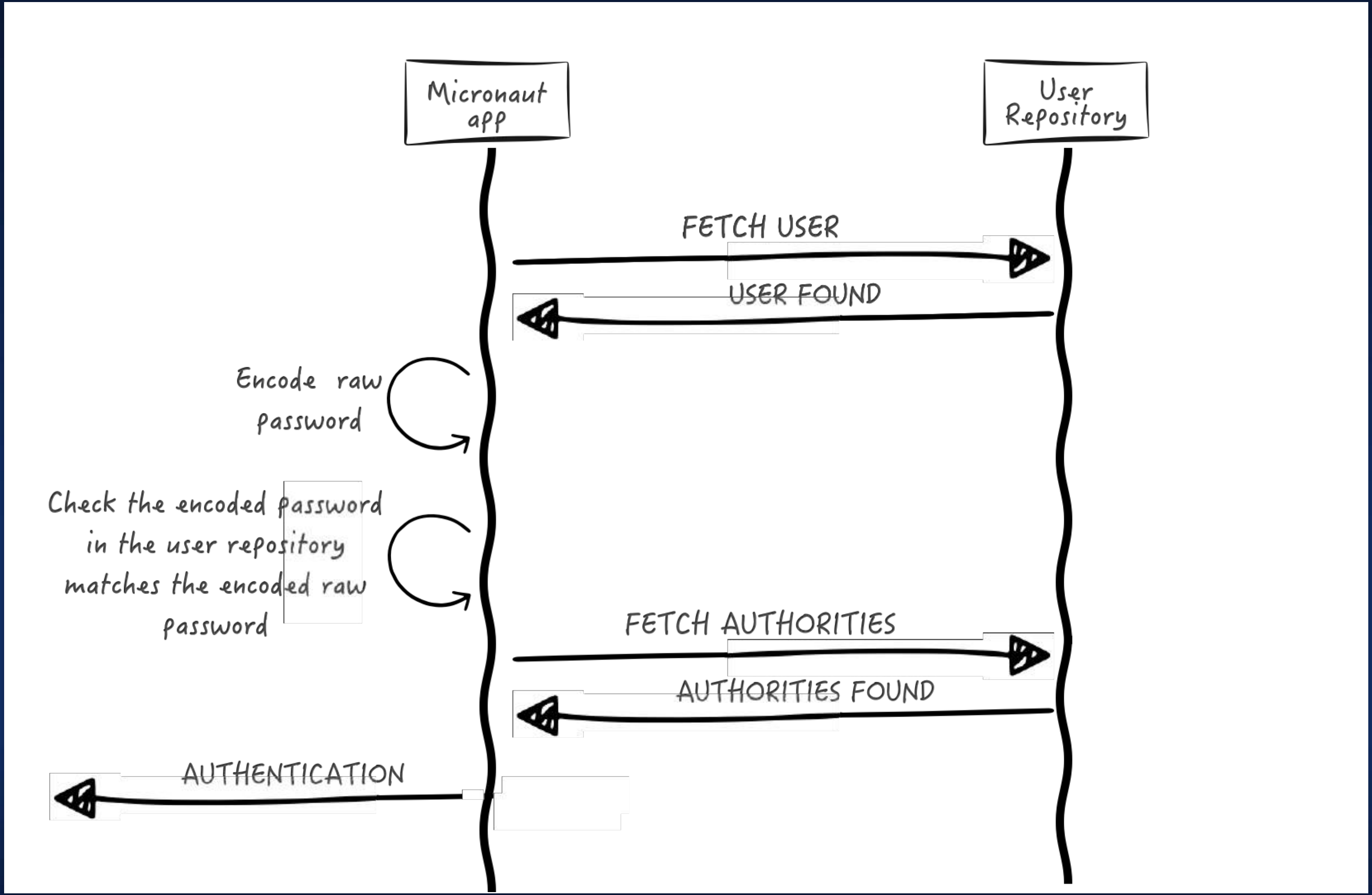
```
$ curl – u name:password http://micronaut.example/books
```

```
import javax.inject.Singleton

@Singleton
public class ExampleAuthenticationProvider implements AuthenticationProvider {
    @Override
    public Publisher<AuthenticationResponse> authenticate(AuthenticationRequest authenticationRequest) {
        if (authenticationRequest.getIdentity().equals("user") &&
            authenticationRequest.getSecret().equals("password"))) {
           UserDetails u = new UserDetails(authenticationRequest.getIdentity(),
                                       Arrays.asList("ROLE_USER"));
           return Flowable.just(u);
        }
        return Flowable.just(new AuthenticationFailed());
    }
}
```

DELEGATING AUTHENTICATION PROVIDER

DELEGATION AUTHENTICATION PROVIDER



DELEGATING AUTHENTICATION PROVIDER

implementation of UserFetcher

```
import javax.inject.Singleton

@CompileStatic
@Singleton
class UserFetcherService implements UserFetcher {

    protected final UserGormService userGormService

    UserFetcherService(UserGormService userGormService) {
        this.userGormService = userGormService
    }

    @Override
    Publisher<UserState> findByUsername(String username) {
        UserState user = userGormService.findByUsername(username) as UserState
        (user ? Flowable.just(user) : Flowable.empty()) as Publisher<UserState>
    }
}
```

DELEGATING AUTHENTICATION PROVIDER

implementation of AuthoritiesFetcher

```
package example.micronaut.services

import io.micronaut.security.authentication.providers.AuthoritiesFetcher
import io.reactivex.Flowable
import org.reactivestreams.Publisher

import javax.inject.Singleton

@Singleton
class AuthoritiesFetcherService implements AuthoritiesFetcher {

    protected final UserRoleGormService userRoleGormService

    AuthoritiesFetcherService(UserRoleGormService userRoleGormService) {
        this.userRoleGormService = userRoleGormService
    }

    @Override
    Publisher<List<String>> findAuthoritiesByUsername(String username) {
        Flowable.just(userRoleGormService.findAllAuthoritiesByUsername(username))
    }
}
```

DELEGATING AUTHENTICATION PROVIDER

implementation of PasswordEncoder

build.gradle

```
dependencies {
  ...
    compile "org.springframework.security:spring-security-crypto:
4.2.5.RELEASE"
}
```

```
import io.micronaut.security.authentication.providers.PasswordEncoder
import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder
import javax.inject.Singleton

@Singleton
class BCryptPasswordEncoderService implements PasswordEncoder {

    org.springframework.security.crypto.password.PasswordEncoder delegate = new BCryptPasswordEncoder()

    String encode(String rawPassword) {
        return delegate.encode(rawPassword)
    }

    @Override
    boolean matches(String rawPassword, String encodedPassword) {
        return delegate.matches(rawPassword, encodedPassword)
    }
}
```

SESSION BASED AUTHENTICATION

SESSION AUTH

build.gradle

```
dependencies {
    ...
    ..
    .
    annotationProcessor "io.micronaut:micronaut-security-session"
    compile "io.micronaut:micronaut-security"
}
```

src/main/resources/application.yml

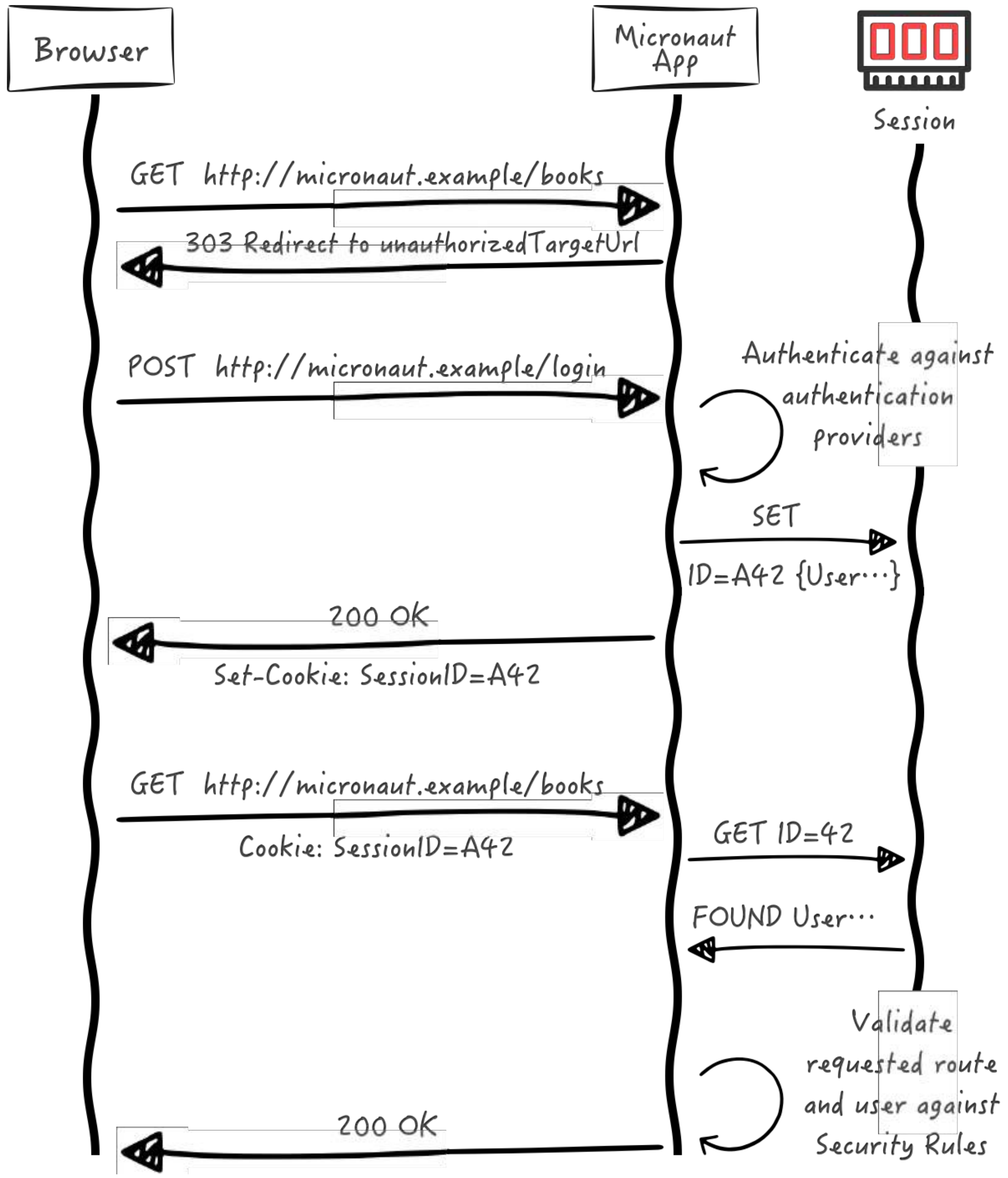
```
micronaut:
  security:
    enabled: true
    session:
      enabled: true
```

SECURITY SESSION CLI INSTALLATION

MICRONAUT SECURITY SESSION

```
$ mn create-app my-app --features security-session
```

Session Auth

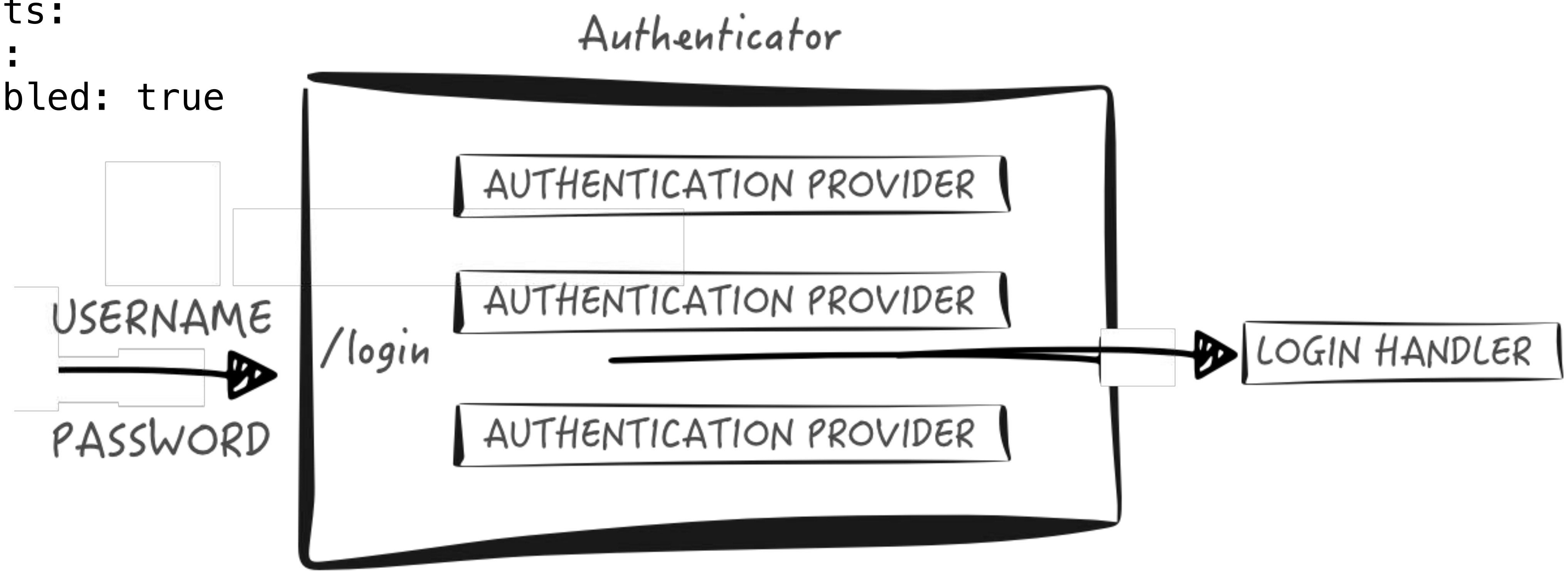

Browser
Micronaut App
Session
GET http://micronaut.example/books
303 Redirect to unauthorizedTargetUrl
POST http://micronaut.example/login
Authenticate against authentication providers
SET
ID=A42 {User…}
200 OK
Set-Cookie: SessionID=A42
GET http://micronaut.example/books
Cookie: SessionID=A42
GET ID=42
FOUND User…
Validate requested route and user against Security Rules
200 OK

ENDPOINTS

LOGIN CONTROLLER

src/main/resources/application.yml

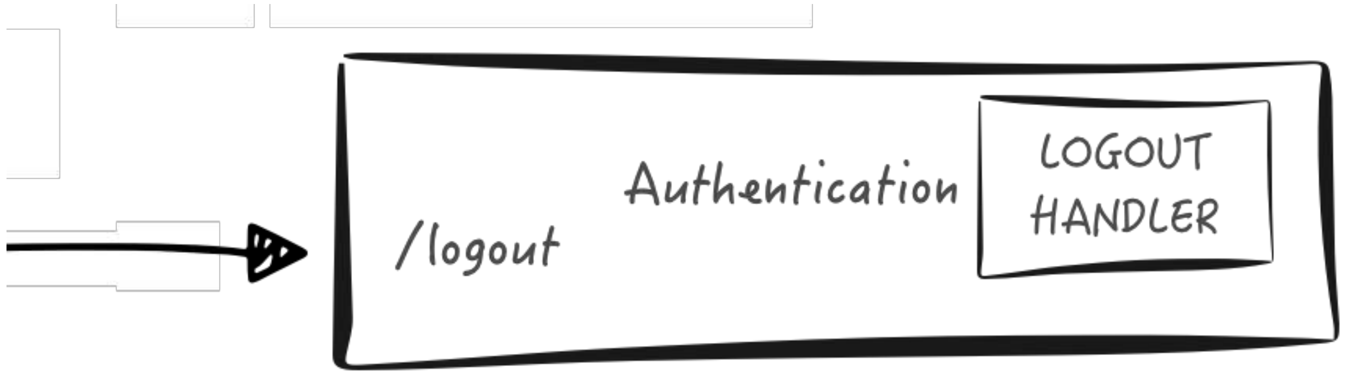
```
micronaut:
  security:
    enabled: true
    endpoints:
      login:
        enabled: true
```



LOGOUT CONTROLLER

src/main/resources/application.yml

```yaml
micronaut:
  security:
    enabled: true
    endpoints:
      logout:
        enabled: true
```



AUTHENTICATION

@Secured IS_AUTHENTICATED

```
import io.micronaut.security.annotation.Secured;

@Controller(“/books")
public class BookController {

    @Secured(SecurityRule.IS_AUTHENTICATED)
    @Get
    public List<Book> index() {
        return Arrays.asList(new Book("1491950358", "Building Microservices"),
                             new Book("1680502395", "Release It!"),
                             new Book("0321601912", "Continuous Delivery"));
    }
}
```

@Secured IS_AUTHENTICATED

```
import io.micronaut.security.annotation.Secured;

@Secured(SecurityRule.IS_AUTHENTICATED)
@Controller(“/books")
public class BookController {

    @Get
    public List<Book> index() {
        return Arrays.asList(new Book("1491950358", "Building Microservices"),
                             new Book("1680502395", "Release It!"),
                             new Book("0321601912", "Continuous Delivery"));
    }
}
```

AUTHORIZATION

AUTHORIZATION

```
import io.micronaut.security.annotation.Secured;

@Controller("/books")
public class BookController {

    @Secured({"ROLE_ADMIN","ROLE_USER"})
    @Get
    public List<Book> index() {
        return Arrays.asList(new Book("1491950358", "Building Microservices"),
                             new Book("1680502395", "Release It!"),
                             new Book("0321601912", "Continuous Delivery"));
    }
}
```

JSR_250 annotations

```
import javax.annotation.security.RolesAllowed;

@Controller("/books")
public class BookController {

    @RolesAllowed({"ROLE_ADMIN","ROLE_USER"})
    @Get
    public List<Book> index() {
        return Arrays.asList(new Book("1491950358", "Building Microservices"),
                             new Book("1680502395", "Release It!"),
                             new Book("0321601912", "Continuous Delivery"));
    }
}
```

RETRIEVE CURRENT USER

Retrieve the Authenticated User

```
import io.micronaut.security.annotation.Secured;
import java.security.Principal;
import javax.annotation.Nullable;

@Controller("/books")
public class BookController {

    @Secured(SecurityRule.IS_ANONYMOUS)
    @Get
    public List<Book> index(@Nullable Principal principal) {
        if (principal != null && principal.getName().equals("Harry Potter")) {
            return Arrays.asList(new Book("9781781102459", "Philosopher's Stone"));
        }
        return Arrays.asList(new Book("1491950358", "Building Microservices"),
                             new Book("1680502395", "Release It!"),
                             new Book("0321601912", "Continuous Delivery"));
    }
}
```

Retrieve the Authenticated User

```
import io.micronaut.security.annotation.Secured;
import java.security.Principal;

@Controller(“/books")
public class BookController {

    @Secured(SecurityRule.IS_AUTHENTICATED)
    @Get
    public List<Book> index(Principal principal) {
        if (principal.getName().equals("Harry Potter”)) {
            return Arrays.asList(new Book("9781781102459", "Philosopher's Stone”));
        }
        return Arrays.asList(new Book("1491950358", "Building Microservices"),
                             new Book("1680502395", "Release It!"),
                             new Book("0321601912", "Continuous Delivery"));
    }
}
```

Retrieve the Authenticated User

```
import io.micronaut.security.annotation.Secured;
import io.micronaut.security.authentication.Authentication;

@Controller(“/books")
public class BookController {

    @Secured(SecurityRule.IS_AUTHENTICATED)
    @Get
    public List<Book> index(Authentication authentication) {
        if (authentication.getName().equals("Harry Potter”)) {
            return Arrays.asList(new Book("9781781102459", "Philosopher's Stone”));
        }
        return Arrays.asList(new Book("1491950358", "Building Microservices"),
                             new Book("1680502395", "Release It!"),
                             new Book("0321601912", "Continuous Delivery"));
    }
}
```

Retrieve the Authenticated User

```java
import io.micronaut.security.annotation.Secured;
import io.micronaut.security.authentication.Authentication;

@Controller("/books")
public class BookController {

    private final SecurityService securityService;

    public BookController(SecurityService securityService) {
         this.securityService = securityService;
    }

    @Secured(SecurityRule.IS_AUTHENTICATED)
    @Get
    public List<Book> index() {
        if (securityService.getAuthentication().getName().equals(“Harry Potter”)) {
               return Arrays.asList(new Book("9781781102459", "Philosopher's Stone”));
          }
        return Arrays.asList(new Book("1491950358", "Building Microservices"),
                             new Book("1680502395", "Release It!"),
                             new Book("0321601912", "Continuous Delivery"));
    }
}
```

LDAP

LDAP

build.gradle

```
dependencies {
    ...
    ..
    .
    annotationProcessor "io.micronaut:micronaut-security"
    compile "io.micronaut:micronaut-security"
    compile "io.micronaut.configuration:micronaut-security-ldap"
}
```

src/main/resources/application.yml

```
micronaut:
  ..
  .
  security:
  ...
  ..
    ldap:
      default:
        enabled: true
        context:
          server: 'ldap://ldap.forumsys.com:389'
          managerDn: 'cn=read-only-admin,dc=example,dc=com'
          managerPassword: 'password'
        search:
          base: "dc=example,dc=com"
        groups:
          enabled: true
          base: "dc=example,dc=com"
```

- LDAP authentication in Micronaut supports configuration of **one or more LDAP servers** to autehtnicate with.
- Each server has it's own settings and can be enabled or disabled

JWT

SECURITY JWT INSTALLATION

build.gradle

```
dependencies {
    ...
    ..
    .
    annotationProcessor "io.micronaut:micronaut-security"
    compile "io.micronaut:micronaut-security-jwt"
}
```

src/main/resources/application.yml

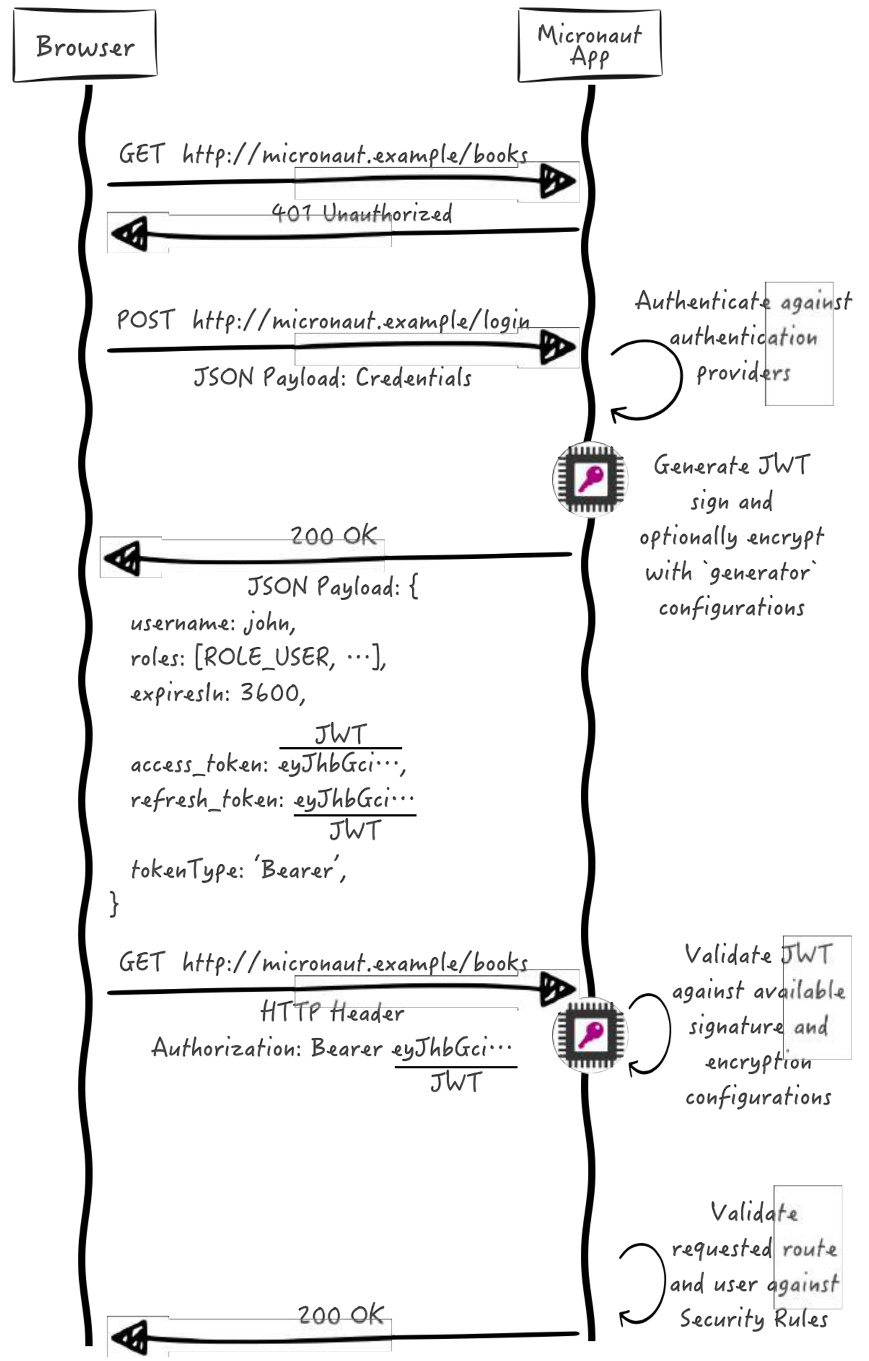
```
micronaut:
  security:
    enabled: true
    token:
      jwt:
        enabled: true
```

SECURITY JWT CLI INSTALLATION

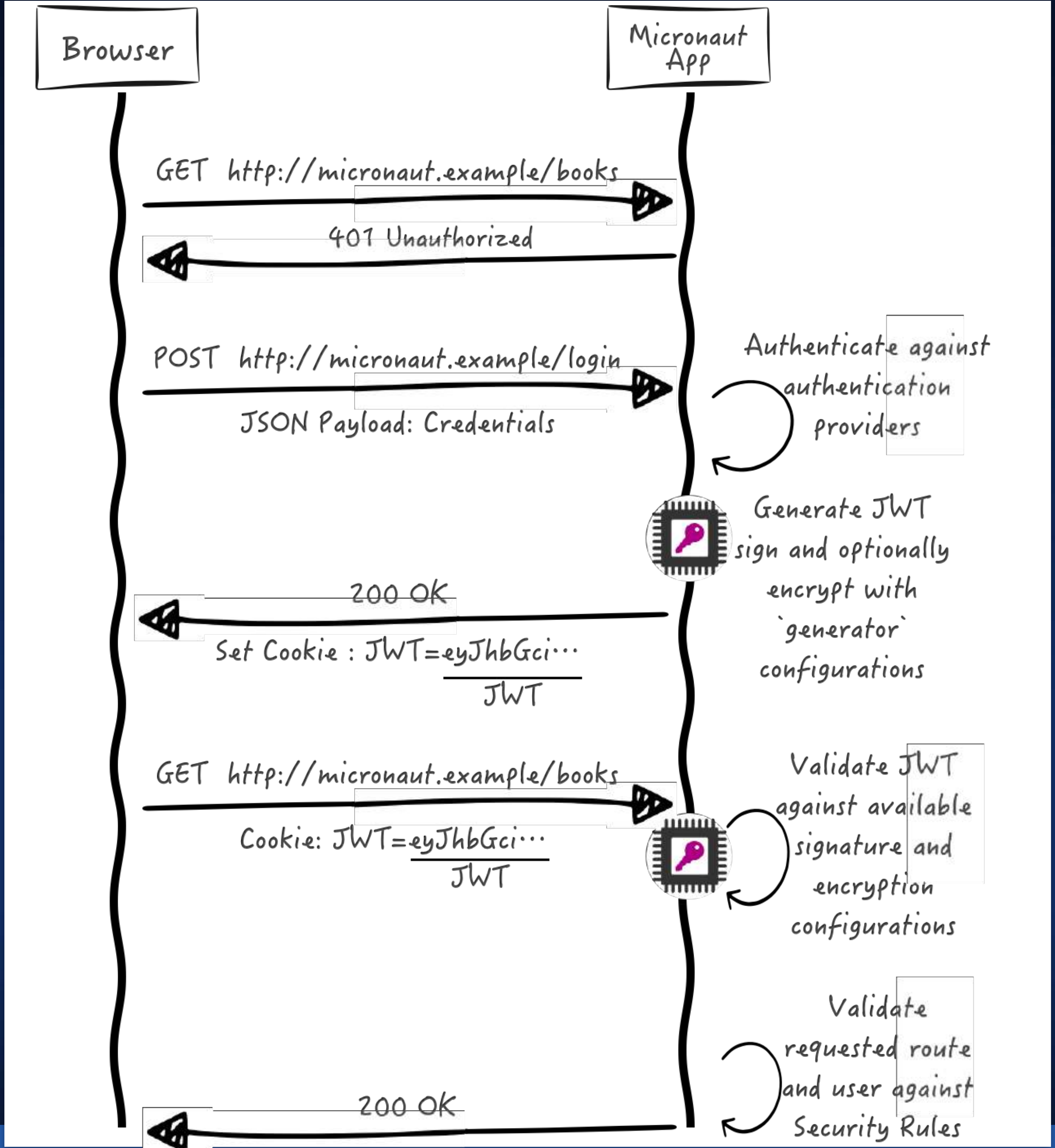
MICRONAUT SECURITY JWT

```
$ mn create-app my-app --features security-jwt
```

Bearer Token


Browser
Micronaut App
GET http://micronaut.example/books
401 Unauthorized
POST http://micronaut.example/login
JSON Payload: Credentials
Authenticate against authentication providers
Generate JWT sign and optionally encrypt with `generator` configurations
200 OK
JSON Payload: {
username: john,
roles: [ROLE_USER, ...],
expiresIn: 3600,
JWT
access_token: eyJhbGci...,
refresh_token: eyJhbGci...
JWT
tokenType: 'Bearer',
}
GET http://micronaut.example/books
HTTP Header
Authorization: Bearer eyJhbGci...
JWT
Validate JWT against available signature and encryption configurations
Validate requested route and user against Security Rules
200 OK

COOKIE JWT


Browser
Micronaut App
GET http://micronaut.example/books
401 Unauthorized
POST http://micronaut.example/login
JSON Payload: Credentials
Authenticate against authentication providers
Generate JWT sign and optionally encrypt with `generator` configurations
200 OK
Set Cookie : JWT=eyJhbGci···
JWT
GET http://micronaut.example/books
Cookie: JWT=eyJhbGci···
JWT
Validate JWT against available signature and encryption configurations
Validate requested route and user against Security Rules
200 OK

JWT Configuration

src/main/resources/application.yml

```
micronaut:
  security:
    enabled: true
    token:
      jwt:
        enabled: true
        signatures:
            secret:
                generator:
                    secret: pleaseChangeThisSecretForANewOne
                    jws-algorithm: HS256
```

JWT Signature Generation and Validation

To enable a JWT signature in token generation, you need to have in your app a bean of type RSASignatureGeneratorConfiguration, ECSignatureGeneratorConfiguration, SecretSignatureConfiguration qualified with name `generator`.

To verify signed JWT tokens, you need to have in your app a bean of type RSASignatureConfiguration, RSASignatureGeneratorConfiguration, ECSignatureGeneratorConfiguration, ECSignatureConfiguration, or SecretSignatureConfiguration.

Claims Validation

`io.micronaut.security.token.jwt.validator.GenericJwtClaimsValidator`

Bean	Description
`ExpirationJwtClaimsValidator`	Validate JWT is not expired.
SubjectNotNullJwtClaimsValidator	Validate JWT subject claim is not null.

Provide your own!

RFRESH CONTROLLER

src/main/resources/application.yml

```
micronaut:
  security:
    enabled: true
    endpoints:
      oauth:
        enabled: true
```

JSON Web Key JWK

A JSON Object that represents a cryptographic key. The members of the object represent properties of the key, including its value.

```
{
  "kty":"EC",
  "crv":"P-256",
  "kid":"test-personal-node",
  "x":"kdoE0JmUQra00UWJXHBwVvQetJ_L7vXt8nuXkaftKjo",
  "y":"PV7FUShMZ8Jg_kc2vjxgfwswEy26w_vWvVCHAGQ9tEQ"
}
```

JWK Set

A JSON object that represents a set of JWKs. The JSON object MUST have a "keys" member, which is an array of JWKs.

```
{
  "keys": [
      {
          "kty":"EC",
           "crv":"P-256",
           "kid":"123",
            "x":"kdoE0JmUQra00UWJXHBwVvQetJ_L7vXt8nuXkaftKjo",
            "y":"PV7FUShMZ8Jg_kc2vjxgfwswEy26w_vWvVCHAGQ9tEQ"
      }
}
```

KEYS CONTROLLER - Expose a JWK Set

```
import com.nimbusds.jose.jwk.JWK;
import io.micronaut.security.token.jwt.endpoints.JwkProvider;
import javax.inject.Singleton;
import java.text.ParseException;

@Singleton
class ExampleJwkProvider implements JwkProvider {
    @Override
    List<JWK> retrieveJsonWebKeys() {
        try {
            return [JWK.parse('''
{
  "kty":"EC",
  "crv":"P-256",
  "kid":"123",
  “x": "kdoE0JmUQra00UWJXHBwVvQetJ_L7vXt8nuXkaftKjo",
  "y":"PV7FUShMZ8Jg_kc2vjxgfwswEy26w_vWvVCHAGQ9tEQ"
}''')]
        } catch (ParseException e) {
            return [] as List<JWK>
        }
    }
}
```

src/main/resources/application.yml

```
micronaut:
  security:
    enabled: true
    endpoints:
      keys:
        enabled: true
```

KEYS CONTROLLER

```
$ curl localhost:8080/keys

{"keys":[{"kty":"EC","crv":"P-256","kid":"test-personal-
node","x":"kdoE0JmUQra00UWJXHBwVvQetJ_L7vXt8nuXkaftKjo","y":"PV7FUShMZ8Jg_kc2vjx
gfwswEy26w_vWvVCHAGQ9tEQ"}]}
```

REMOTE JWKS VALIDATION

src/main/resources/application.yml

```
micronaut:
  security:
    enabled: true
    token:
      jwt:
        enabled: true
        signatures:
          jwks:
            securityservice:
              url: "http://localhost:8081/keys"
```

SECURITY EVENTS

Security Events

Event Name	Description
LoginFailedEvent	Trigger when an unsuccessful login takes place.
LoginSuccessfulEvent	Trigger when a successful login takes place.
LogoutEvent	Triggered when the user logs out.
TokenValidatedEvent	Trigger when a token is validated.
AccessTokenGeneratedEvent	Trigger when a JWT access token is generated.
RefreshTokenGeneratedEvent	Trigger when a JWT refresh token is generated.

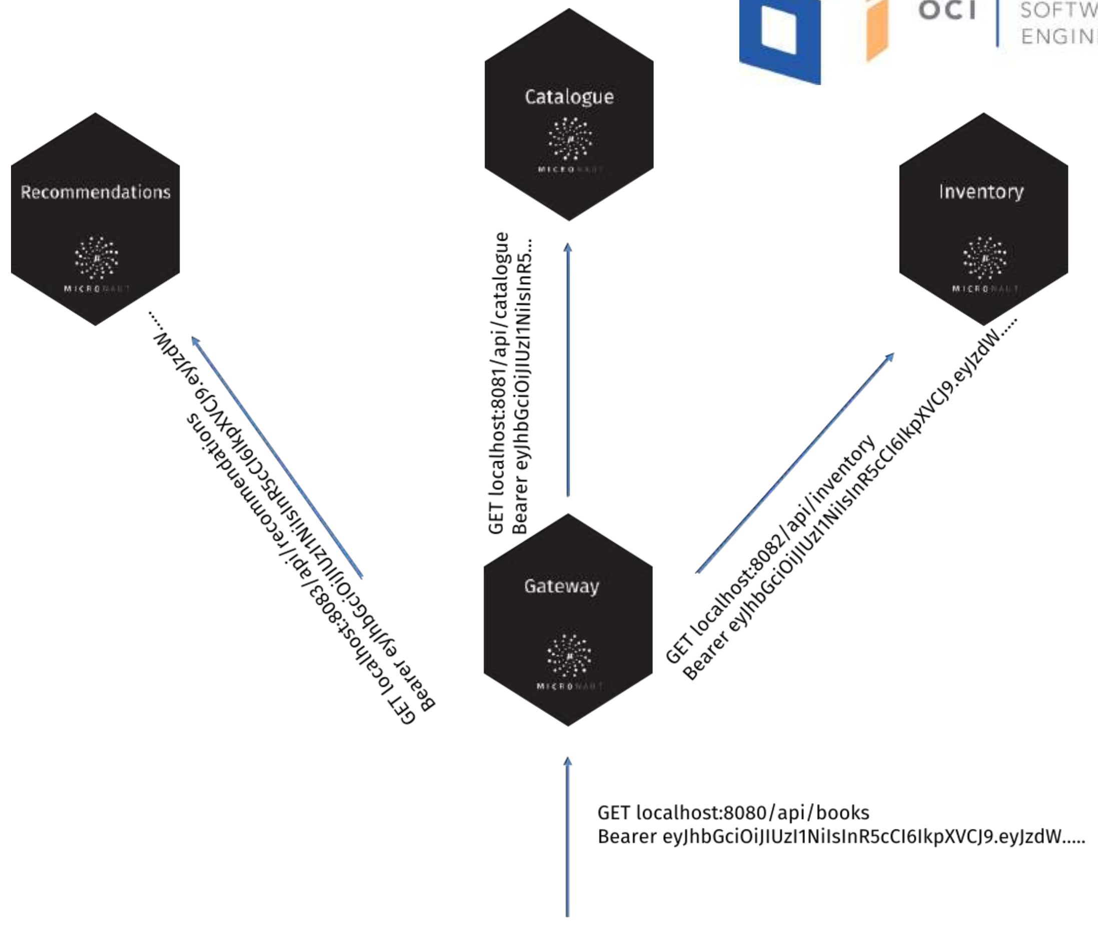
```
@Singleton
class LogoutFailedEventListener implements ApplicationEventListener<LogoutEvent> {
   @Override
    void onApplicationEvent(LogoutEvent event) {
        println "received logout event"
    }
}
```

TOKEN PROPAGATION

Token Propagation

src/main/resources/application.yml

```yaml
micronaut:
  security:
    enabled: true
    token:
      jwt:
        enabled: true
      writer:
          header:
              enabled: true
      propagation:
          enabled: true
          service-id-regex: "recommendations|catalogue|inventory"
```


Catalogue
Recommendations
Inventory
Gateway
GET localhost:8083/api/recommendations Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdW.....
GET localhost:8081/api/catalogue Bearer eyJhbGciOiJIUzI1NiIsInR5...
GET localhost:8082/api/inventory Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdW.....
GET localhost:8080/api/books Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdW.....

MICRONAUT OAUTH 2

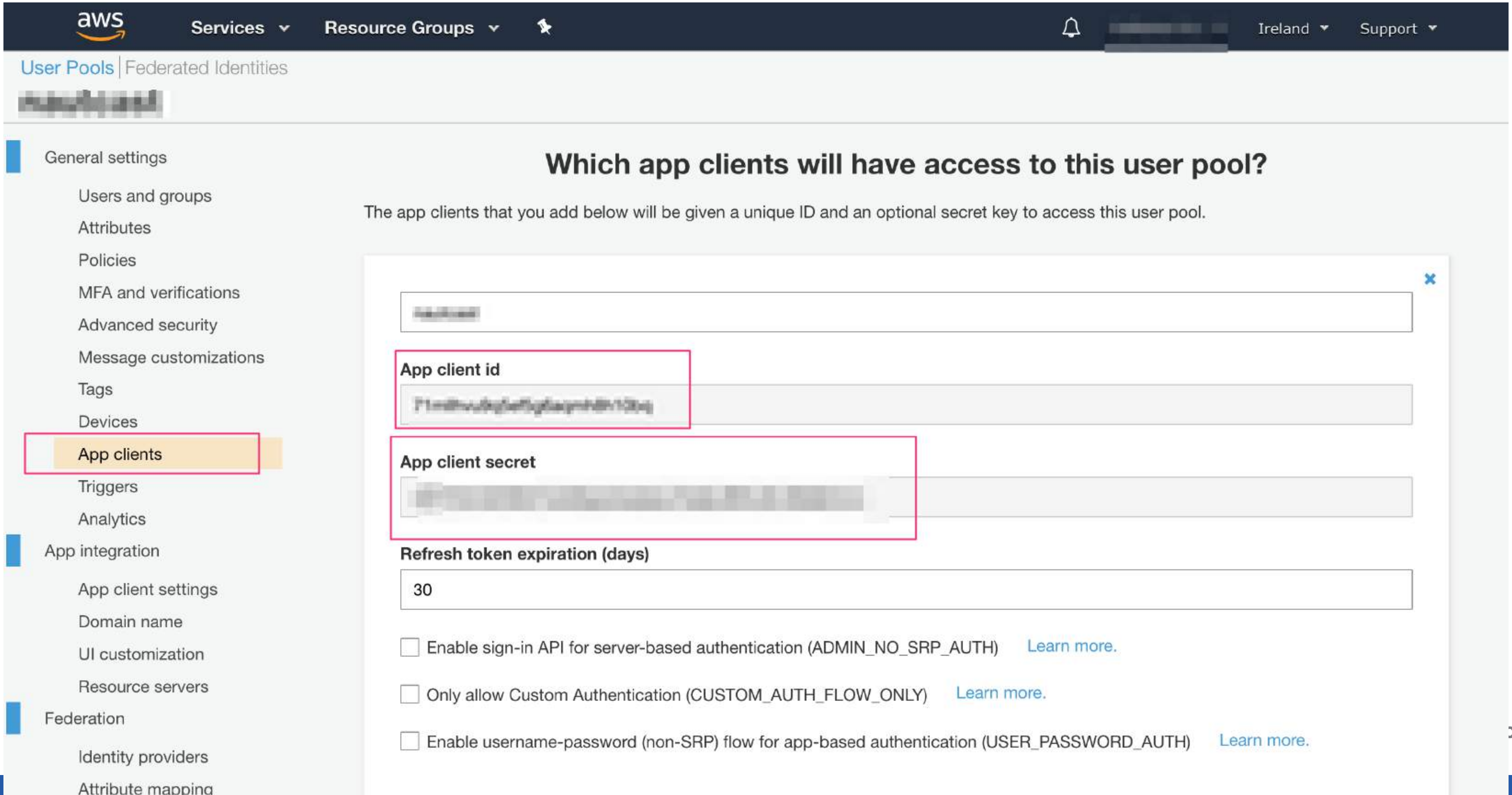
OAUTH 2

build.gradle

```
dependencies {
    ...
    ..
    .
    annotationProcessor "io.micronaut:micronaut-security"
    compile "io.micronaut:micronaut-security"
    compile "io.micronaut.configuration:micronaut-oauth2:1.0.0.BUILD-SNAPSHOT"
}
```

https://micronaut-projects.github.io/micronaut-oauth2/snapshot/guide/index.html

OAUTH 2



aws Services Resource Groups [redacted] Ireland Support

User Pools | Federated Identities

[redacted]

- General settings
  - Users and groups
  - Attributes
  - Policies
  - MFA and verifications
  - Advanced security
  - Message customizations
  - Tags
  - Devices
  - App clients
  - Triggers
  - Analytics
- App integration
  - App client settings
  - Domain name
  - UI customization
  - Resource servers
- Federation
  - Identity providers
  - Attribute mapping

Which app clients will have access to this user pool?

The app clients that you add below will be given a unique ID and an optional secret key to access this user pool.

[redacted]

App client id

[redacted]

App client secret

[redacted]

Refresh token expiration (days)

30

- [ ] Enable sign-in API for server-based authentication (ADMIN_NO_SRP_AUTH) Learn more.
- [ ] Only allow Custom Authentication (CUSTOM_AUTH_FLOW_ONLY) Learn more.
- [ ] Enable username-password (non-SRP) flow for app-based authentication (USER_PASSWORD_AUTH) Learn more.

OAUTH 2 Configuration

src/main/resources/application.yml

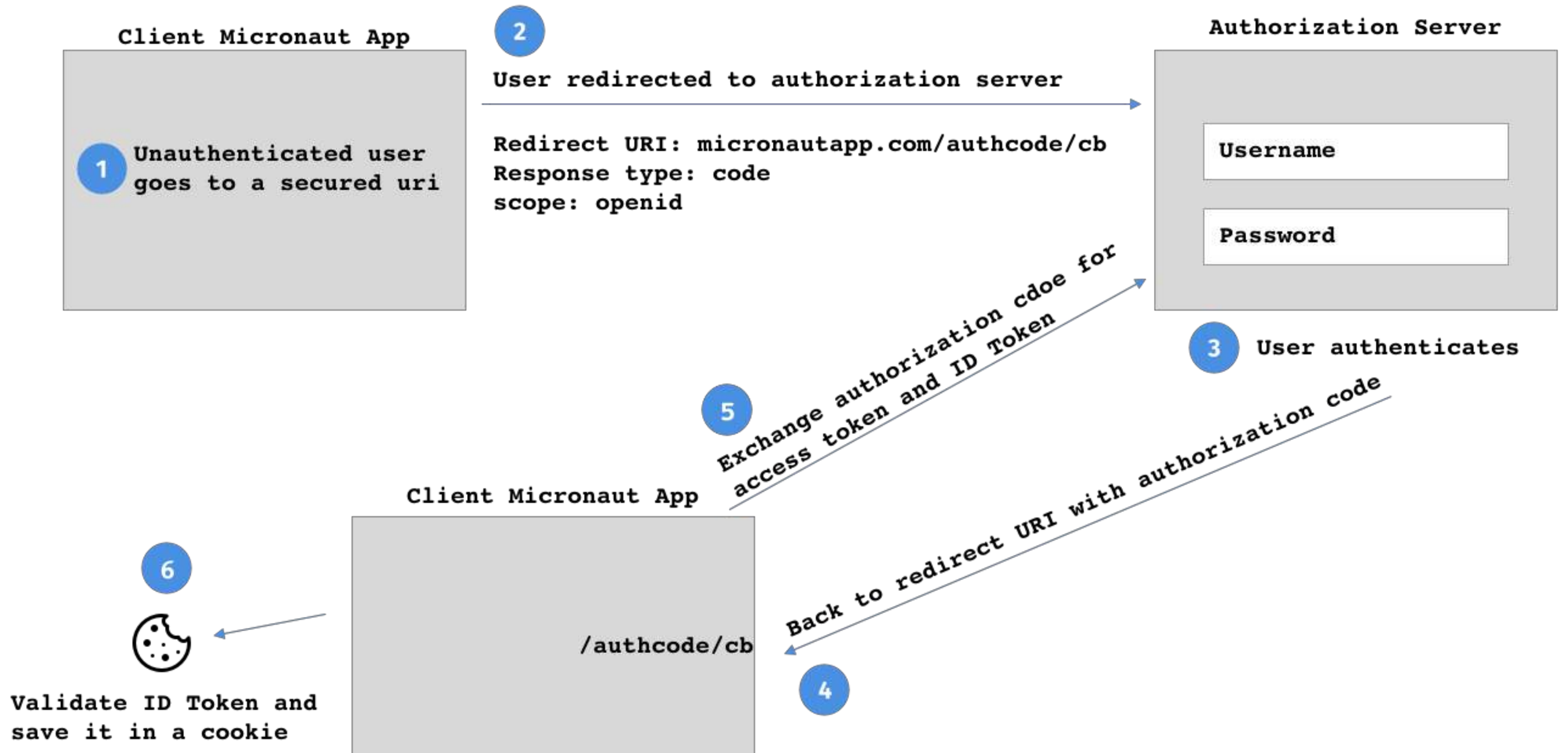
```
micronaut:
  security:
    enabled: true
    oauth2:
      client-secret: '${OAUTH_CLIENT_SECRET}'
      client-id: '${OAUTH_CLIENT_ID}'
      issuer: 'https://cognito-idp.${AWS_REGION}.amazonaws.com/${COGNITO_POOL_ID}'
```

OpenID Connect Authorization Code Flow

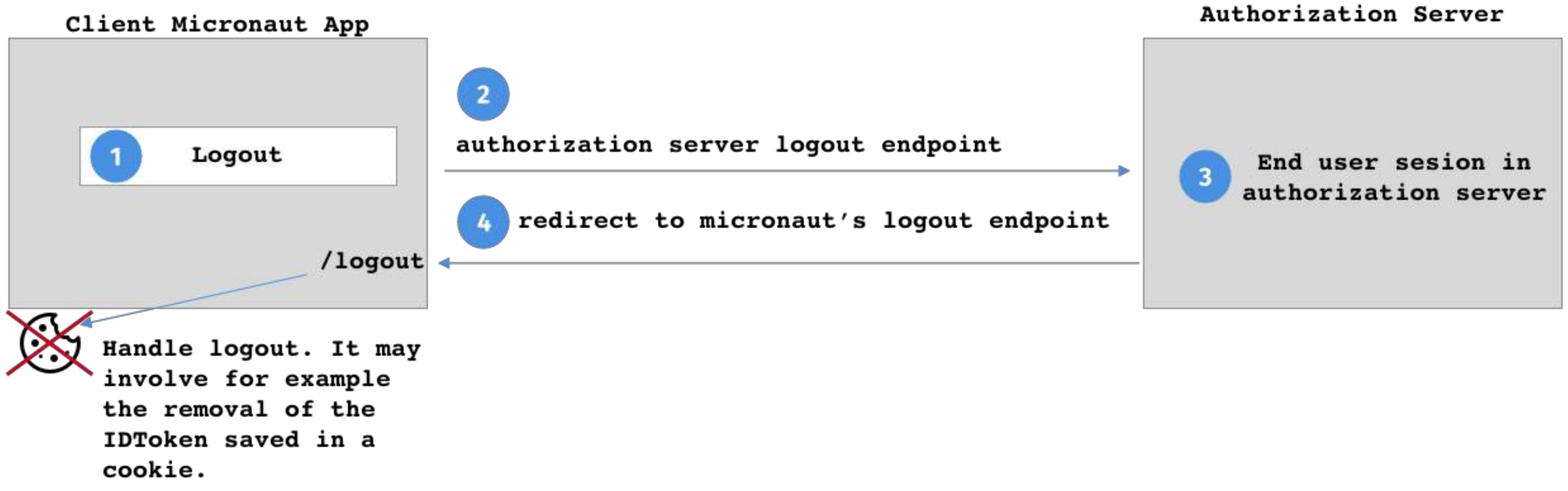
src/main/resources/application.yml

```yaml
micronaut:
  security:
    enabled: true
    oauth2:
      client-secret: '${OAUTH_CLIENT_SECRET}'
      client-id: '${OAUTH_CLIENT_ID}'
      issuer: ‘https://cognito-idp.${AWS_REGION}.amazonaws.com/${COGNITO_POOL_ID}'
    token:
      jwt:
        enabled: true
      cookie:
        enabled: true
```

OpenID Connect Authorization Code Flow

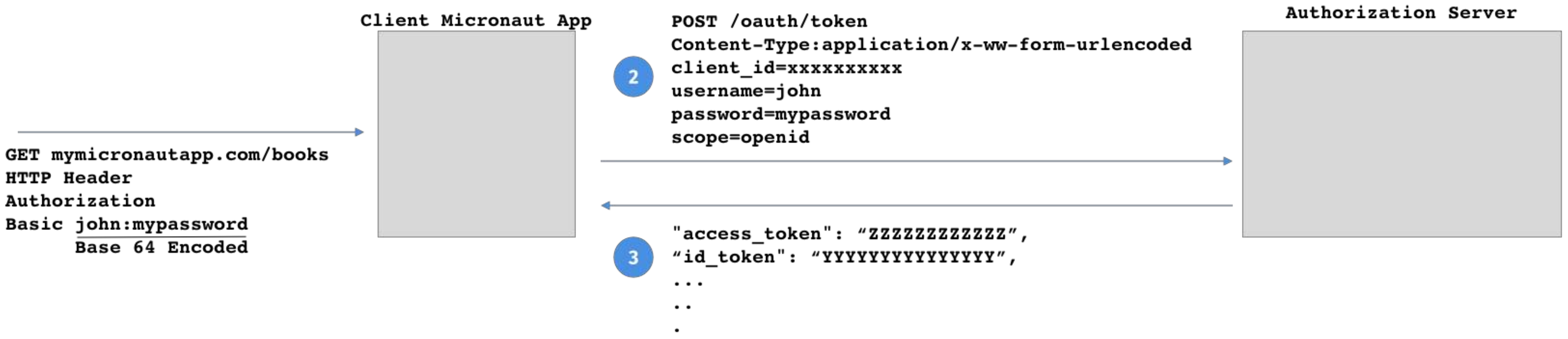

Client Micronaut App
1
Unauthenticated user goes to a secured uri
2
User redirected to authorization server
Redirect URI: micronautapp.com/authcode/cb
Response type: code
scope: openid
Authorization Server
Username
Password
3
User authenticates
Back to redirect URI with authorization code
4
Exchange authorization cdoe for access token and ID Token
5
Client Micronaut App
/authcode/cb
6
Validate ID Token and save it in a cookie

OpenID Connect Authorization Code Flow



Client Micronaut App
1 Logout
/logout
Handle logout. It may involve for example the removal of the IDToken saved in a cookie.
2
authorization server logout endpoint
4 redirect to micronaut's logout endpoint
Authorization Server
3 End user sesion in authorization server

Password Grant Type



```
GET mymicronautapp.com/books
HTTP Header
Authorization
Basic john:mypassword
      Base 64 Encoded
```

```
POST /oauth/token
Content-Type:application/x-ww-form-urlencoded
client_id=xxxxxxxxxx
username=john
password=mypassword
scope=openid
```

```
"access_token": “ZZZZZZZZZZZZ”,
“id_token": “YYYYYYYYYYYYYYY”,
...
..
.
```

SAMPLES

Micronaut Guides

Guides
Micronaut Basic Auth
Session based Authentication
Micronaut JWT Authentication
Micronaut JWT Authentication with Cookies
LDAP and Database authentication Providers
Micronaut Token Propagation
Secure a Micronaut app with Okta

https://guides.micronaut.io/tags/security.html

Questions?

LEARN MORE ABOUT OCI EVENTS & TRAINING

Events:

- objectcomputing.com/events

Training:

- objectcomputing.com/training
- grailstraining.com
- micronauttraining.com

Or email info@ocitraining.com to schedule a custom training program for your team online, on site, or in our state-of-the-art, Midwest training lab.

OCI | WE ARE SOFTWARE ENGINEERS.

CONNECT WITH US

1+ (314) 579-0066

@objectcomputing

objectcomputing.com