import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.ResponseEntity;

import org.springframework.security.authentication.AuthenticationManager;

import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;

import org.springframework.security.core.Authentication;

import org.springframework.web.bind.annotation.\*;

@RestController

@RequestMapping("/authenticate")

public class AuthController {

@Autowired

private AuthenticationManager authenticationManager;

@Autowired

private JwtTokenProvider jwtTokenProvider;

@PostMapping

public ResponseEntity<?> authenticate(@RequestParam String username, @RequestParam String password) {

Authentication authentication = authenticationManager.authenticate(

new UsernamePasswordAuthenticationToken(username, password)

);

String token = jwtTokenProvider.generateToken(authentication);

return ResponseEntity.ok(new AuthResponse(token));

}

}

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.security.config.annotation.authentication.builders.AuthenticationManagerBuilder;

import org.springframework.security.config.annotation.web.builders.HttpSecurity;

import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;

import org.springframework.security.config.annotation.web.configuration.WebSecurityConfigurerAdapter;

@Configuration

@EnableWebSecurity

public class SecurityConfig extends WebSecurityConfigurerAdapter {

@Override

protected void configure(HttpSecurity http) throws Exception {

http.csrf().disable()

.authorizeRequests()

.antMatchers("/authenticate").permitAll()

.anyRequest().authenticated();

}

@Override

protected void configure(AuthenticationManagerBuilder auth) throws Exception {

// Configure your authentication provider here

}

}

import org.springframework.web.bind.annotation.RequestHeader;

@PostMapping

public ResponseEntity<?> authenticate(@RequestHeader("Authorization") String authHeader) {

String[] credentials = decodeAuthHeader(authHeader);

String username = credentials[0];

String password = credentials[1];

Authentication authentication = authenticationManager.authenticate(

new UsernamePasswordAuthenticationToken(username, password)

);

String token = jwtTokenProvider.generateToken(authentication);

return ResponseEntity.ok(new AuthResponse(token));

}

private String[] decodeAuthHeader(String authHeader) {

String base64Credentials = authHeader.substring("Basic ".length()).trim();

String credentials = new String(Base64.getDecoder().decode(base64Credentials));

return credentials.split(":", 2);

}

import io.jsonwebtoken.Jwts;

import io.jsonwebtoken.SignatureAlgorithm;

import org.springframework.security.core.Authentication;

import org.springframework.stereotype.Component;

import java.util.Date;

@Component

public class JwtTokenProvider {

private final String SECRET\_KEY = "your\_secret\_key"; // Use a strong secret key

private final long EXPIRATION\_TIME = 86400000; // 1 day

public String generateToken(Authentication authentication) {

return Jwts.builder()

.setSubject(authentication.getName())

.setIssuedAt(new Date())

.setExpiration(new Date(System.currentTimeMillis() + EXPIRATION\_TIME))

.signWith(SignatureAlgorithm.HS512, SECRET\_KEY)

.compact();

}

}

